AN AUTOMATIC MORPHOLOGICAL ANALYSIS SYSTEM FOR INDONESIAN

PRIHANTORO

A thesis submitted for the degree of PhD in Linguistics

Department of Linguistics and English Language

Lancaster University

For indeed, with hardship [will be] ease (The Holy Quran 94:5)

فَإِنَّ مَعَ ٱلْعُسْرِ يُسْرًا ٥

Abstract

This thesis reports the creation of SANTI-morf (*Sistem Analisis Teks Indonesia – morfologi*), a rule-based system that performs morphological annotation for Indonesian. The system has been built across three stages, namely preliminaries, annotation scheme creation (the linguistic aspect of the project), and system implementation (the computational aspect of the project).

The preliminary matters covered include the necessary key concepts in morphology and Natural Language Processing (NLP), as well as a concise description of Indonesian morphology (largely based on the two primary reference grammars of Indonesian, Alwi et al. 1998 and Sneddon et al. 2010, together with work in the linguistic literature on Indonesian morphology (e.g. Kridalaksana 1989; Chaer 2008).

As part of this preliminary stage, I created a testbed corpus for evaluation purposes. The design of the testbed is justified by considering the design of existing evaluation corpora, such as the testbed used by the English Constraint Grammar or EngCG system (Voutilanen 1992), the British National Corpus (BNC) 1994 evaluation data[1], and the training data used by MorphInd (Larasati et al. 2011), a morphological analyser (MA) for Indonesian. The dataset for this testbed was created by narrowing down an existing very large bit unbalanced collection of texts (drawn from the Leipzig corpora; see Goldhahn et al. 2012). The initial collection was reduced to a corpus composed of nine domains following the domain categorisation of the BNC)[2]. A set of texts from each domain, proportional in size, was extracted and combined to form a testbed that complies with the design cited informed by the prior literature.

The second stage, scheme creation, involved the creation of a new Morphological Annotation Scheme (MAS) for Indonesian, for use in the SANTI-morf system. First, a review of MASs in different languages (Finnish, Turkish, Arabic, Indonesian) as well as the Universal Dependencies MAS identifies the best practices in the field. From these, 15 design principles for the novel MAS were devised. This MAS consists of a morphological tagset, together with comprehensive justification of the morphological analyses used in the system. It achieves full

[1] http://www.natcorp.ox.ac.uk/docs/bnc2error.htm (last accessed 26/05/2021)
[2] http://www.natcorp.ox.ac.uk/docs/URG/BNCdes.html (last accessed 26/05/2021)

morpheme-level annotation, presenting each morpheme's orthographic and citation forms in the defined output, accompanied by robust morphological analyses, both formal and functional; to my knowledge, this is the first MAS of its kind for Indonesian. The MAS's design is based not only on reference grammars of Indonesian and other linguistic sources, but also on the anticipated needs of researchers and other users of texts and corpora annotated using this scheme of analysis. The new MAS aims at

The third stage of the project, implementation, consisted of three parts: a benchmarking evaluation exercise, a survey of frameworks and tools, leading ultimately to the actual implementation and evaluation of SANTI-morf.

MorphInd (Larasati et al. 2012) is the prior state-of-the-art MA for Indonesian. That being the case, I evaluated MorphInd's performance against the aforementioned testbed, both as just5ification of the need for an improved system, and to serve as a benchmark for SANTI-morf. MorphInd scored 93% on lexical coverage and 89% on tagging accuracy. Next, I surveyed existing MAs frameworks and tools. This survey justifies my choice for the rule-based approach (inspired by Koskenniemi's 1983 Two Level Morphology, and NooJ (Silberztein 2S003) as respectively the framework and the software tool for SANTI-morf.

After selection of this approach and tool, the language resources that constitute the SANTI-morf system were created. These are, primarily, a number of lexicons and sets of analysis rules, as well as necessary NooJ system configuration files. SANTI-morf's 3 lexicon files (in total 86,590 entries) and 15 rule files (in total 659 rules) are organised into four modules, namely the Annotator, the Guesser, the Improver and the Disambiguator. These modules are applied one after another in a pipeline. The Annotator provides initial morpheme-level annotation for Indonesian words by identifying their having been built according to various morphological processes (affixation, reduplication, compounding, and cliticisation). The Guesser ensures that words not covered by the Annotator, because they are not covered by its lexicons, receive best guesses as to the correct analysis from the application of a set of probable but not exceptionless rules. The Improver improves the existing annotation, by adding probable analyses that the Annotator might have missed. Finally, the Disambiguator resolves ambiguities, that is,

words for which the earlier elements of the pipeline have generated two or more possible analyses in terms of the morphemes identified or their annotation.

NooJ annotations are saved in a binary file, but for evaluation purposes, plain-text output is required. I thus developed a system for data export using an in-NooJ mapping to and from a modified, exportable expression of the MAS, and wrote a small program to enable re-conversion of the output in plain-text format. For purposes of the evaluation, I created a 10,000 -word gold-standard SANTI-morf manually-annotated dataset. The outcome of the evaluation is that SANTI-morf has 100% coverage (because a best-guess analysis is always provided for unrecognised word forms), and 99% precision and recall for the morphological annotations, with a 1% rate of remaining ambiguity in the final output.

SANTI-morf is thus shown to present a number of advancements over MorphInd, the state-of-the-art MA for Indonesian, exhibiting more robust annotation and better coverage. Other performance indicators, namely the high precision and recall, make SANTI-morf a concrete advance in the field of automated morphological annotation for Indonesian, and in consequence a substantive contribution to the field of Indonesian linguistics overall.

Published works based on this thesis

Prihantoro. (2021). The Morphological Annotation of Reduplication-Circumfix Intersection in Indonesian. In B. Bekavac, Kocijan, S. K, & S. K. M, *Formalising Natural Languages: Applications to Natural Language Processing and Digital Humanities. NooJ 2020. Communications in Computer and Information Science* (pp. 37-48). Zagreb: CCIS.

Prihantoro. An Evaluation of the Morphological Annotation Scheme for Indonesian Used in MorphInd Program. *Corpora* 16 (3)

Declaration

To the best of my knowledge and belief, I declare that the work presented in this thesis is original and my own work. The material has not been submitted for a degree at this or any other university. The parts of the thesis which have been published as academic articles are listed in the section *Published works based on this thesis*, above.

Prihantoro

Acknowledgement

This work would not have been possible without the support and generosity of many people and organisations around me. I am highly indebted to:

- My wife, my daughter, and my son (Dyka Santi, Yumi, and Dito), to whom this thesis is dedicated. I can't thank them enough for their supports, prayers, and sacrifices. For this reason, I use 'Santi' to name the system, as in 'SANTI-morf'; I prepended 'Dyka', 'Yumi' and 'Dito' to all SANTI-morf's resource file names.
- My parents, who gave me constant encouragements.
- My supervisor, Andrew Hardie who taught me a lot of new things, and patiently guided me to complete this thesis.
- Elena Semino and Jonathan Culpeper for their timely responses and support to help me resolve some urgent issues I experienced during my study.
- NooJ author, Max Silberztein, who also serves as the external examiner on my PhD Viva
- Tony McEnery, the internal examiner on my viva
- *Lembaga Pengelola Dana Pendidikan* (Indonesian Endowment Fund for Education) for its full sponsorship of my PhD program in Lancaster University.
- ESRC Centre for Corpus Approaches to Social Science (CASS) and Linguistic and English Language (LAEL) program, Lancaster university.
- Raffa, Andressa, Irene, Tanjun, and all of my CASS mates.

Needless to say, this is a non-exhaustive list; there are many other people and organisations that are not in this list. Their supports are highly appreciated.

Table of contents

List of tables

List of figures

CHAPTER 1

INTRODUCTION

The aim of this Ph.D. project is to create a novel system for automatic *morphological annotation* of Indonesian, and thus to make an advance on the prior state of the art of computational morphological analysis for this language. As later chapters will show, drawbacks in work in the field so far require the creation of this new system, which is to be called SANTI-morf (*Sistem Analisis Teks Indonesia – morfologi*). This chapter is dedicated to introducing the nature of the project. Sections 1.1 and 1.2 explain certain concepts in *morphology* and *Natural Language Processing*, respectively. I discuss the aims, scope, and procedures of the project in section 1.3, and the organisation of this thesis in section 1.4.

1.1 Morphology

Morphology is the study of the internal structure of words. Haspelmath & Sims (2013: 2-3) explain that morphology provides an understanding of the systematic relations among the elements of words and how words are built from these elements. Words, and the morphological elements of which words are composed, are central in the analysis of morphology.

1.1.1 Words and lexemes

The *word* is often defined briefly as a meaningful linguistic unit which can be realised concretely by sounds or orthography. Crystal (2008:522) also defines *phonological words* and *orthographic words*. He explains that phonological words are particular sequences of sounds associated with particular meanings. We produce phonological words as we speak; they can be represented in phonetic transcription. For instance, when a standard British English speaker says 'word', the utterance can be transcribed as [wɜːd][3].

Orthographic words, on the other hand, are realised by contiguous sequences of letters, often bounded by spaces or punctuation marks. For instance, the orthographic word *word* is written with four consecutive letters or characters, *w-o-r-d,* and is bounded by spaces. These concrete realisations of words in the form of speech or orthography are called *word forms* (Haspelmath & Sims 2013:15, Katamba 1993:18, and Booij 2007:3).

The second key concept in morphology, alongside that of the word form, is the *lexeme*. A lexeme is an abstract unit, consisting of a group of word forms which share a core meaning. Crystal (1988: 276) notes that each abstract lexeme underlies a set of word forms whose variation is grammatically conditioned. *Grammatically conditioned* means that the alternation of the forms or variants is driven by the syntax or morphology (Matthews 2007: 165).

Carnstairs-McCarthy (2002:40) illustrates the concept of variation being driven by syntax or morphology by identifying, as an example, the group of word forms *performs*, *perform*, *performed*, and *performing* as the possible realisations of a single lexeme. He points out that alternation among these forms is driven by their tense, number, and person. For instance, *performs* is used in the context of third person singular present tense, as in *she performs* or *he performs*. The word form *performed* is used in the context of past tense, or perfect/passive participle>, regardless of the number or person, as in *we performed* or *she has performed.* Although these word forms are not identical in grammatical function, they have one core meaning, the basic sense of *perform*: *to do an action or a piece of work.*

Unlike the word form, which is a concrete unit, the lexeme is abstract. An alternative term to lexeme is *lemma.* While *lexeme* and *lemma* refer to the same entity, the latter is more frequently used in corpus linguistics and NLP, henceforth I will use only *lemma.*

[3] Throughout this thesis, phonetic transcription of English represents UK pronunciation, not US.

1.1.2 Morphemes, morphs, and allomorphs

A word can be divided into minimal abstract units carrying meaning or grammatical function. These units are called *morphemes* (Katamba 1993:19), and are often identified by observing the distribution of meaningful sub-units within the word forms of related lexemes; a recurring form with similar functions in each of its settings is a good candidate as a morpheme. Like lexemes, morphemes are abstract units, and thus are not directly present in the word forms. The concrete realisation of a morpheme in the language's orthography or phonetics is referred to as a *morph* (Katamba 1993:24). The identification of morphs within words is illustrated in Table 1.1.

Orthographic form	Phonetic Form	Segmented Phonetic form
caps	[kæps]	[kæp]+[s]
cabs	[kæbz]	[kæb]+[z]
bags	[bægz]	[bæg]+[z]
backs	[bæks]	[bæk]+[s]
bags	[bægz]	[bæg]+[z]
badges	[bæʤɪz]	[bæʤ]+[ɪz]

Table 1.1. Morphs of some English plurals (adapted from Fromkin et al. 2011:267)

Looking at the forms in Table 1.1 allows us to identify a recurrent pattern of interchange among three different morphs, [s], [z] and [ɪz], which share a common grammatical function, namely the plural. These morphs are thus referred to as *allomorphs*, a group of morphs that represent or realise a single abstract morpheme (in this case, the plural morpheme).

Allomorphs of a single morpheme always occur in complementary distribution (Katamba 1993:27). The *distribution* is the total set of contexts where a given form occurs. Hence, complementary distribution means that the environments where different allomorphs appear do not overlap with one another (Haspelmath & Sims 2013:23). For instance, it might be that morph

A occurs only in environment X, and morph B occurs only in environment Y. As the distribution is fixed for each allomorph, it would not be possible for A to appear in Y, or B to appear in X.

The factors that dictate the distribution of allomorphs are called *conditioning*. In the case of the English plural morpheme, as Table 1.2 shows, allomorphs [s], [z], and [iz] are phonologically conditioned by one phonetic property of the final sound of the morph they attach to.

Morph	After	Environment
[z]	[b], [d], [g], [v], [ð], [m], [n], [ŋ], [l], [r], any vowel	Voiced non-sibilant
[s]	[p], [t], [k], [f], [θ]	Voiceless non-sibilant
[ɪz]	[s], [ʃ], [z], [ʒ] , [tʃ], [dʒ]	Sibilant

Table 1.2. Distribution of English plural morphs (adapted from Fromkin et al. 2011:269)

Another type of conditioning is when the choice of allomorph is determined by the presence of some specific adjacent morpheme. This is known as lexical or morphological conditioning. For instance, Katamba (1993:31) contrasts the English plural form o*xen* with regular English plurals. Certain lexical morphemes, like *ox*, do not take the allomorph that would be selected by the usual phonological conditioning of the English plural morpheme, which here would result in **oxes* *[ɒksɪz][4]. Instead, the use of the allomorph *-in* in *oxen* is dictated by the lexical morpheme itself (*ox*).

Another type of conditioning, rarely discussed in the literature on morphology, is orthographic conditioning. Table 1.3 illustrates this using two groups of English adjectives, one group that ends in *y* and another group that ends in other letters. The adjectives in the first group have two orthographic allomorphs, for instance *happy* and *happi*. The allomorph *happy* occurs at the end of a word, whereas *happi* occurs elsewhere, for instance in *happiness*. Unlike the adjectives in the first group, the adjectives in the second group (e.g. *clever*) do not have any

[4] As standardly in linguistics, an asterisk (*) is used to indicate that the form or construction thus labelled is not observed in the language under discussion.

allomorphy; their orthographic form is the same regardless of whether they are at the end of the word or elsewhere.

Unlike allomorphy driven by phonological conditioning, orthographic conditioning affects only the written form. All the adjective morphemes in Table 1.3, whether in the first or second group, have the same phonetic forms regardless of whether the morpheme is word-final or not; only the orthography changes.

1st Group	Allomorph 1		Allomorph 2					
	word end		preceding -*er*		preceding -*est*		preceding -*ness*	
	happy	[hæpi]	*happier*	[hæpɪə]	*happiest*	[hæpɪɪst]	*happiness*	[hæpɪnɪs]
	heavy	[hɛvi]	*heavier*	[hɛvɪə]	*heaviest*	[hɛvɪɪst]	*heaviness*	[hɛvɪnɪs]
	easy	[iːzi]	*easier*	[iːzɪə]	*easiest*	[iːzɪɪst]	*easiness*	[iːzɪnɪs]
2nd Group	No allomorphy							
	clever	[klɛvə]	*cleverer*	[klɛvərə]	*cleverest*	[klɛvərɪst]	*cleverness*	[klɛvənɪs]
	hard	[hɑːd]	*harder*	[hɑːdə]	*hardest*	[hɑːdɪst]	*hardness*	[hɑːdnɪs]
	soft	[sɒft]	*softer*	[sɒftə]	*softest*	[sɒftɪst]	*softness*	[sɒftnəs]

Table 1.3. Two groups of morphemes preceding *-er, -est*, and *-ness* (adapted from Duran & Katamba 2014, with additional examples)

Orthographic conditioning is rarely given detailed attention in the morphological literature, due to the general primacy of speech over writing in linguistics. I have discussed this type of conditioning here because my project deals with written language, where only orthographic changes, rather than phonetic changes, are directly present.

1.1.3 Categorisations of morphemes

1.1.3.1 Free versus bound morphemes

An important distinction among morphemes is whether or not a given morpheme can stand as an independent word. Some morphemes, like the English plural morpheme (*-es/-s*), always attach to another word, to nouns in the case of the plural, as illustrated in example (1.1). This plural morpheme cannot occur alone as a word (1.2), while the nouns that it attaches to *can* occur as independent words (1.3).

(1.1) *buses, boxes, smiles*
(1.2) **es, *s*
(1.3) *bus, box, smile*

Morphemes that can stand by themselves as words are called *free morphemes*. Those that cannot stand alone, but need to attach to other morphemes, are called *bound morphemes* (Coates, 1999:3). In (1.3), each of the word forms consists of a single morpheme. These are thus free morphemes, being able to stand as independent words. But neither allomorph of the plural morpheme can occur alone as a full word; thus, it is a bound morpheme.

1.1.3.2 Lexical versus grammatical morphemes

Morphemes can also be categorised according to whether they carry semantic content or grammatical function. Morphemes that carry semantic content usually have clear informative meanings in themselves; they are called *lexical morphemes* (Katamba 1993:41). *Grammatical morphemes*, by contrast, encode grammatical categories, functions, or syntactic relations (Lipka 1992:70).

	Free	Bound
Lexical	noun: *hymn* adjective: *clever* verb: *link*	*hydro-* in *hydrology* *geo-* in *geography* *cran-* in *cranberry*
Grammatical	article: *a, an* pronoun: *this, that* conjunction: *and, if*	-*s* (plural) in *hymns* -*ed* (past) in *linked*

Table 1.4. Samples of lexical and grammatical morphemes (drawn from Katamba 1993:41 and Coates 1999:30)

Table 1.4 illustrates that both lexical and grammatical morphemes can be either free or bound. Free lexical morphemes are known as *content words* (nouns, adjectives, verbs and adverbs in English), while free grammatical morphemes are called *function words*, such as articles, pronouns, or conjunctions in English (Fromkin et al. 2011: 93). Such categories of free morphemes as noun, article and so on, are usually described as *Part of Speech* (POS) categories, or lexical categories, or word classes, because morphemes in these categories can stand freely as independent words.

Table 1.4 also shows that some lexical morphemes are bound; they cannot stand as independent words. To form a word, these morphemes have to be attached to another morpheme. In the same vein, some grammatical morphemes are also bound, such as the English plural morpheme, as previously discussed.

1.1.3.3 Affixes and bases

Grammatical bound morphemes like English plural *-s* are termed *affixes*. The units to which affixes can be attached are called *bases* (Katamba 1993:45). Bases can be composed of a single morpheme, or a combination of morphemes. *Affixes* can be categorised based on their form and their function, as outlined in Table 1.5.

Formal category	Prefix	Suffix	Infix	Circumfix
Examples (with functions)	English *un-* (negation) *re-* (repetition)	English *-s* (plural) *-ed* (past tense)	Tagalog *-um-* (verb nominaliser)	Javanese *ke—an* (adjective nominaliser)

Table 1.5. Affix categorisation (adapted from Katamba 1993:45-51 and Ewing 2005:24)

An affix can be categorised in terms of its position relative to the base, that is its form, and in terms of its function (Aranoff & Fudeman 2011:3-4). A *prefix* is an affix that precedes the base, such as English negative *un-* (e.g. *unequal, unable, undo*). A *suffix* is an affix that follows the base, such as English plural *-s*. An *infix* attaches within the base, for example Tagalog nominaliser *-um-* (e.g. *sulat* 'write' > *sumulat* 'writer). A *circumfix* is an affix that surrounds its base, for instance Javanese *ke—an,* which nominalises an adjective base (e.g. *sugih* 'rich' > *kesugihan* 'wealth'; Ewing 2005:24).

1.1.3.4 Inflectional and derivational affixes

Affixes can additionally be categorised as inflectional or derivational. Before discussing this distinction, it is necessary to define the two different kinds of word formation: inflection and derivation.

Booij (2007:99) defines *inflection* as the morphological marking on a base which produces a set of word forms with the same meaning as the base. Thus, word forms created by inflection are all realisations of a single lexeme, as discussed in section 1.1.1. *Derivation*, on the other hand, is the process of creating new lexemes (Katamba 1993:47). This is because derivation produces forms which are either distinct in meaning from their bases (and, thus, realise different lexemes than their sources), or else distinct in lexical category (and, thus, realise different lexemes than their sources).

In many languages, both inflection and derivation are coded by affixes. Hence, affixes can be referred to as inflectional or derivational affixes. Some English examples are given in Table 1.6 and Table 1.7.

Base	Inflectional affix	Word
perform	*-s*	*performs*
	-ed	*performed*
	-ing	*performing*

Table 1.6. Some inflectional affixes with the base *perform*

Base	Inflectional affix	Word
happy	*un-*	*unhappy*
	-ness	*happiness*

Table 1.7. Some derivational affixes with the base *happy*

The meaning of word forms such as *performs, performed,* and *performing*, as exemplified in Table 1.6, is not different from the meaning of their base, *perform*; thus, they are forms of (or realise) a single lemma. For this reason, *-s, -ed* and *-ing* are categorised as inflectional affixes.

The derivational affixes in Table 1.7 form words which are distinct in meaning or in POS category from the base to which they are affixed. The meaning of *unhappy* is not the same as the meaning of its base, *happy*, albeit they remain related. On the other hand, *happiness* is distinct in POS category to its base *happy*. Thus, these forms represent different lexemes from those represented by their bases. For these reasons, *un-* and *-ness* are categorised as derivational affixes.

1.1.3.5 Roots, bases, and stems

The *root morpheme* (or just *root*) is the irreducible part of a word without anything attached to it (Katamba 1993:41). The root possesses the core meaning of the word. Root morphemes can be bound or free, as the examples in Table 1.8 show.

	Free	Bound
Root	*faith* (*faiths, faithful, faithful, unfaithfulness*) *hard* (*hardship*, *hardness*)	*-mit* (e.g. in *permit*) *-ceive* (e.g. in *receive*)
Base	*permit* (*permits, permitting*) *receive* (*receives, receiving*) *faith* (*faiths, faithful, faithful, unfaithfulness*) *faithful* (*faithfulness, unfaithful*) *hard* (*hardship*, *hardness*) *hardship (hardships)*	*recept-* (e.g. in *receptive*) *permiss-* (e.g. in *permission*)

Table 1.8. Examples of English roots, bases, and corresponding affixed word forms (in brackets) (adapted from Katamba, 1993:41)

Faith and *hard* are examples of free roots, able to occur as independent words, but also as bases for affixation, both inflectional (e.g. faiths) and derivational (e.g. faithful) However, not all bases are also roots, or even morphemes of any kind; for example, *permit* and *receive* can be bases for affixation, but they are not morphemes, but rather words composed of multiple morphemes (*per-mit, re-ceive*). The roots of *permit* and *receive* are *-mit* and *-ceive*, respectively. These are examples of bound roots, because they cannot occur as independent words.

Word forms created by affixes may be bases for further affixation, including the non-root bases in Table 1.8. For example, the derivational suffix *-ship,* added to the base *hard,* produces *hardship*; *hardship* is subsequently the base for *hardships*, produced by appending the inflectional suffix *-s*. Likewise *faith* combines with derivational suffix *-ful* to make *faithful*, which in turn is the base to which derivational suffix *un-* is added to produce *unfaithful*.

A *stem* is a base to which an inflectional affix can attach (Katamba 1993:45-46). Thus, it consists of minimally of a root (e.g. *tie*, to which inflectional suffix *-s* can attach to form *ties*) but may also be composed of a root plus derivational morpheme(s) (e.g. *un-tie*, to which *-s* can attach to form *unties*) or multiple roots in a compound (e.g. *tie-break*, to which derivational suffix *-er* can attach to form *tiebreaker*).

1.1.4 Presenting the internal structures of words

The categorisation of morphemes helps linguists to explain the *internal structure of words*, which is one of the primary aims of morphology. Morphological categories can be annotated on linguistic examples using some specialised notation to present the internal structure of the word under discussion. In this section, I expound three ways of presenting this information, namely *bracket notation, tree diagrams,* and *glossing*.

1.1.4.1 Bracket notations and tree diagrams

In the morphological literature, two common notations used to represent the internal structure of words are bracket notation and tree diagrams (Delahuntey 2010:138-140). These notations are particularly useful for characterising how bases receive multiple affixation.

The first step for either notation is to separate out the morphs so they are isolated within the complete word. For instance, the word *unsystematic* can be segmented to the following sequence of morphs: *un-, system, -atic*[5] (Fromkin et al. 2011: 93). Subsequently, labels for each morph's morpheme type can be assigned: AF (affix) to *un-* and *-atic* and N (noun) to *system*.

The principle that underlies both bracket notation and tree diagrams is the attempt to correctly lay out the patterns of morpheme combination within the word. This relates to one of the tasks of morphologists, which is to describe the grammar of words within some language. Valid affixation patterns are formulated on the basis of observed examples, and should capture all possible affix combinations with roots of all different categories. Violating the patterns thus established usually results in unnatural forms, that is, morpheme combinations that do not form an actually existing word. Morphological notations help morphologists in this endeavour. The earlier example, *unsystematic* (two affixes and one base), might result from two possible affixation patterns, given in (1.4) and (1.5).

(1.4) base + *atic*
(1.5) *un* + base

In both these patterns, the base is given as an unfilled variable slot. In the particular case of *unsystematic*, *system* is the base (and root). So next we determine whether or not *system* can be a base in each of these two patterns. If we apply the two patterns unrestrictedly to *system*, pattern (1.6) will produce **unsystem*, which is not actually a word. However, pattern (1.4) produces *systematic*, which if then used as the base for pattern (1.5), generates the valid output *unsystematic*.

The base for *systematic* is the noun *system*, while for *unsystematic* it is the adjective *systematic*. Observation of other words that carry the two affixes under consideration will show that these affixes cannot occur with any base, but only a base of the appropriate POS (*un-* does also attach to nouns and verbs, but with slightly different meanings). We can capture this by rewriting the two patterns as (1.6) and (1.7) to use the POS symbols N and A for the variable slots. These patterns may be observed in word forms such as *problematic, idiomatic* or *symptomatic* for (1.6) or *unusual, unfair* or *unacceptable* for (1.7).

[5] It is possible to analyse *-atic* as a combination of two morphemes: *un-system-at-ic*. My segmentation follows Fromkin et al. (2011:93), who treat *-atic* as a single suffix.

(1.6) N + atic
(1.7) un + A

Words formed by multiple affixations arise from an *ordered* sequence of changes, that is, a sequence where each change has a specified priority. The foregoing account of the formation of *systematic* shows that the affixation patterns in (1.6) and (1.7) must have that order of precedence. The concept of precedence helps describe the formation of words in terms of the building blocks of the patterns. Both bracket notation and tree diagrams are elaborations of this basic idea.

In bracket notation, the operation of the highest-precedence pattern (1.6) is presented in the most deeply nested brackets. In a tree diagram, this pattern is at the lowest level of the hierarchy. The adjective thus formed, *systematic*, becomes the base in the next layer outwards of brackets, and at the next level up in the tree. This new base takes *un-* , following pattern (1.7), to generate *unsystematic*, linked to both the node at the top of the tree diagram in Figure 1.1 and the outermost of the nested bracket notation in example (1.8).

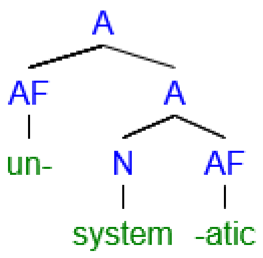


Figure 1.1. Tree diagram[6] for the structure of the word form *unsystematic* (adapted from Fromkin, et al. 2011:93)

(1.8) [A [AF un-] [A [N system] [AF -atic]]]

This can be expressed more formally as follows. In the tree diagram, *system* and *–atic* are placed at the bottom of the hierarchy. Their categories, N and AF, are annotated above them.

[6] This image was generated using the following tree generator http://mshang.ca/syntree/ (last accessed 17/05/2021).

Together, these form an adjective base (labelled A). This base takes affix *un-* (labelled AF), forming adjective *unsystematic* (A) at the top of the hierarchy.

Similarly, in bracket notation, each category is annotated next to the left bracket of the pair that encloses the corresponding morph or combination. The two innermost pairs of brackets, [N *system*] and [AF *–atic*], are nested within another pair of brackets, marked (A), which indicates the formation of *systematic*. This then provides a base for *un-* (AF), with the last pair of brackets indicating the formation of *unsystematic,* also an adjective, as indicated by the label (A) for the outermost pair of brackets.

1.1.4.2 Glossing

While the general aim of bracket notation, tree diagram, and glossing is presenting the internal structure of words, the specific aim of glossing is slightly different. The purpose of *glossing* is to facilitate the description of the structure of a language which might not be familiar to readers. Thus, there are two languages involved in glossing; the language of the observed example, and the language used to describe the example. Morphological phenomena in the observed example, particularly grammatical categories, often lack precise translations in the language of description. The gloss is a way to indicate the position and function of each morph(eme) precisely, together with the morphological function(s) it expresses. Glossing helps linguists present all such morphological features overtly, even if the language used to describe the observed example does not have those features.

A glossed example is presented across three lines. Lehmann (2004:1831) explains that the first line gives the sequence of linguistic unit(s) in the observed language that serves as an example. This is Romanised if not originally in the Latin alphabet. This sequence can consist of just one word or more, with morph boundaries inserted as necessary. A literal morph-by-morph translation, including a label for the categorisation of each grammatical morpheme, is annotated on the second line. Lexical morphemes (see the discussion of roots and bases in 1.1.3.5) are translated literally, and thus are not given grammatical labels. The third line gives a free translation in the language of description. Example (1.9) demonstrates the glossing of a Korean sentence (reproduced from Lehmann 2004:1842):

(1.9)	*Toli-neun*	*kae-hako*	*cal*	*non-ta*	(Source: Korean)
	Toli-TOP	dog-ADD	often	play:PRS-DECL	(Gloss)
	'Toli likes to play with the dog.'				(Translation: English)

The words on the first line of the example are divided into morphs with morph breaks (-). The categories of these morphs, or morphemes, are annotated on the second line with grammatical labels that indicate their function in the observed language, which might be different from the morphology of the equivalent words in the English free translation.

The glossed example tells us that *Toli-neun* is a Korean word composed of two morphemes. *Toli* is a proper name, and so is neither literally translated (see above) nor given a category label on the second line. It is also left untranslated in the third line. Affix *-neun* is given the label TOP (topic marker) on the second line; in Korean, the category of topic is morphologically marked (Lee 1999: 317-342). A similar interpretation applies to the other three words in the example.

One suggested standard guide for glossing, and for category labels to be used in morphological annotation, is the Leipzig *Interlinear Morpheme-by-Morpheme Gloss*[7] format*,* derived from Lehmann (1982: 199-124) and Croft (2003: xix-xxv). However, some authors follow other standards, or even develop their own labels or format. Particularly in projects related to automatic morphological annotation, the formats and analytic labels may be quite different from those used in this standard glossing format.

[7] https://www.christianlehmann.eu/ling/ling_meth/ling_description/grammaticography/gloss/index.php (last accessed 26/05/2021)

1.1.4.3 Automatic morphological annotation

Another type of representation of the internal structure of a word is that produced by a computer program called a *Morphological Analyser* (MA). Here, morph breaks and category labels are assigned automatically by this program, instead of written in manually by human linguists. In the output format of the Turkish MA known as TR Morph[8] (Coltekin 2014:1079-1080), for instance, the Turkish word *evimden* 'from my house' is presented as in example (1.10).

(1.10) ev<N><P1S><ABL> (Oflazer et al. 2018:220)

Oztaner (1996:20) describes *evimden* as follows. The noun base *ev* <N> is followed by two inflectional morphemes. The first suffix is *-im,* which is the first person singular possessive suffix <P1S>. The second is the ablative suffix <ABL> *-den*. This indicates that TR Morph output represents the structure using an alternative method to bracket notation. Only the morph of the base *en* is actually present. The morphs that indicate the base's inflectional suffixes are left implicit, indicated only by the analytic labels, which are explicitly presented in consecutive order after that of the base morph *en*.

Other MAs use different output formats. For instance, many use a simple slash to demarcate a morph from the corresponding analytic label. The Korean Morphological Analyser, or KOMA[9], uses this format. If this format were applied to Turkish, the analysis of *evimden* 'from my house' might look like example (1.11).

(1.11) *ev*/N *im*/P1S *den*/ABL

The output generated by an MA can be used by another program for further language processing tasks. For instance, a corpus-processing program would typically import MA output to allow users to perform automated corpus queries, including queries based on the forms of the

[8] http://coltekin.net/cagri/trmorph/index.php (last accessed 26/05/2021)
[9] http://kle.postech.ac.kr:8000/demos/KOMA_KTAG/ekma.html (in English; last accessed 17/05/2021) and http://nlp.kookmin.ac.kr/HAM/kor/ham-intr.html (in Korean; last accessed 17/05/2021)

morphs and/or their analytic labels.

A corpus-processing program often requires input in its specific data format, including for morphological annotation. This is a possible reason why the output format of MAs is not as standardised as are bracketing notation, tree diagrams, and glossing. The choice of formats depends on the overall goal of the NLP or corpus analysis system.

1.2 Natural language processing

1.2.1 Common NLP applications

One of the most common NLP applications is the *Part of Speech* (POS) tagger. A POS Tagger is a program that automatically links each word in a text with an analysis of its part of speech (Voutilanen 2003:210). The depth of the analysis may range from basic lexical categorisation (into nouns, verbs, prepositions, conjunctions, and so on) to more detailed subcategories within these POS (proper nouns, infinitive verbs, locative prepositions, coordinating conjunctions, and so on). Sometimes the analysis performed by a POS tagger is called *morphosyntactic analysis,* as it often includes inflectional features such as person or number, which are morphologically marked but deeply involved in aspects of syntax.

Another common NLP application is the *Morphological Analyser* (MA), which performs automatic morphological analysis (discussed previously in 1.1.4.3). After segmenting each word in its input into morphs, an MA links each morph to an analysis of relevant morphological features and categories (Oflazer 1999: 175).

MAs are particularly useful for languages where a word is *typically* composed of multiple morphemes, such as Turkish, Finnish, or Arabic. For instance, an MA can be used to isolate affixes from their bases, or to identify roots. In section 1.1.3.1, I discussed how this could be performed manually by linguists to analyse the internal structure of words. An MA can automate this process, partially or entirely.

POS taggers and MAs often feed word category information to other NLP applications, such as a *grammatical parser,* sometimes referred to just as a *parser*. Parsing is the analysis of sentence structures (Samuelsson & Wiren, 2000: 59). A parser analyses a sentence using a set of given rules (a grammar) or probabilistic computations. Both rules and probabilistic computations need to refer to the POS information to select a correct, or likely, parse.

To represent sentence structure, a parse breaks down a sentence into the smaller constituents, such as clauses or phrases, of which the sentence is composed. For instance, given the English sentence *the dog ran*, a rule-based parser can analyse *the dog* as a noun phrase (NP) composed of a determiner and a noun by reference to a rule expressing that structure in terms of POS tags. Alternatively, *the dog* can be determined to be an NP by probabilistic means, since this analysis is highly probable for the sequence *the dog* or indeed any sequence POS-tagged as determiner then noun. Other possible phrase types, such as verb phrase (VP) or prepositional phrase (PP), would have lower probability due to less frequent presence (or absence) in whatever data the probabilistic system generated its statistics from.

Some NLP applications are designed for more than one task. The Stanford Parser (Klein & Manning, 2003), for instance, is an integrated POS tagger and parser. Other examples are Intex (Silberztein 1993;1997) and NooJ (Silberztein 2003). Some NLP applications (again, Intex and NooJ are examples) are multi-language; they are not programmed to work with any specific language but rather allow users to build and utilise the resources necessary for morphological analysis, POS tagging, and/or grammatical parsing in different languages.

1.2.2 Token

There are at least two reasons why the *token* is a key concept in NLP. First, the process by which tokens are created, *tokenisation*, is an early step in the majority of text processing systems. Second, *token* is a basic term frequently used in the definition of other, more advanced

NLP concepts. For these reasons, it is important to clarify this concept.

The process by which tokens are identified in running text, *tokenisation*, can be described as the segmentation of the raw sequence of characters in a digital text into meaningful units for the analysis targeted by the NLP application (Mikheev, 2003: 201). These segments of the text's stream of characters are the units of analysis, which are called tokens.

The most common form of tokenisation is word segmentation. Once the word tokens in a text have been identified, that text is ready for further processing or analysis. In many languages, the presence of spaces that demarcate one word from another is an important cue for the tokenisation. Tokenisation can be more challenging if space characters are not used to separate the words, as is the case in the so-called *unsegmented languages,* such as Chinese and Thai, whose writing systems do not indicate word breaks explicitly. Another form of tokenisation, as section 1.1.3 discussed, is the isolation of morphs within a word, for the purpose of subsequently identifying the morphemes. Since at this level we consider the morpheme (instead of the word) to be the token, i.e. the unit of analysis, morpheme-level tokenisation cannot rely on spaces as separators between tokens.

The fact that either the word or the morpheme can be treated as the token shows that the precise definition of *token* depends on what units are to be processed by the NLP application. A sequence of characters which is considered to be a single token by one application is not necessarily treated as a single token by another application. For instance, an English MA would typically analyse *buses* as two separate tokens, *bus* and *es*, as the word *buses* is composed of two morphemes. Unlike an MA, an English POS tagger would treat *buses* as a single token, because this latter application is designed to categorise words, not morphemes.

1.2.3 Annotation

Annotation is a key concept in NLP (as well as in corpus linguistics) because it is the core task of many important NLP applications. This includes those applications discussed in the previous section. In a broad sense, annotation is the combination of a text with an analysis (or analyses) of the text (Silberztein 2003: 206). Wilcox (2009:1) defines *linguistic annotation* specifically as the assignment of linguistic features to appropriate linguistic elements in a text (such as words or sentences). From an NLP perspective, annotation can thus also be defined as the integration of tokens with analytical labels that code for the linguistic features in question. Such labels express some item of interpretative linguistic information regarding the tokens they are attached to. A basic example is given as (1.12), in which is shown a single sentence grammatically annotated by the CLAWS POS Tagger (Garside 1987). Here, for instance, tag NN2 is applied to the token *buses* to indicate that the CLAWS tagger categorises *buses* as a plural noun.

(1.12) There_EX are_VBR two_MC buses_NN2 ._.

Each underscore in (1.12) delimits a token and its analytical label. These analytical labels are usually called *tags*. The full collection of tags used for a particular task, in this case annotating a corpus for POS, is called a *tagset* (Bird 2009: 179). The process of annotating a tag to each token, as shown in 1.12, is therefore often called *tagging*. For instance, Voutilanen (2003:220) states that tagging means "automatic assignment of descriptors or tags to input tokens." This is equivalent to the definition of *annotation* above, and indeed, for the purposes of this thesis, *tagging* can be considered synonymous with *annotation*.

Systems of tags may differ from one application to another. For instance, as we have seen, NN2 in the CLAWS tagset signifies plural common nouns. But in the Penn tagset, the same category is indicated by tag NNS. Thus, users must carefully verify their understanding of what the tags mean by consulting the tagset documentation[10] before conducting further processing or undertaking research based on the tagged data.

[10] I use the term *tagset documentation* following Bird (2009: 180). However, other terms such as tagset *manual* or *specification* are also in use. Such documentation usually contains a list of tags used in the system, the full label that each tag abbreviates, and sometimes examples of tokens that would receive each tag. More extensive documentation may include discussions of the criteria by which tokens are deemed to be in one category versus another and/or guidelines for deciding borderline cases.

1.2.4 Lexicon, rules and annotated corpus

Lexicon, *rules,* and *annotated corpus* are key concepts in NLP. These terms refer to types of source information used by NLP applications such as MAs, POS taggers or parsers to perform annotation. For this reason, together they are commonly referred to as annotation *resource*s.

1.2.4.1 Lexicon

An NLP *lexicon* is a listing of items at some level of linguistic analysis, each of which is associated with a potentially wide range of linguistic information. These items are sometimes referred to as *entries*, because they are usually organised in a list, just like dictionary entries. In general, Litkowsi (2005:753) argues that a lexicon includes a wide array of information associated with entries (words, phrases or concepts). However, a lexicon for automatic morphological analysis could include morphological entries (see Silberztein 2016:220). The type of information associated with lexicon entries may include POS, inflectional features, or other grammatical properties. The nature of some probabilistic systems also requires lexicon entries to be associated with statistical information (Schiller & Kartunnen 1999:136).

NLP scholars sometimes use other terms to refer to a lexicon, such as *lexical database*. The term *dictionary* can also refer to an NLP lexicon, as the function of a lexicon is similar to that of a dictionary: to store (information regarding) the vocabulary of a language. To distinguish an NLP dictionary or lexicon from a conventional dictionary, sometimes the term *machine-readable dictionary* is used.

A lexicon can be used by an NLP application to perform annotation. In most POS taggers, for instance, a lexicon plays a crucial role (Voutilanen 1999:6). One subtask of a tagger is to identify, for a token in a text, which entry or entries, if any, in the system's lexicon correspond(s) to the form of that token. If the token does match an entry, the information in that entry can be assigned as a tag or tags. For instance, when an English POS tagger encounters *the*, the lookup module checks whether an entry for this form is present in the lexicon. If such an entry is found, as of course is likely, the system tags the token with the analyses present in that entry. In this case, the information in the lexicon entry for *the* would include a determiner or article tag (e.g. DET).

A lexicon can also play a crucial role in the earlier tokenisation phase, especially in an MA, where the tokens are not delimited by spaces and tokenisation is thus not trivial. The MA would check whether sub-sequences of characters inside a word string match entries in the lexicon, to determine whether to analyse those sub-sequences as separate tokens.

1.2.4.2 Rules

Chun-Hsien & Honavar (2000:880) describe two major types of lexicons: *root lexicons* and *full-form lexicons*. A full-form lexicon includes (ideally) all possible word forms (variants or inflected forms of their lexeme) in the language or variety the system targets. For instance, a full-form English lexicon would ideally contain an entry for *help* as well as entries for *helps, helping,* and *helped*. A root lexicon, on the other hand, only contains entries for roots (or, perhaps, a related type of entity such as stems, lemma, or uninflected forms). A root lexicon is more compact than a full-form lexicon, but requires the software to support a *rule* system in order to handle polymorphemic words. A *rule* in this sense can be defined as a formalised word formation pattern or, sometimes, a joined set of such patterns.

Silberztein (2003:97) gives an example, reproduced here as (1.13), of a possible entry for `<help>` in an NLP system's root lexicon. This entry is associated with the POS information `<V>` and an inflectional code `<FLX=ASK>`. ASK is a symbol which refers to a set of inflectional rules defined as in (1.14).

(1.13) `help,V+FLX=ASK`

(1.14)
```
ASK = <E>/INF | <E>/PR+1+2+s | <E>/PR+1+2+3+p | s/PR+3+s | ed/PP |
ed/PRT | ing/G ;
```

The combination of the root entry for `<help>` and this rule allows the system to recognise not only tokens of *help* but also tokens of *helps*, *helped*, and *helping*. For each matching token, the rule attaches the relevant inflectional features to the token as annotation tags. This process is also called *lexical analysis*. Sproat (2000: 37) defines lexical analysis as the determination of (lexical) features for each word in the text. Silberztein (2016:210) points out that the result of a lexical analysis may be ambiguous, because multiple possible (sets of) features can be annotated. So, for instance, in rule (1.14), the fragment `ed/PP | ed/PRT` assigns two tags to each instance of the form *helped*: PP (past participle) or PRT (preterite or past tense). This means that the result of lexical analysis is ambiguous between PP and PRT. In the sentence *I helped you,* the correct analysis is PRT, and thus the other analysis (PP) should ideally be removed. The opposite is the case in the sentence *I have helped you*.

1.2.4.3 Annotated corpora

In addition to lexicons and rules, an *annotated corpus* can also be a resource for automated annotation; this is, therefore, also a key NLP concept. A *corpus* is a collection of natural language data, likely to be in the form of machine-readable texts (McEnery & Wilson 2001:31); thus, an annotated corpus can simply be defined as a collection of tagged or annotated texts. Taggers or parsers can be created by exploiting an annotated corpus. Often, a tagging system uses annotated corpus data to build a probabilistic model of some kind. An annotated corpus being used in this way can be called a *training corpus*, from its utilisation as training material to build the model (Brill 1999:266). Subsequently, the model derived from this corpus is used by the system to perform or to enhance annotation.

Annotated corpora also serve other functions, as van Halteren & Voutilanen (1999: 111) point out: an illustration of the tagging scheme; a model with which to build hand-written rules; and testing material for a final evaluation of a complete system. Thus, an annotated corpus is still useful even when the system does not employ a probabilistic method for annotation.

1.2.5 Disambiguation

Disambiguation is the final task of NLP applications such as POS taggers, MAs or parsers[11]. Disambiguation is the removal of tags deemed likely to be incorrect, or the selection of the most likely correct tag, for a token which has initially received multiple tags from lexical analysis. Voutilanen (2003:226-227) illustrates this concept by considering the lexical analysis of the word *design*, which receives a noun tag and a verb tag. In example (1.15), which I reproduce from Voutilanen, the proper category for *design* is noun. However, in the contrasting example that I give in (1.16), the correct category is verb.

(1.15) The *design* of the guesser is very complex
(1.16) Computer scientists *design* various guessers

Lexical analysis will assign at least two tags to each instance of *design*: noun (NN1) and verb (VV0) [12]. A tagger can utilise contextual cues to resolve this ambiguity. For instance, an article tends to precede a noun rather than a verb. Thus, selection of the noun tag can be conditioned on the presence of *the* before *design*. If this is implemented, *design* in (1.15) is likely to receive the NN1 tag. Meanwhile, *design* in (1.16) is likely to receive the VV0 tag, as it is preceded by a plural noun, *scientists*.

This exemplifies one of several methods of resolving ambiguities, that is, the use of *disambiguation rules*. These rules take into account relevant contextual cues, such as a preceding or following word or tag, as previously explained. The rules must be formally expressed in a format that the system can understand to enable the system to remove incorrect tags. These rules can be built manually, in which case they are known as *handcrafted* rules (Voutilanen 1999:217-247).

[11] Disambiguation by parser programs was already introduced in section 1.2.1
[12] The tags NN1 and VV0 (singular common noun, base-form lexical verb) are drawn from the CLAWS tagset (Garside 1987).

Disambiguation can also be performed by exploiting a probabilistic model, such as a Hidden Markov Model (El-Beze & Merialdo 1999:263), or a machine learning approach (Daelemans 1999:285). A probabilistic tagger consults its model in each case of an ambiguous token in order to determine which tags to remove or to retain in the analysis. The model is usually derived from an annotated corpus (as discussed in 1.2.4). It is also possible to combine handcrafted rules and probabilistic models (Brill 1999: 248-262).

1.3 Aims, scope and procedures of the thesis project

As I noted at the outset of this chapter, the primary aim of this project is to create a novel system for automatic morphological annotation of Indonesian, which will be named SANTI-morf. To achieve this primary aim, I will proceed by addressing a number of subsidiary aims. These three subsidiary aims will be completed in three stages; each stage will lead to one or more specific outputs produced in fulfilment of the corresponding subsidiary aim, as laid out in Table 1.9.

Stage	Subsidiary Aim	Output
1 Preliminaries	To provide a description for this project containing aims, scope and procedures, a brief introduction of the target language's structure, and the creation of a testbed for the system.	Project description and testbed
2 Scheme creation	To design a novel Morphological Annotation Scheme (MAS) for use in this project, in the form of a document that discusses theoretical issues, and a choice of annotation scheme.	Document defining the MAS (as a chapter of this thesis)
3 Implementation	To evaluate the performance of the state-of-the-art MA for Indonesian, and identify its limitations; to build a set of morphological annotation resources for a novel MA; to perform and to evaluate automatic morphological annotation using the created resources, with comparison to the state-of-the-art system.	SANTI-morf system and results of evaluation exercise

Table 1.9. Subsidiary aims and corresponding desired outputs of this thesis

Some aspects of annotation are beyond the scope of this project. First, in terms of the level of annotation, I focus on morphological annotation. Thus, syntactic annotation (as a parser does), morphosyntactic annotation (as a POS tagger does), and other types of annotation that are beyond the morphological level are excluded. Second, this project does not aim to address issues in morphological theory, as it focuses solely on developing a practical application. Third, the novel MA system will be designed primarily to handle written rather than spoken data. Fourth, the morphological annotation is aimed at Indonesian specifically and not any other varieties of Malay (see discussion in Chapter 2).

The procedures to be carried out for each stage of the overall project are as follows:

I. Preliminaries

  a. introduce key terms used in the thesis
  b. introduce Indonesian, particularly Indonesian morphology
  c. build a corpus of Indonesian for use in later stages of the project

II. Scheme creation

  d. review existing Morphological Annotation Schemes (MASs), and identify best practices for the creation of MASs
  e. devise a novel MAS for Indonesian based on these best practices

III. Implementation

  f. evaluate the output and performance of the present state-of-the-art MA system for Indonesian
  g. choose a framework for the novel system via a literature review of work in the field of MA creation
  h. survey existing annotation software, and determine which program is optimal for applying the novel MAS
  i. build annotation resources following the MAS whose format is compliant with the chosen annotation software
  j. apply and evaluate the performance of the resulting system

k. compare and contrast the performance of the novel system with that of the existing state-of-the-art system (see above).

1.4 Organisation of this thesis

The thesis consists of nine chapters. Chapters 1 and 2 present the first stage of the project, the preliminaries. Chapter 1 has introduced key concepts in morphology and NLP. It has also introduced the primary aim, subsidiary aims, and scope of this thesis.

Chapter 2 will introduce aspects of the Indonesian language and its morphology which are relevant to morphological annotation. This chapter will also describe an Indonesian corpus built for use in this project. Textual data drawn from this corpus will be used for two purposes: to evaluate an existing MA (in Chapter 5) and to test the performance of SANTI-morf (in Chapter 7).

Chapters 3 and 4 present the second stage of the project, annotation scheme creation. In Chapter 3, I will conduct a literature review of schemes of analysis for morphological annotation in different languages (but with a focus on languages with a high degree of agglutination). The goal of the review is to identify best practices for such MASs. The findings from this chapter, particularly on the MASs used by existing Indonesian MAs, will be important to set a benchmark for an eventual evaluation of SANTI-morf to underpin my claim that it represents an advance on existing systems.

In Chapter 4, building on the outcomes of Chapter 3, I will present the development of my MAS, the annotation scheme that SANTI-morf will use. One of the primary aims is to justify the choice of morphological features tackled in this project. Here, I will provide justifications for why certain features need to be excluded or be included, and why some annotation styles are preferred over others.

Chapters 5 to 7 present the third and final stage of the project, implementation. Chapter 5 will first review existing Indonesian MAs, identifying one of these as the state-of-the-art MA whose performance any novel system must aim to surpass. This state-of-the-art system will then be subject to an extensive evaluation. It will be applied to a sample of the corpus and its performance will be measured. A thorough examination of the outputs will further allow it to be determined which aspects of the state-of-the-art MAs still need improvement. This chapter will provide important input data for later chapters on the creation of SANTI-morf.

Chapter 6 will discuss the formalism and software tool used in this project. First, a review of relevant background in the theory of formal language will be given to clarify the basic concepts that underpin contemporary MAs, including especially regular grammars, regular expressions, and regular relations, and the related notions of Finite State Machines (FSMs) and other automata. Relevant concepts in early generative morphology will also be reviewed. Then, work on computational MAs beginning in the 1970s will be surveyed, in order to illustrate the emergence of Koskenniemi's seminal Two Level Morphology (TLM) model (Koskenniemi 1983), which has influenced nearly all subsequent MA systems. On the basis of this review, I will justify my choice of one particular formalism and program for implementation of the novel MA over other possible candidates.

In Chapter 7, I will describe the architecture of SANTI-morf. I will also describe the creation and organisation of the SANTI-morf resources for use in the program of my choice. I will apply the completed SANTI-morf system to the testbed (see section 2.2) and evaluate the results to measure how well SANTI-morf performs. I will fully report multiple relevant evaluation measures. This exercise will determine whether or not SANTI-morf can outperform the state-of-the-art system evaluated in Chapter 5 and make it possible for firm claims to be made regarding what advancements have been accomplished by the novel MA.

Chapter 8 will conclude this thesis. First, I will review each part of the thesis project, summarising the outcomes and achievements, and in particular the resources developed. Subsequently, after a discussion of the limitations of the project and of possible directions for further work in the future (including plans for the development of SANTI-morf), I will conclude the thesis with some remarks to highlight the contribution to knowledge that this project has accomplished.

CHAPTER 2

BACKGROUND ISSUES TO THE MORPHOLOGICAL ANNOTATION OF INDONESIAN

This chapter introduces some background on Indonesian, the language for which automatic morphological annotation shall be developed in this thesis. The first section outlines the linguistic structure of Indonesian. The second section describes an Indonesian corpus built for use in this project.

2.1 An overview of the structure of Indonesian

2.1.1 Background

Indonesian (ISO 639-3 *Ind*), or *Bahasa Indonesia* (its autonym), is one of the standardised varieties of Malay or *Bahasa Melayu*. Indonesian is, by far, the most widely spoken Malay variety with more than 250 million speakers (Lewis 2009). It is the sole official language, as well as the national language, of the Republic of Indonesia. Indonesian is used as the medium of instruction in schools and universities in Indonesia. It is also used to write literature, for day-to-day communication, and is moreover widely used in both formal and casual situations in either spoken or written mode among Indonesians.

2.1.2 Phonetics and phonology

As Table 2.1 and Figure 2.1 illustrate, there exist 22 consonants (including 4 that are non-native, only found in loanwords), 6 simple vowels, and 3 vowel diphthongs in present-day Indonesian (Soderberg & Olson, 2008: 210-211).

	Bilabial	Labio-dental	Dental	Alveolar	Post-alveolar	Palatal	Velar	Glottal
Plosive & affricate	p b		t̪	d	tʃ dʒ		k g	(ʔ)
Nasal	m			n		ɲ	ŋ	
Flap/trill				r				
Fricative		(f)		s (z)	(ʃ)			h
Approximant	w					j		
Lateral approximant				l				

Table 2.1. Indonesian Consonants, reproduced from Sodeberg and Olson (2008:210)

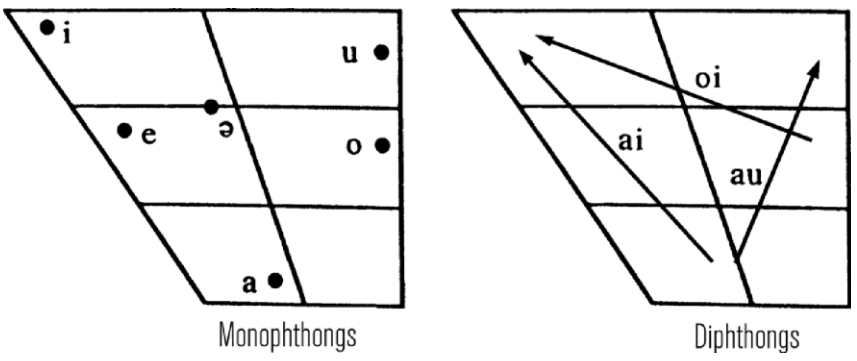
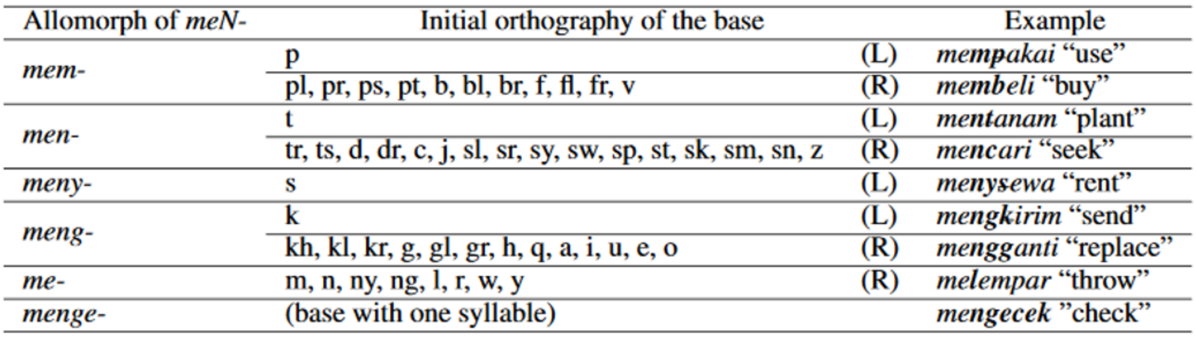


Figure 2.1. Indonesian Monophthongs and Diphthongs, reproduced from Sodeberg and Olson (2008: 211)

The great majority of Indonesian roots are disyllabic, each syllable being CVC, VC, CV, or V (Prentice, 1987: 190) (C = consonant, V = vowel). Prentice notes that Indonesian roots with one or three or more syllables are most probably loanwords (e.g. *bom* [bom] 'bomb' from Dutch or *jendela* [dʒən.de.la] 'window' from Portuguese). Similarly, non-(C)V(C) syllable structures (e.g. *struk* [struk] 'cash receipt', from Dutch) are likely to indicate loanwords.

A frequently discussed topic in Indonesian phonology and morphology is the sound changes that apply upon certain affixations (Alwi et al 1998:110-113; Sneddon 2010:13-17), known as *morphophonemic alternations* (or, sometimes, *sandhi*, but I do not use this term). For instance, *menge-, meng-, meny-, men-, mem-,* and *me-* are allomorphs of the morpheme whose citation form is often written *meN-*. The uppercase N in *meN-* represents the varying nasals [ŋə], [ŋ], [ɲ], [n], [m], and [ø], whose alternation depends on the phonetic environment, that is the root-initial sound. Specifically, there is homorganic nasal assimilation with an initial consonant of the root, which may be lost (L) or retained (R) in the process. Moeljadi et al. (2015: 17) summarise the rules as shown in Table 2.2.



Allomorph of *meN-*	Initial orthography of the base		Example
mem-	p	(L)	***mempakai*** "use"
	pl, pr, ps, pt, b, bl, br, f, fl, fr, v	(R)	***membeli*** "buy"
men-	t	(L)	***mentanam*** "plant"
	tr, ts, d, dr, c, j, sl, sr, sy, sw, sp, st, sk, sm, sn, z	(R)	***mencari*** "seek"
meny-	s	(L)	***menysewa*** "rent"
meng-	k	(L)	***mengkirim*** "send"
	kh, kl, kr, g, gl, gr, h, q, a, i, u, e, o	(R)	***mengganti*** "replace"
me-	m, n, ny, ng, l, r, w, y	(R)	***melempar*** "throw"
menge-	(base with one syllable)		***mengecek*** "check"

Table 2.2. meN- prefixation rules (root-initial consonant: L=lost, R=retained), reproduced from Moeljadi et al. (2015:17).

So, for instance, root *beli* 'buy' begins with *b*, so the cooccurring allomorph is *mem-* and the root-initial consonant is retained, yielding *membeli* 'buy'. Conversely, for root *pakai* 'wear', the allomorph is the same but the initial *p* is lost, yielding *memakai* 'wear'.

2.1.3 Morphology and syntax

2.1.3.1 Clause structure

The basic transitive word order in Indonesian is SVO (Subject-Verb-Object), a common word order in the world's languages. Some languages employ copulas to link subjects and non-verbal predicates, such as English *be*. Whether this is true of Indonesian is a contentious matter. Prentice (1987: 204) argues that this category of verb is absent in Indonesian; clauses such as *saya guru* 'I am a teacher', literally *I + teacher,* or *dia gembira* 'he is happy', literally *he + happy*, are well-formed. Uzawa (2007:315-338) argues that, in closely-related Malaysian, the words *ialah* and *adalah* have copulative functions, but doubts that they are verbs. By contrast Prentice (1987:204) notes that verbs *ialah* and *adalah* are used in equational sentences (with subject and subject complement), without categorising them as copulas.

2.1.3.2 Inflection and derivation in Indonesian

Views differ on the existence of inflectional affixes in Indonesian. Some scholars, including Prentice (1987:193) and Chaer (2008:37-41), maintain that the voice-marking prefixes *meN-* and *di-* are inflectional, but other affixes are derivational. However, they further argue that *meN-* is ambiguous because it can also be derivational; thus there are actually two *meN-* prefixes. Mueller (2007:1221-1222) argues that definite/pronominal suffix *-nya* is also inflectional.

Other scholars doubt or reject the notion that any affix can be considered inflectional. Musgrave (2001:5), for instance, argues that Indonesian is exclusively derivational in its morphology.

Yet other scholars avoid discussing the inflection-derivation distinction, including Kridalaksana (1989) and Alwi et al. (1998), instead focussing on describing affixes' *meaning* (in terms of what functional grammatical and semantic categories they indicate). Alwi et al.'s (1998) reference grammar uses the term *penurunan* 'derivation' for all affixation, but whether this is intended to assert that Indonesian morphology is exclusively derivational is unclear.

Derivational affixes do not exhibit regularity and productivity as inflectional affixes typically do. A few of these irregularities are as follows. First, affixes do not always apply to every base in a relevant category. For instance, of intransitive verbs *bangkit* ' get up' and *bangun* ' wake up', only *bangkit* can take nominaliser circumfix *ke—an*: *ke-bangkit-an* but **ke-bangu-nan*. Second, some functional grammatical categories can be present whether or not an affix that marks that function is present. For instance, *me-* (allomorph of *meN-*) *optionally* marks active voice in example (2.1); without it, the voice remains active. Other voice affixes (e.g. passive *di-*) cannot be omitted thus.

(2.1)	*saya*	*(me-)makan*	*burger itu*
	1s	(ACV-)eat	burger DEM
	'I ate that burger'		

Third, a number of affixes are ambiguous, with various means of being disambiguated. Morphological contexts resolve some, e.g. the POS of the base; for instance, prefix *teR-* is usually to be understood as forming a superlative adjective when the base is an adjective (see (2.2), whereas elsewhere it is likely to be read as deriving a verb.

(2.2) *ter-cantik*
SUP-beautiful
'the most beautiful'

In other cases, the ambiguity might be resolved by higher-level analysis. For instance, in *pem-buka* 'opener', from verb root *buka* 'open', nominaliser prefix *pem-* can either be agentive or instrumental, as indeed can English *-er*. This ambiguity is typically resolved via syntactico-semantic, or even extra-linguistic, factors. However, some verb bases force one reading: e.g. in *pem-bohong* 'liar', from *bohong* 'lie', *pem-* is definitely agentive.

2.1.3.3 Productivity

Morphologically *productive* affixes create a wide range of full forms and have a regular function, e.g. *meN-*, which regularly marks active voice, and can occur with most verbs and nouns as well as certain bases with other POS. Unproductive affixes occur with closed sets of bases and cannot be extended to new words. These include certain infixes, e.g. *-em-* in *g-em-etar* 'tremble' from *getar* 'vibration'.

2.1.3.4 Polymorphemic words

Along with affixation, *compounding* and *reduplication* are the principal morphological operations forming complex words in Indonesian (Mueller, 2007:1208-1215), as in other western Austronesian languages (such as Tagalog, Javanese, or other Malay varieties). Complex words may also include clitics or particles, written with or without spaces.

Examples (2.3) and (2.4) illustrate complex word formation with more than one type of morphological operation: respectively reduplication plus affixation and compounding plus affixation. Central to these formations is the root, the core morpheme to which all operations (reduplication, affixation, and cliticisation) apply.

(2.3) *ku=men-cari -cari =mu=pun*
1s=ACV-search-search=2s=even
'even if I am looking for you over and over

(2.4) *men-(t)anda -tangan -i*
ACV-sign- hand -APPL
'provide a signature'

2.1.3.4.1 Roots

In traditional morphological analyses, the core of a word may be analysed as a root, base or stem. Here, I focus on the root because in morphological annotation, words are formally tokenised into individual morphemes; bases and stems, being possibly composed of multiple morphemes, are less important concepts.

POS	Examples
Noun	*nasi* 'rice', *jagung* 'corn', *London* 'London'
Pronoun	*aku* 'I' (personal)*, kenapa* 'why' (interrogative)*, sini* 'here' (demonstrative)
Numeral	*satu* 'one' (cardinal), *pertama* 'first'(ordinal), *semua* (indefinite)
Classifier	*ekor* 'animal classifier', *orang* 'human classifier'
Verb	*pergi,* 'go', *makan* 'eat', *lari* 'run'
Adjective	*cantik* 'beautiful'*, cepat* 'quick'*, lama* 'long'
Adverb	*selalu* 'always'*, jarang* 'seldom'*, hanya* 'only'
Preposition	*di* 'at', *ke* 'to', *dari* 'from'
Conjunction	*dan* 'and', *atau* 'or', *ketika* 'when'
Interjection	*hai* 'hi', *aduh* 'ouch', *astaga* 'oh my god'
Article	*si* 'the(derogatory)', *sang* 'the (honorific)'
Particle	*kah, lah, pun* (all emphasis)

Table 2.3. Twelve classes of Indonesian root morphemes

Table 2.3 illustrates some roots according to their POS. Morphological operations do not apply to some categories of root (interjections and articles). Cross-linguistically common categories (noun, verb, adjective) require no further explanation here. Others merit further comment.

First, the category of *classifier* is widely present across Asian languages including Japanese, Korean, Vietnamese, and Assamese. Classifiers group enumerated head nouns into semantic domains, such as person, animal, or plant (see further Aikhenvald 2000). The majority of Indonesian classifiers also occur as nouns. For instance, *orang* 'person', *ekor* 'tail' and *buah* 'fruit' are also human, animal, and generic classifiers respectively. Classifier versus noun ambiguity can be resolved from position within the noun phrase, since classifiers always immediately follow numerals in Indonesian. The numeral-classifier complex normally precedes (2.5) but sometimes follows (2.6) the head noun.

(2.5)	*tiga*	*ekor*	*ikan*
	Three	CLA	fish
	'three fish'		
(2.6)	*ikan*	*tiga*	*ekor*
	fish	three	CLA
	'three fish'		

Numeral *one* can be expressed in free form *satu* or in bound form *se=*. Sneddon (2010:60) argues that *se-* is a prefix. However, I would argue that *se=* is a clitic form of *satu* owing to the definition of a clitic as a phonologically dependent but syntactically independent unit, and following Alwi et al.'s (1998:280) use of the term *proclitic numeral* for certain loan numerals in clitic form.

Words like *some* and *all* are normally described as quantifiers in English. According to some scholars, the Indonesian equivalents (e.g. *semua* 'all'; *beberapa* 'some') make up one of the subcategories of numerals, i.e. *indefinite numerals* (Kridalaksana 2007:80, Alwi et al. 1998:279), since their quantity is indeterminate.

Some Indonesian adjectives can be used as manner adverbs without any affixation, e.g. *cepat* 'quick' is used as adverb in (2.7) but adjective in (2.8). Resolving this ambiguity requires syntactic information. *Cepat* as a manner adverb follows a verb (here, *ber-jalan* ' walk') to which it may be linked by instrumental-comitative preposition *dengan* ‘with’. However, *cepat* not preceded by a verb is read as an adjective.

(2.7)	*kereta ini*	*ber-jalan*	*(dengan)*	*cepat*
	train DEM	INTR-run	with	quick
	‘the train runs quickly’			

(2.8)	*kereta*	*ini*	*sangat*	*cepat*
	train	DEM	very	fast
	‘this train is very quick’			

Particle is a category whose formal definition varies from one language to another (see 2.1.3.4.5 and □3.5.1.1.4.2.2.2). Indonesian particles express discourse functions, e.g. focus particle *lah* in expressions such as *saya=lah* (‘it is I’).

Indonesian has two articles, *si* and *sang*, which have derogatory and honorific purposes, but are rarely used. A more frequently used definite marker in Indonesian is suffix *–nya*. However, none of these behaves similarly to typical articles like English *a* or *the*.

Indonesian pronouns, including cliticised pronouns, have the same form whether syntactically functioning as a subject, object, or possessive pronoun. First-person singular *aku*, for instance, occurs in *aku kirim* ‘I send’ (subject), *kirim aku* ‘send me’ (object), and *rumah aku* ‘my house’ (possessive). The inanimate pronoun, equivalent to *it* in English, is only present in clitic form, not free form. For instance, in the phrase *mengirim uang* ‘send money’, *uang* ‘money’ cannot be replaced by an independent pronoun, but only by clitic *=nya*, as in *meng-(k)irim=nya* ‘send it’ (see 2.1.3.4.5 for more on Indonesian clitics).

Bound roots cannot surface as monomorphemic words (see 1.1.3.1). Prentice (1983:183) terms these *precategorial*. To form a word, such a root must undergo affixation, compounding, or another morphological process. For example, the roots of *pengungsi* ‘refugee’ and *pelanggan* ‘subscriber’, which are *ungsi* ‘refuge’ and *langgan* ‘subscribe’, cannot occur alone as monomorphemic words. Some affix, if not *peN-* then another, is needed.

2.1.3.4.2 Affixation

The content of this section draws on the accounts of Alwi et al. (1998), Sneddon et al. (2010), Kridalaksana (1989;2007), and Chaer (2008).

2.1.3.4.2.1 Form

Four formal categories of affix exist in Indonesian, *prefix*, *suffix*, *circumfix*, and *infix*, of which circumfixes and infixes are less common cross-linguistically than prefixes and suffixes. A circumfix is an affix that surrounds its base, and thus is composed of two elements (opening and closing). An example of an Indonesian circumfix is nominaliser circumfix *ke—an,* as in for instance *ke-mampu-an* 'ability' from *mampu* 'be able to'.

All Indonesian circumfixes pair elements which also occur separately as prefix/suffix in other contexts. Thus, it could be argued that each circumfix is merely a combination of a prefix and a suffix. However, I do not accept this argument, on the basis that, in contrast to a combination of prefix and suffix, the two elements of each circumfix must occur together. Removing one of the elements may make the word invalid (e.g. **ke-mampu*) or change its function (e.g. *mampu-an* 'more able to (informal)'). That said, Alwi et al. (1998:32) observe that in some cases, whether to treat a word as formed by a circumfix or a formally identical prefix+suffix combination is an analytic decision that must be made with caution. They illustrate this by contrasting examples of elements *ber* and *an*: *ber-datang-an* 'come randomly' and *ber-halang-an* 'be under constraint'. The former involves the random action circumfix *beR—an* applied to *datang* 'come', while the latter combines prefix *beR-* and suffix *-an*. The verbal root *halang* 'constrain' is nominalised by *-an*, and subsequently, the nominal base *halangan* 'constraint' is prefixed with intransitive verbaliser *beR-* yielding *ber-halang-an*.

An infix is an affix that intervenes within a root. Indonesian infixation is no longer productive and occurs only with a closed set of roots. For instance, root *tunjuk* 'point at' can be nominalised by infix *-el-* into *t-el-unjuk* 'index finger'. A full list of Indonesian affixes is given in Table 2.4 and Table 2.5 (including functional details to be explained subsequently).

Type	Affix	P	Outcome	Other function	Word	Root
PFX	*beR-(1)*	H	Verb	intransitive	*ber-satu* 'unite'	*satu* 'one'
PFX	*beR-(2)*	L	Verb	reflexive	*ber-cermin* 'look at self on a mirror'	*cermin* 'mirror'
PFX	*di-*	H	Verb	passive	*di-ambil* 'be taken'	*ambil* 'take'
PFX	*ke-*	L	Numeral	ordinal-collective	*ke-dua* 'both/second'	*dua* 'two'
PFX	*meN-*	H	Verb	active	*meng-ambil* 'take'	*ambil* 'take'
PFX	*pe-*	L	Noun	patientive	*pe-suruh* 'person to be commanded'	*suruh* 'command'
PFX	*pel-*	U	Noun	patientive	*pel-ajar* 'student'	*ajar* 'teach'
PFX	*peN-(1)*	H	Noun	agentive	*peny-(s)uruh* 'commander'	*suruh* 'command'
PFX	*peN-(2)*	H	Noun	instrumental	*peny-(s)erap* 'absorber'	*serap* 'absorb'
PFX	*peR-(1)*	H	Verb	causative	*per-besar* 'enlarge'	*besar* 'big'
PFX	*peR-(2)*	L	Noun	profession	*pe-tani* 'farmer'	*tani* 'farm'
PFX	*peR-(3)*	U	Noun	nominaliser	*per-tapa* 'meditator'	*tapa* 'meditate'
PFX	*peR-(4)*	M	Verb	Verb marker	*per-buat* 'do'	*buat* 'make'
PFX	*se-(1)*	H	Adjective	equative	*se-cantik* 'as beautiful as'	*cantik* 'beautiful'
PFX	*se-(2)*	H	Adjective	collective	*se-kantor* 'the whole office'	*kantor* 'office'
PFX	*teR-(1)*	M	Verb	accidental passive	*ter-telan* 'be swallowed accidentally'	*telan* 'swallow'
PFX	*teR-(2)*	M	Verb	Abilitative passive	*ter-beli* 'can be bought/buyable'	*beli* 'buy'
PFX	*teR-(3)*	M	Verb	Stative passive	*ter-tulis* 'be written'	*tulis* 'write'
PFX	*teR-(4)*	H	Adjective	superlative	*ter-cantik* 'most beautiful'	*cantik* 'beautiful'
PFX	*teR-(5)*	L	Noun	patientive	*ter-sangka* 'suspect'	*sangka* 'suspect'

Table 2.4. Indonesian prefixes (P=Productivity, H=High, M=Medium, L=Low, U=unproductive)

Type	Affix	P	Outcome	Other function	Word	Root
SFX	*-an*	H	Noun	nominaliser	*makan-an* 'food'	*makan* 'eat'
SFX	*-i (1)*	H	Verb	verbaliser	*kepala-i* 'lead'	*kepala* 'head'
SFX	*-i (2)*	H	Verb	applicative	*kirim-i* 'send to'	*kirim* 'send'
SFX	*-i (3)*	H	Verb	causative	*panas-i* 'apply heat'	*panas* 'hot'
SFX	*-i (4)*	H	Verb	iterative	*ketok-i* 'knock iteratively'	*ketok* 'knock'
SFX	*-i (5)*	H	Verb	applicative & iterative	*pukul-i* 'punch sth iteratively using sth'	*pukul* 'punch'
SFX	*-kan (1)*	H	Verb	verbaliser	*gambar-kan* 'draw'	*gambar* 'picture'
SFX	*-kan (2)*	H	Verb	applicative	*kirim-kan* 'send for'	*kirim* 'send'
SFX	*-kan (3)*	H	Verb	causative	*periksa-kan* 'have sth examined'	*periksa* 'examine'
SFX	*-nya (1)*	H	Noun	definite	*buku-nya* 'the book'	*buku* 'book'
SFX	*-nya (2)*	H	Noun	deadjectival /deverbal	*sakit-nya* 'the pain'	*sakit* 'sick'
SFX	*-nya (3)*	L	Adverb	adverbialiser	*biasa-nya* 'usually'	*biasa* 'usual'
SFX	*-wan*	U	Noun	male	*warta-wan* 'male reporter'	*warta* 'news'
SFX	*-wati*	U	Noun	male	*warta-wati* 'female reporter'	*warta* 'news'
CFX	*beR—an (1)*	L	Verb	reciprocal	*ber-pukul-an* 'hit one another'	*pukul* 'hit'
CFX	*beR—an (2)*	L	Verb	reciprocal & iterative	*ber-pukul-an* 'hit one another iteratively'	*pukul* 'hit'
CFX	*beR—an (3)*	L	Verb	random action	*ber-jatuh-an* 'fall randomly'	*jatuh* 'fall'
CFX	*beR—kan*	L	Verb	possessive	*ber-senjata-kan* 'have weapon'	*senjata* 'weapon'
CFX	*ke—an (1)*	H	Noun	nominaliser	*ke-baik-an* 'kindness'	*baik* 'kind'
CFX	*ke—an (2)*	L	Verb	adversative	*ke-hujan-an* 'get caught in rain'	*hujan* 'rain'
CFX	*ke—an (3)*	L	Adjective	'-ish'	*ke-merah-an* 'reddish'	*merah* 'red'
CFX	*peN—an*	H	Noun	deverbal /deadjectival	*peny-(s)atu-an* 'unification'	*satu* 'one'
CFX	*peR—an*	M	Noun	nominaliser	*per-satu-an* 'unity'	*satu* 'one'
CFX	*peR—i*	L	Verb	verbaliser	*per-baik-i* 'fix'	*baik* 'good'
CFX	*peR—kan*	L	Verb	verbaliser	*per-tahan-kan* 'maintain'	*tahan* 'hold'
CFX	*se—an*	U	Adverb	duration	*se-hari-an* 'all day long'	*hari* 'day'
CFX	*se—nya*	H	Adverb	manner	*se-cepat-nya* 'as soon as possible'	*cepat* 'quick'
IFX	*-el-*	U	Noun	nominaliser	*g-el-embung* 'bubble'	*gembung* 'swollen'
IFX	*-em-*	U	Noun	plural	*j-em-ari* 'fingers'	*jari* 'finger'
IFX	*-em-*	U	Verb	verbaliser	*g-em-etar* 'shake'	*getar* 'vibrate'
IFX	*-er-*	U	Noun	nominaliser	*s-er-uling* 'flute'	suling 'flute'
IFX	*-er-*	U	Noun	plural	*g-er-igi* 'teeth'	*gigi* 'tooth'

Table 2.5. Indonesian suffixes, circumfixes and infixes (abbreviations as in Table 2.4)

It is possible for an affixed word to be a base for further morphological processes. For instance, *per-besar* 'enlarge', with prefix *per-*, can serve as base for passive *di-*, as in *di-per-besar* 'be enlarged'. The circumfixed word *ke-mampu-an* 'ability' can serve as base for intransitiviser *beR-*, as in *ber-ke-mampu-an* 'possess ability'.

2.1.3.4.2.2 Function

Affixes can also be analysed by functional grammatical categories. One such is *outcome POS category*, that is the POS of the word that the affixation creates. For instance, the outcome POS of prefix *meN-* is a verb (whether or not the base is a verb). In some cases, outcome POS category is ambiguous. For instance, *jatuh* 'fall' plus *ke—an* yields *kejatuhan*, which may be a noun 'fall' or a verb 'get hit by something that fell'. The list of affixes reorganised according to outcome POS is given in Table 2.6, Table 2.7, and Table 2.8. The examples given in Table 2.4 and Table 2.5 are not repeated.

Affix	Outcome	Other function
beR-(1)	Verb	intransitive
beR-(2)	Verb	reflexive
di-	Verb	passive
meN-	Verb	active
peR-(1)	Verb	causative
peR-(4)	Verb	verb marker
teR-(1)	Verb	accidental
teR-(2)	Verb	abilitative
teR-(3)	Verb	stative
-i (1)	Verb	verbaliser
-i (2)	Verb	applicative
-i (3)	Verb	causative
-i (4)	Verb	iterative
-i (5)	Verb	applicative
-kan (1)	Verb	verbaliser
-kan (2)	Verb	applicative
-kan (3)	Verb	causative
beR—an (1)	Verb	reciprocal
beR—an (2)	Verb	reciprocal & iterative
beR—an (3)	Verb	random action
beR—kan	Verb	possessive
ke—an (2)	Verb	adversative
peR—i	Verb	verbaliser
peR—kan	Verb	verbaliser
-em-	Verb	verbaliser

Table 2.6. Verb outcome affixes

Affix	Outcome	Other function
pe-	Noun	patientive
pel-	Noun	patientive
peN-(1)	Noun	agentive
peN-(2)	Noun	instrumental
peR-(2)	Noun	profession
peR-(3)	Noun	nominaliser
teR-(5)	Noun	patientive
-an	Noun	nominaliser
-nya (1)	Noun	definite
-nya (2)	Noun	deadjectival/deverbal
-wan	Noun	Male
-wati	Noun	Male
ke—an (1)	Noun	nominaliser
peN—an	Noun	deverbal/deadjectival
peR—an	Noun	nominaliser
-el-	Noun	nominaliser
-em-	Noun	plural
-er-	Noun	nominaliser
-er-	Noun	plural

Table 2.7. Noun outcome affixes

Affix	Outcome	Other function
se-(1)	Adjective	equative
se-(2)	Adjective	collective
teR-(4)	Adjective	superlative
ke—an (3)	Adjective	-ish
-nya (3)	Adverb	adverbialiser
se—an	Adverb	duration
se—nya	Adverb	manner
ke-	Numeral	ordinal-collective

Table 2.8. Adjective, adverb, and numeral outcome affixes

Affixes have other functions beyond the POS they derive. For example, *di-* and *meN-* mark active and passive voices respectively. Some scholars suggest that different terminology should be used to describe these voices in Indonesian, such as *agent and patient orientation* (Prentice 1987:193), or *actor voice* and *patient voice*, plus also *undergoer voice* (Mistica et al. 2009:46). To explain the reasons for these proposal is beyond the scope of this thesis. Likewise, this thesis does not aim to explore this debate or to argue for any particular proposal. For sake of practicality, I will use the familiar terms *active* and *passive*, because they are widely understood by not only linguists but also non-linguists.

2.1.3.4.3 Reduplication

Reduplicated words have two elements, one a root, and the other the realisation of some abstract reduplication morpheme. I will refer to the former as the *source*, and the latter as the *copy*. There are three formal categories of reduplication in Indonesian, full, partial, and imitative, discussed in Alwi et al. (1998:147-151,196-197,200,238-241), Sneddon (2010:18-26), and Chaer (2008:178-209).

In full reduplication, the *source* and *copy* are exactly alike, e.g. *hari-hari* 'days' from *hari* 'day'. Here, it is irrelevant which unit is deemed the source or copy. However, for imitative and partial reduplication, the two elements are distinct, and the source (root) and copy are identifiable. For instance, in the imitative reduplication in *sayur-mayur* 'a variety of vegetables' from *sayur* 'vegetable', *sayur* is the source and *mayur* the copy. In the partial reduplication in *je-jaring* 'webs' from *jaring* 'web' *je-* is the copy. The various grammatical functions of reduplication are given in

Full reduplication			
1	plural	*hari* 'day'	*hari-hari* 'days'
2	distributive	*lima* 'five'	*lima-lima* 'five each'
3	distributive	*cantik* 'beautiful'	*cantik-cantik* 'each is beautiful'
4	iterative	*pukul* 'hit'	*pukul-pukul* 'hit iteratively'
5	manner adverbialiser	*pelan* 'slow'	*pelan-pelan* 'slowly'
Imitative reduplication			
6	plural	*sayur* 'vegetable'	*sayur-mayur* 'a variety of vegetables'
7	plural	*warna* 'color'	*warna-warni* 'a lot of colours'
8	iterative	*balik* 'return'	*bolak-balik* 'go back and forth'
Partial reduplication			
9	plural	*daun* 'leaf'	*de-daun-an* 'leaves'
10	plural	*jamur* 'mushroom'	*je-jamur-an* 'mushrooms'
11	plural	*batu* 'stone'	*be-batu-an* 'stones'

Table 2.9. Reduplication and affixation may interact in complex ways; this is discussed further in □4.2.4.2

Full reduplication			
1	plural	*hari* 'day'	*hari-hari* 'days'
2	distributive	*lima* 'five'	*lima-lima* 'five each'
3	distributive	*cantik* 'beautiful'	*cantik-cantik* 'each is beautiful'
4	iterative	*pukul* 'hit'	*pukul-pukul* 'hit iteratively'
5	manner adverbialiser	*pelan* 'slow'	*pelan-pelan* 'slowly'
Imitative reduplication			
6	plural	*sayur* 'vegetable'	*sayur-mayur* 'a variety of vegetables'
7	plural	*warna* 'color'	*warna-warni* 'a lot of colours'
8	iterative	*balik* 'return'	*bolak-balik* 'go back and forth'
Partial reduplication			
9	plural	*daun* 'leaf'	*de-daun-an* 'leaves'
10	plural	*jamur* 'mushroom'	*je-jamur-an* 'mushrooms'
11	plural	*batu* 'stone'	*be-batu-an* 'stones'

Table 2.9. Full, imitative, and partial reduplication in Indonesian

	Function	Reduplication and affixation	Root	Root
1	iterative	*mem-(p)ukul-m-(p)ukul* 'hit iteratively'	*pukul* 'hit'	*pukul* 'hit'
2	plural	*pem-(p)ukul-an-pem-(p)ukul-an* 'acts of hitting'	*pukul* 'hit'	*pukul* 'hit'
3	iterative-reciprocal	*pukul-mem-(p)ukul* 'hit each other iteratively'	*pukul* 'hit'	*pukul* 'hit'
4	adverbialiser	*ber-hari-hari* 'for days'	*hari* 'day'	*hari* 'day'
5	adverbialiser	*se-cantik-cantik-nya* 'as beautiful as possible'	*cantik* 'beautiful'	*cantik* 'beautiful'
6	'no matter how'	*se-cantik-cantik-nya* 'no matter how beautiful'	*cantik* 'beautiful'	*cantik* 'beautiful'

Table 2.10. Reduplication with affixation

A compound or even a phrase may undergo reduplication. For instance, in *es krim-es krim* 'ice creams'; each element of the compound (*es* 'ice' and *cream* 'cream') is reduplicated. The next section discusses compounding in Indonesian.

2.1.3.4.4 Compounding

Most Indonesian compounds are left-headed, unlike typically right-headed English compounds. Some compounds, e.g. *mata-hari* 'sun' from *mata* 'eye' and *hari* 'day' (Mueller 2007:1208), do not admit a space. In other cases, a spaceless compound is distinct from the same elements *with* a space, typically by the compound having idiomatic interpretation, for example *orang tua* 'old person' (from *orang* 'person' and *tua* 'old') versus *orangtua* 'parents'. In yet other cases, the combination means the same regardless of the presence or absence of space, e.g. *tandatangan* or *tanda tangan* 'signature' (from *tanda* 'sign' and *tangan* 'hand') (Sneddon et al. 2010:26-28). Orthographic convention requires that a compound must not include a space when surrounded by a circumfix or prefix-suffix combination.

2.1.3.4.5 Cliticisation

A root morpheme may be analysed formally as a clitic if it is syntactically independent, but phonologically dependent, and thus, attached to another word (which I shall refer to as the clitic's *host*). The free and clitic forms are distinct. For example, the clitic form of the first person pronoun *aku* is *=ku* or *ku=*. A handful of morphemes only occur in clitic form, for instance, interrogative particle *=kah*.

Like affixes, clitics can be categorised based on their position relative to the host. In *ku=ambil* 'I take'. the clitic *ku=* is attached to the start of *ambil* 'take'; it is thus a *proclitic*. In *mem-(p)ukul=ku* 'hit me', by contrast, *=ku* is attached to the end of the host and is thus an *enclitic*.

Very few Indonesian morphemes can be cliticised: three personal pronouns, namely *aku* 'I', *kamu* 'you', and *dia* 's/he'; a number of discourse particles including interrogative *=kah*; and one numeral, *se=*, the clitic form of *satu* 'one'. *Aku* 'I' can be proclitic or enclitic; the other pronouns are only enclitics, and *se=* only proclitic.

There is no independent inanimate third-person pronoun in Indonesian (like *it* in English; see also 2.1.3.4.1). Demonstratives *ini* 'this' and *itu* 'that' are used instead, never *dia* – but *dia*'s clitic form, *=nya*, *is* used for inanimates. However, *nya* can also be read as a suffix (definite marker, adverbialiser, or deadjectival/deverbal noun marker; see Table 2.7). This adds further ambiguities. For example, *bukunya* can be glossed as *buku-nya* 'the book' or *buku=nya* 'his/her book', depending on the context.

2.2 A testbed corpus of Indonesian

2.2.1 Purpose

A corpus for evaluation purposes, or *testbed*, is needed in order to assess performance of morphological annotation software (a task to be carried out in Chapter 5 and section 7.5). This section details its purpose, design, and creation.

My testbed is not intended as a reference corpus of general Indonesian. Many issues that arise for general reference corpora are thus irrelevant here. A morphological testbed should ideally be representative in terms of morphological complexity and vocabulary coverage. However, this is hard to guarantee; no standard exists by which the morphological complexity of an Indonesian text could be measured. How, then, can suitable testbed data be selected and obtained?

The first alternative is to use an existing gold-standard morphologically annotated corpus (see Hierschman & Mani 2003:415; Wissler et al. 2014) as a basis for comparison. If such a corpus is available, it is important to ensure that it uses the same morphological annotation scheme as the program to be evaluated. However, to my knowledge, no such gold-standard corpus exists for Indonesian. Thus, this alternative is ruled out.

The second alternative, which I adopt, is to collect new corpus data so as to maximise vocabulary coverage generally, and coverage of morphologically complex words specifically.

2.2.2 Design

I hypothesise that the variety of morphologically complex words will be higher in corpus data from varied domains than from varied media; thus, the design of the corpus should emphasise variation of domain rather than of medium of publication. A single-domain corpus is likely to repeat domain-specific vocabulary, even if the source medium varies. On the other hand, a similar amount of text drawn from multiple domains is likely to contain more distinct word types, and thus provide better coverage of vocabulary and morphological phenomena.

In support of this hypothesis, I manually inspected and compared a 3,000-word text from a collection of poems and a 3,000-word text from a collection of science articles. I discovered that clitics *ku=* 'I'*, =ku* 'my/me'*,* and *=mu* 'your/you' occur in the former, but not in the latter. A corpus of science articles, then, would be unlikely to include words with these clitics – whose annotation ought to be part of any assessment.

I reviewed the variety of domains present in already existing Indonesian corpora. There are two open-access corpora of written Indonesian: 1) a 500K-word Indonesian national newspaper corpus, used in the PAN localisation project (Mirna & Riza 2009)[13]; and 2) a 5M-word corpus of Indonesian academic articles compiled by the Agency of Language Development and Cultivation[14].

These corpora, however, do not represent a wide range of domains. The PAN localisation corpus includes four (sports, economy, science, and international news); the academic writing corpus also includes four (health, life science, social science, and physics). I consider neither adequate in range.

Another open-access Indonesian corpus is available in the Leipzig corpora collection[15] (Goldhahn et al. 2012). This corpus consists of (1) encyclopaedia entries (from the Indonesian Wikipedia); (2) news articles (from online news portals); and (3) texts retrieved by random web-crawling, a category labelled "mixed" in the online catalogue[16]. While much larger than the PAN localisation corpus at 260M+ words , this corpus has only two clearly separate domains, and is thus not suitable overall, although subsections might be (see 2.2.3).

As a point of departure, I considered the British National Corpus or BNC1994 (Aston & Burnard 1998), a major English reference corpus. The written texts in BNC1994 are drawn from nine domains: *applied science, arts, belief & thought, commerce & finance, imaginative/creative, leisure, natural sciences, social sciences,* and *world affairs*. I opted to follow this model for the domains in my testbed.

[13] http://www.panl10n.net/english/OutputsIndonesia2.htm (last accessed 08/10/2019)
[14] https://korpusindonesia.kemdikbud.go.id/index.php?r=site/home (last accessed 26/05/2021)
[15] http://wortschatz.uni-leipzig.de/en/download/ (last accessed 26/05/2021)
[16] https://wortschatz.uni-leipzig.de/en/download/Indonesian (last accessed 26/05/2021)

The next point to consider is the overall target size for the testbed, and the amount of text per domain. The part-of-speech tagging of the BNC1994 was evaluated[17] over a 50K-word test sample, i.e. approximately 0.05% of the overall BNC1994. By this standard, a 10K-word testbed would be representative of an actual corpus of about 20M words. However, the BNC1994 annotations in question were word-level, not morpheme-level.

MorphInd (Larasati et al. 2011), the state-of-the-art morphological analysis system for Indonesian (see Chapter 5), was trained on a corpus of 100 sentences (Larasati, personal communication). This corpus is not publicly available. Assuming a sentence to be approximately 10-15 words, its size would be around 1 to 1.5K words.

Voutilanen et al. (1992) report a performance test of the English Constraint Grammar tagger (EngCG) using only a 2,167-word corpus. This is a highly relevant point of comparison. First, although EngCG is a parser, it also performs morphological analysis and ambiguity resolution. Second, EngCG uses a *rule-based* approach, which (as will be explained in section 6.9) is the approach I use. Third, Voutilanen et al. (1999:18) report that several evaluations using more text give essentially the same figures.

On the basis of these precedents, I set a target of 10K words for my testbed. Although this emerges from a review of practice for English, whose morphology is relatively simpler than Indonesian, two considerations support the decision. First, the tagging for English previously referred to is POS tagging, not morphological tagging. Arguably, POS in English is at least in the same league of difficulty as morphology in Indonesian. Second, for morpheme-level annotation, the number of units of evaluation is the number of morphemes, not words as in POS analysis. For every polymorphemic word composed of 3 morphemes, for instance, there will be 3 analyses to evaluate. Thus, the number of evaluations may be much greater than the wordcount. Finally, a practical consideration should be noted: since the data is to be manually processed, any much larger testbed could not be checked word-by-word within the time constraints of a PhD project.

[17] http://www.natcorp.ox.ac.uk/docs/bnc2error.htm (last accessed 26/05/2021)

2.2.3 Creation

The testbed was created as a subset of the Leipzig Indonesian corpus collection, the largest of the three mentioned above. This corpus is openly downloadable, with free-to-use-and-transform status (Creative Commons CC-BY-NC[18]). Along with the texts, the source URLs of the texts are provided. This made it possible to reclassify texts by domain to build a smaller, more varied corpus from the Leipzig data.

The reclassification was performed as follows. I identified *seed terms* for each of the nine BNC1994-derived domains by two methods. First, I translated the names of the domains to Indonesian, and used them as node terms to search the corpus; their collocates were manually inspected to identify further seed terms. Second, I used my native speaker introspection to identify additional seed terms for those domains. This resulted in nine groups of seed terms reflecting nine different domains.

I built a small PHP program to process the Leipzig corpus's URLs, classifying each according to the nine domains. Any text whose URL contains one of the seed words is deemed to belong to the domain that that seed word relates to. The script then downloads the original text from the internet. This generates a collection of texts from nine different domains, in total more than 18M words, balanced at 2M words per domain. This is only a fraction of the original Leipzig corpus, because many URLs could not be classified or were no longer accessible on the web. At this point, paragraphs were randomly selected (by searching for new-line characters) from each domain to form the 10K-word testbed (1.1K words per domain).

[18] https://creativecommons.org/licenses/by-nc/4.0/

CHAPTER 3
A REVIEW OF MOPRHOLOGICAL ANNOTATION SCHEMES

In Chapter 1, I stated several aims, among which the creation of a novel Morphological Annotation Scheme (MAS) is of central importance. Constructing a new MAS necessitates the identification of the current accepted best practices in this area. An account of these best practices will be the outcome of this chapter.

These best practices will be identified by means of a literature review of MASs. I consider, in order of presentation, MASs for Finnish, Turkish, Arabic, and Indonesian; and one universal MAS. The choice of these MASs are discussed in section 3.1. Each MAS is then reviewed separately (sections 3.2 to 3.6). The best practices identified from the reviews of these MASs are laid out in section 3.7.

3.1 Scope and organisation of this review

There exist many MASs, of which a high proportion are likely to share common features. Thus, reviewing all MASs in full would lead to redundant multiple discussions of these shared aspects. To avoid such unnecessary redundancy, I establish criteria to determine which MASs to include in this review. First, it is important that the MASs' coverage be broad. Some MASs target only specific morphemes, specific grammatical functions, or specific word formations. For instance, Neme's (2011) MAS focuses only on handling Arabic verbs. Neme & Laporte (2013) target only Arabic nouns, specifically the so-called broken (irregular) plurals. Such limited MASs are less relevant for present concerns, and are thus not included in this review. Second, the MASs to be reviewed must have been implemented in the annotation of a corpus. MASs that are still under development or not yet applied are not considered in this review. Third, I consider only MASs for languages whose morphological structures are as rich as or richer than Indonesian's (see 2.1), namely Finnish, Turkish, and Arabic. MASs for languages with little morphology, such as Chinese, are passed over as less relevant. Finally, I also include the cross-language Universal Dependencies (UD) MAS, because this annotation scheme has been adopted in many NLP systems for basic applications such as POS tagging, syntactic parsing, or higher level applications such as machine translation, text mining, and information retrieval.

For each language, I consider existing MASs in chronological order. A MAS used in an actual Morphological Analysis (MA) system and with relatively broad coverage of its language was first built for Finnish in the early 1980s by Koskenniemi (1983). Other Finnish MASs were developed from the early 1990s to the early 2010s. Turkish MASs were pioneered by Oflazer (1994). His work on Turkish MASs and NLP systems began in the early 1990s. Many later Turkish MASs and NLP systems would be driven by Oflazer's precedent. In addition to Oflazer's work, I also consider Coltekin's (2010) MAS for Turkish. As of the late 1990s to the late 2000s, Arabic MASs and NLP systems began to be developed. The two Arabic MASs considered here are the Buckwalter Arabic Morphological Analyser (BAMA) scheme (Buckwalter 1999) and the Standard Arabic Language Morphological Analysis (SALMA) scheme (Sawalha et al. 2013). For Indonesian, I consider two distinct MASs used in different MA systems: Pisceldo et al.'s (2008) Two-Level Morphological Analyser (PMA) and MorphInd (Larasati et al. 2011).

Around the early 2010s, a cross-linguistic annotation scheme called Universal Dependencies (UD) was created (McDonald et al. 2013). The morphological subsystem of UD is commonly used in modern NLP applications; it is the final MAS reviewed here, having been developed more recently than those mentioned above.

3.2 Some Finnish MASs

3.2.1 Kimmo Koskenniemi's MAS

The first widely-known Finnish MAS was associated with Koskenniemi's (1983) seminal Two-Level Morphology formalism, henceforth TLM. The MAS that Koskenniemi used is referred to here as *Kimmo Koskenniemi's* MAS or KKM. TLM was Koskenniemi's innovation in response to generative grammar, a highly influential theory based in the work of Chomsky (1957; 1963; 1965; 1968); see further Chapter 6. The current implementation of TLM which applies this MAS is Fintwol[19].

3.2.1.1 Some preliminary remarks on TLM

Chomskyan generative morphology describes word structure in terms of a sequence of transformations between an *abstract* (or *underlying*) *representation* and a *surface form*, formalised as transformational *rules.* Due to limitations of computer memory in the 1980s, it was not possible to implement this linguistic formalism as a working NLP system. Thus, Koskenniemi designed TLM as an alternative formalism that *could* be feasibly implemented (Koskenniemi 1983).

TLM bypasses all intermediate stages between underlying and surface forms, simplifying all processes into just two levels: the *lexical* and *surface* representations. In TLM, the surface representation is not seen as the result of transformations of the lexical representation. Instead, the two representations are seen as corresponding to one another. The technical details of TLM system will be discussed in section 6.6. I here focus on KKM.

3.2.1.2 Documentation of KKM in the literature

KKM was the first MAS successfully implemented in a fully working automatic MA system (there were earlier partial systems; see 6.5). The first published description of KKM is distributed throughout the text of Koskenniemi (1983), primarily an account of the software implementation. The documentation of a later version of Fintwol (Koskenniemi 1995) also covers both scheme and software. Covering MAS and software together makes the account concise but poses challenges for this review of KKM: its description presupposes some understanding of TLM (as covered above) and of Finnish morphology.

Finnish is a highly agglutinative Finno-Ugric language. A Finnish verb, for instance may be composed of seven morphemes or even more (Karlsson 1999:25). A polymorphemic word in

[19] http://www2.lingsoft.fi/cgi-bin/fintwol (last accessed 26/05/2021); http://www2.lingsoft.fi/doc/fintwol/intro/tags.html (last accessed 26/05/2021)

Finnish is usually built by concatenating *endings* onto roots. In Finnish morphology, *ending* is a cover term for enclitics, case suffixes, possessive suffixes, and other suffix-like morphemes (Karlsson 1999:4); however, I will stick to the more common linguistic terms *suffix* and *enclitic*. The interaction of a Finnish root with its suffixes (and clitics) involves much morphophonemic alternation, due to phenomena such as consonant gradation and vowel harmony (Karlsson 1999:28-38).

An important item of terminology in KKM is *feature*. Koskenniemi (1983:24) has two distinct definitions for *feature*. On the one hand, he uses it to refer to a morphological property or morphological boundary; features may trigger the application of morphophonological rules (Koskenniemi 1983:24). On the other hand, he also uses the term *feature* for functional features, such as the morphosyntactic feature of number (Koskenniemi 1983:48). This latter is more akin to the meaning of *feature* in usual linguistic terminology. For consistency, I will use *feature* only for the latter concept in this review.

3.2.1.3 KKM's handling of stems and suffixes

In KKM, Finnish suffixes are classified based on the categories of stem POS to which they can be affixed. The main categories of stem POS in Finnish are nominal and verbal. Each category of suffix is additionally subcategorised on the basis of how they combine with different sets of actual stems. These categories are called *continuation classes*, or simply *classes*; they are used for internal management of morphophonemics.

3.2.1.3.1 Nominal stem suffixes

Koskenniemi defines four classes within the category of nominal suffixes (S), namely nominative (S0), singular (S1), plural (S2) and other (S3/S4/S5). Class definitions are based on functional grammatical features as well as technical morphophonemic considerations (Koskenniemi 1983:48-51).

S0 contains the null suffix/zero morpheme, which characterises unmarked nominal roots whose reading is nominative singular. The other classes contain explicit case/number suffixes. Table 3.1 lists the possible values for case and number, and the tags that indicate them; Table 3.2 illustrates the use of these tags to annotate various nominal suffixes.

Nominal feature (S)	
Number • Singular (SG) • Plural (PL)	Case • Nominative (NOM) • Genitive (GEN) • Essive (ESS) • Partitive (PTV) • Inessive (INE) • Illative (ILL) • Adessive (ADE) • Ablative (ABL) • Commitative (CMT) • Instructive (INS) • Translative (TRA)

Table 3.1. Nominal suffix inflectional features and values in KKM

Continuation class	Nominative (S0)	Singular (S1)	Plural (S2)	Other (S3/4/5)
Example suffixes	(unmarked) NOM SG	*-n* GEN SG *-t* NOM PL *-ksi* TRA SG	*-ien* GEN PL *-ita* PTV PL *-issa* INE PL	*-ta* PTV SG *-ten* GEN PL *-tta* PTV SG

Table 3.2. Examples of KKM tags for nominal suffixes (adapted from Koskenniemi 1983:48)

It is odd that there should be a plural suffix, *-t*, in class S1 (labelled "singular"). If the categorisation were to be consistent, S1 ought only to include suffixes marking singular number. Koskenniemi (1983:48) explains, however, that this inconsistency is for technical reasons: the S2 class holds plural suffixes that begin with *-i*, and since nominative plural *-t* does not begin with *-i*, it is in class S1. However, this indicates that the S0 to S5 subclasses are not analytically meaningful, despite the titles that Koskenniemi gives them. Another indication of this is that grammatical features exhibit no exclusivity to particular continuation classes, e.g. genitive case is present in all of S1 to S5. Since this subclassification according to morphophonemic form and combining behaviour is not a part of the analytic annotation, but rather a part of the system of suffix recognition used in the implementation, I will pass in silence over continuation class for the remainder of my review of KKM.

3.2.1.3.2 Adjectival stem suffixes

Two features are defined in KKM for adjectival suffixes, namely degree (comparative (CMP), positive (POS), superlative (SUP)) and manner (MAN; present/absent) (Koskenniemi 1983:57). Although not explicitly defined, MAN seems to flag the function of suffixes which derive manner adverbs from adjectives. If so, this is an inconsistency in KKM, because MAN is unlike the other KKM tags for derivation (see Table 3.3).

(3.1)	*-mpi*	CMP
(3.2)	*-in*	SUP
(3.3)	*-sti*	POS MAN
(3.4)	*-mmin*	CMP MAN
(3.5)	*-immin*	SUP MAN

3.2.1.3.3 Verbal stem suffixes

The features of Finnish verbal suffixes are more complex than those of adjectival and nominal suffixes. They include: participle, infinitive, tense, mood, voice, person and number, case, negation, and derivation. The values of these features and corresponding tags in KKM are shown in Table 3.3 below.

Verbal feature (V)		
Person and Number • First person singular (Sg1) • Third-person plural (Pl3) • Fourth person (Pe4)	Case • Inessive (INE) • Translative (TRA) • Instructive (INS)	Voice • Active (ACT) • Passive (PSS)
Participle • Present Participle (PCP1) • Past Participle (PCP2)	Infinitive • 1st infinitive (INF1) • 2nd infinitive (INF2)	Mood • Conditional (COND) • Imperative (IMPV) • Potential (POTN)
Tense • Present (PRES) • Past (PAST)	Derivation • DVMI = 'mi' ending • DVMA = 'ma' ending'	Participles • Present (PCP1) • Past (PCP2)

Table 3.3. Inflectional and derivational features for verbal suffixes in KKM, with example values (adapted from Koskenniemi 1995)

Verbal suffixes can indicate up to four features at the same time. For instance, in (3.6) and (3.7), *-koon* and *-kaame* both mark imperative mood and active voice but indicate different person/number agreement categories.

(3.6)	*-koon*	IMPV ACT SG3
(3.7)	*-kaamme*	IMPV ACT PL1

Much of KKM is dedicated to representing inflectional suffixes. However, some derivational suffixes may be found within the class of verbal suffixes. The functional analyses for these derivational suffixes are distinctive from those for inflectional suffixes, as a comparison of (3.8) to (3.11) and (3.12) to (3.13) may serve to illustrate.

(3.8)	*-da*	PRES PSS
(3.9)	*-taisi*	COND PSS
(3.10)	*-da*	INF1 NOM
(3.11)	*-tu*	PCP2 PSS
(3.12)	*-taessa*	DVMA PSS INE
(3.13)	*-ma*	DVMA ACT

Each tag for a derivational suffix begins with DV, short for derivation applied to a verb. The remainder of the tag is based on the form of the derivational suffix. For instance, DVMA is the tag for suffix *-ma* in (3.13). Whereas tags for inflectional suffixes are functional grammatical analyses based on abbreviations commonly used in linguistic glossing, no functional characterisation of the derivation processes is included in the MAS. A tag which uses the morpheme's actual form to represent a functional category will be referred to henceforth as *self-labelled functional category*.

3.2.1.3.4 Clitics

Clitics, often referred to as *clitic particles* in the literature on Finnish, are annotated with self-labelled functional categories. For example, the tag for the interrogative mood clitic is *kan*. This is identical to its form *=kan.* Likewise, the emphatic clitic *=han* is annotated with tag *han* (Koskenniemi 1983:60). Koskenniemi does not discuss the clitics' functions. However, Karlsson (1999:20) reports that clitics *=han* and *=pa* are used for emphasis. Other clitics can be used to mark interrogative mood or for stating options (equivalent to English *either*).

3.2.1.4 Output format

The representation of KKM generated as output by the earliest TLM program (Koskenniemi, 1983) is laid out in two lines per word. The first line gives the word's morphemes, encoded in TLM formalism rather than citation or orthographic form. Fintwol imports this representation from its TLM-format lexicon resources. For instance, passive suffix *-tu* is represented as *ZTU. Example (3.14) shows this in the context of a full word. This formalism shows explicit morpheme boundaries (with +) and other details which support morphophonemic and morphotactic rule processing.

(3.14)	*Hakatuimassa*	word
	hakKa$t*$ZTU$+imPA$+ssA	first annotation line
	Hit V PCP2 PSS SUP INE SG	second annotation line

The second line presents a series of analytic labels: first a literal translation of the word's root into English, then a tag for the root's POS category, and finally tags for the values identified from the analysis of the non-root morphemes. In contrast to the first line, no morpheme boundaries are represented or implied. Thus, the correspondence between morphemes and analytic features is not preserved. The tags are all associated equally with the word as a whole.

3.2.1.5 KKM as a word-level analysis

The foregoing review makes clear the strong tendency of KKM to address word-level analysis instead of morpheme-level analyses. Koskenniemi's decision to include analytic categories associated with null morphemes attests to this. The categories of nominative singular, or positive degree, are represented by tags assigned to words, even though they correspond to no actual morpheme (nor even a zero element added to output line1).

This tendency is also evident from the lack of morpheme boundaries in the second line of the output format (the analyses). For instance, example (3.14) illustrates an analysis of six tags, with no morpheme boundaries given, for a word composed of a root and three suffixes. While the root translation and the first tag are associated with the root by definition, the remaining tags could relate to any of the suffixes. This approach is completely acceptable for word-level analysis, but does not fit well with morpheme-level analysis, as it is not possible to determine which analytic categories correspond to which morphemes.

3.2.2 Later derivatives of KKM

The Fintwol program is in continuous development; see 3.2.1.2. Since the version of Fintwol described by Koskenniemi (1983), some minor adjustments have been made. One is that the morpheme form analysis is no longer shown in the output. Thus, Fintwol output now consists of a single line, as in example (3.15). The literal English translation of the root has also been removed from the output. One point which has not changed, however, is that more recent versions of Fintwol still produce word-level (instead of morpheme-level) analyses.

(3.15) *koirillannekaan* (no translation provided)
"koira" N ADE PL 2PL kAA

Creutz et al. (2004) report on a corpus of Finnish (and English) which are annotated using KKM, but with certain adjustments. For instance, Creutz et al.'s MAS has only nine POS categories. Creutz et al. also introduce a special POS, A/N, for words ambiguous between adjective and noun. Another novel category introduced by Creutz et al. is PFX for prefix. Finally, certain derivational suffixes are defined as marking the agentive nominalisation, such as *-ma* (DV-MA) and *-ja* (DV-JA). This functional elaboration is absent from KKM, as explained above; yet the categories are still self-labelled.

Creutz et al. 's (2004) analysis is implemented using the *Hutmegs* software (*Helsinki University Technology Morphological Evaluation Gold-standard Packages*) instead of Fintwol (1995). However, it is reported elsewhere that Hutmegs' analysis is based on Fintwol's resources (Creutz & Linden 2004: 20). The Creutz & Linden MAS is, in consequence, another derivative of KKM.

Two substantial changes in this derivative of KKM mean that Hutmegs' analyses are morpheme-level instead of word-level. First, all tags that correspond to word-level analyses by virtue of analysing unmarked or "zero" suffixes have been removed from the MAS (e.g. nominative singular for a noun; see discussion in 3.2.1.3.1). Second, in the single-line Hutmegs output, each tag is linked to the orthographic form of the specific morpheme it analyses, as shown in (3.16).

(3.16) arvo:arvo|N a:PTV mme:1PL, arvo:arvo|N amme:amme|N

The word form *arvoamme* is ambiguous, with two alternative readings, which are analysed and presented comma-separated in the output. Whether this form is the simple inflected word 'of our value' or the compound word 'valuable bathtub' depends on whether *amme* is read as two suffixes or a single element (a compounded root). Regardless, the format is consistent: each analysis consists of the morpheme's orthographic form, then a colon, then its canonical form, then a pipe, and then the morpheme's tag. In the interpretation of *arvoamme* as a compound, both elements are tagged as roots.

3.2.3 The MAS used in the Finnish Treebank

One of the sub-systems of the annotation of the Finnish Treebank (Voutilainen et al. 2012) is morphological analysis. This was applied primarily using Omorfi (Pirinnen 2015), a Finnish morphological analyser, and secondarily using a number of additional post-processors (Voutilainen, personal communication). Henceforth, this MAS will be referred to as the Finnish Treebank MAS (FTM).

One major difference from the MASs discussed previously is that FTM introduces a new category of tags called *other* for various non-word elements based on their orthographic form. In corpus annotation generally, the POS of such elements is commonly termed *residual*. In FTM, the residual category is split into the subcategories of abbreviation, punctuation, acronym, sentence ending, and truncated compound.

3.2.3.1 Derivation tags

Another significant difference is the inconsistent organisation of the derivation tags in FTM. FTM's derivation tags are encoded identically to KKM, but their placement in the analysis output differs. In most cases, derivation tags are placed at the end of the morphological analysis, as illustrated by the self-labelled tag for suffix *-aise* in example (3.17). In example (3.18), the derivation tag is not shown despite the presence of the derivational suffix *-elle*. In example (3.19), the derivation tag for suffix *-llinen* appears in first position, indicating that in FTM, the order of tags (or at least of DV-tags) is meaningless.

(3.17) *sutaista* (Voutilainen et al. 2012: 56)
V Act Imprt Sg2 DV-AISE

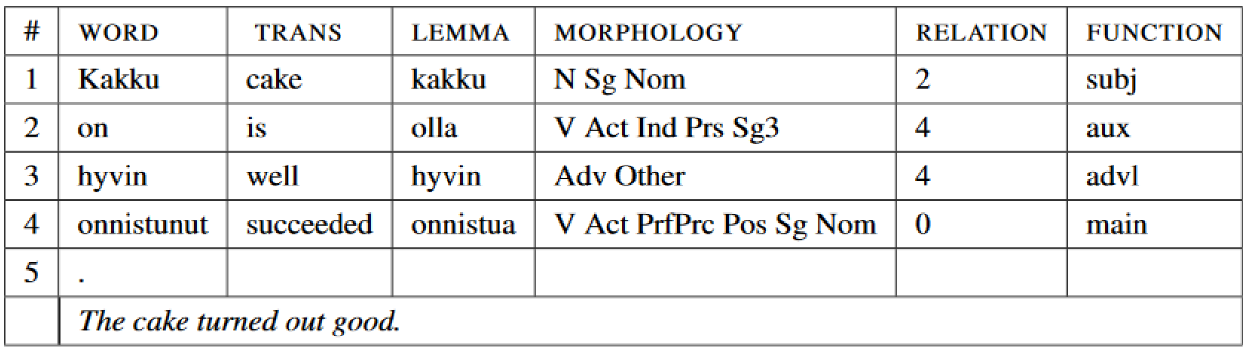
(3.18) *ponnistella* (Voutilainen et al. 2012: 68)
V Act Ind Prs Pl3

(3.19) *kaupallinen* (Voutilainen et al. 2012: 74)
DV-LLINEN N Nom Sg

Voutilainen (personal communication) explains that derivation tags are added *after* the other tags, which explains their variable placement. He moreover confirms that these analyses do not account for all derivational phenomena in Finnish.

3.2.3.2 Output format

FTM-annotated data is formatted as a five-column table, where the fourth column contains the morphological analyses (see Table 3.4). This and KKM's format share many similarities. The POS category of the word is followed by suffix tags, without no link from tags to corresponding morphemes; thus, FTM is another word-level MAS, not a morpheme-level MAS.



#	WORD	TRANS	LEMMA	MORPHOLOGY	RELATION	FUNCTION
1	Kakku	cake	kakku	N Sg Nom	2	subj
2	on	is	olla	V Act Ind Prs Sg3	4	aux
3	hyvin	well	hyvin	Adv Other	4	advl
4	onnistunut	succeeded	onnistua	V Act PrfPrc Pos Sg Nom	0	main
5	.					
	The cake turned out good.					

Table 3.4. FTM sample (reproduced from Voutilainen et al. 2012:53)

To summarise this review of Finnish MASs, I would observe that only the MAS used by Hutmegs (Creutz & Linden 2014) expresses a morpheme-level analysis. KKM and FTM, and the programs which implement them, provide word-level analyses, an approach which I would argue is more appropriate for morphosyntactic tagging than for morphological tagging.

3.3 Some Turkish MASs

Turkish is a highly agglutinative language of the Turkic family. It has much in common typologically with Finnish (and many other languages across central and northern Asia), including its agglutinativity. Oflazer (2018:23) notes that a Turkish word can theoretically be composed of up to 12 morphemes. As in Finnish, Turkish words are formed by productive suffixation with morphophonemic adjustments (geminations, alternations, and elisions of both consonants and vowels depending on the phonological environment).

The following notes on Turkish morphology draw on Goksel and Kerslake (2005). Turkish nominals can be marked by number, possession and case suffixes. Nominative singular stems are morphologically unmarked, as in Finnish. However, there are fewer nominal cases in Turkish than in Finnish. Locatives are generally expressed by postpositions instead of cases. Turkish verbal stems can be marked by voice and tense suffixes, as well as suffixes for subject agreement (person and number).

3.3.1 Oflazer's MAS

Oflazer was one of the first scholars to successfully apply TLM to a language other than Finnish, in this case Turkish. Oflazer's success justifies Koskenniemi's claim that TLM can serve as a general approach to computational analysis (or annotation) of morphology.

The first version of Oflazer's MAS (henceforth, OM) (Oflazer 1994) reflects Koskenniemi's work in that the distinction between the overall TLM formalism and the MAS specifically is not clear-cut. In the most recent version (Oflazer 2018), however, the discussion of TLM is kept apart from the discussion of the MAS, and no TLM formalism is used for the description of the MAS.

3.3.1.1 Roots, inflections, and derivations

In OM, Turkish roots are classified into 13 POS categories (shown in Table 3.5). One of these is onomatopoeic words. Treating onomatopoeia as a POS category is strictly inaccurate, because onomatopoeia is not a POS, but a word formation process; but this seems to be for practical purposes (see Goksel and Kerslake 2005:56). The categories of noun, numeral, and pronoun have subcategories.

Tag	POS
+Noun	Noun
+Adverb	Adverb
+Pron	Pronoun
+Num	Number
+Det	Determiner
+Ques	Question clitic
+Dup	Onomatopoeic words
+Adj	Adjective/modifier
+Verb	Verb
+Postp	Postposition
+Conj	Conjunction
+Interj	Interjection
+Punc	Punctuation

Table 3.5.Turkish root POS (adapted from Oflazer 2018:46)

Noun		Pronoun		Numeral	
+Prop	Proper noun	+Demons	Demonstrative pronoun	+Card	Cardinal number
		+Ques	Interrogative pronoun	+Ord	Ordinal number
		+Reflex	Reflexive pronoun	+Dist	Distributive number
		+Pers	Personal pronoun		
		+Quant	Quantifying pronoun		

Table 3.6. Subcategories of noun, pronoun and numeral in OM's POS system for Turkish

The OM tagset for inflection is presented in Oflazer (2018:47-51) and summarised in Table 3.7. In OM, all three types of nominal suffixes (number, possession and case) are considered inflectional. There are eight cases in Turkish; the singular and nominative are marked by null morphemes but are tagged explicitly in OM (Oflazer 2018:29, 48-50). Like KKM, it is a word-level analysis.

Number and Person	Possession
+A1sg = 1st person singular	+P1sg = 1st person singular possessive
+A2sg = 2nd person singular	+P2sg = 2nd person singular possessive
+A3sg = 3rd person singular nouns	+P3sg = 3rd person singular possessive
+A1pl = 1st person plural	+P1pl = 1st person plural possessive
+A2pl = 2nd person plural	+P2pl = 2nd person plural possessive
+A3pl = 3rd person plural nouns	+P3pl = 3rd person plural possessive
	+Pnon = No possessive
Case	TAM
+Nom= Nominative	+Past = Past tense
+Acc= Accusative	+Narr = Evidential past tense
+Dat= Dative	+Fut = Future tense
+Abl= Ablative	+Prog1 = Present continuous tense—process
+Loc= Locative	+Prog2 = Present continuous tense—state
+Gen= Genitive	+Aor = Aorist mood
+Ins= Instrumental	+Desr = Desiderative mood
+Equ= Equative	+Cond = Conditional aspect
Polarity	+Neces = Necessitative aspect
+Pos = Positive polarity	+Opt = Optative aspect
+Neg = Negative polarity	+Imp = Imperative aspect

Table 3.7. OM's inflectional features

Not all features marked by verbal suffixes are treated as inflectional. Polarity, person-number agreement, and a number of Tense-Aspect-Mood (TAM) suffixes are considered inflectional, whereas the suffixes marking voice and modality are considered derivational (Oflazer 2018:49-50). By contrast, Goksel and Kerslake (2005:69) analyse voice as an inflectional feature. We will see a similar phenomenon in Indonesian, described later in section 3.5, where Indonesian scholars dissent over the classification of Indonesian voice marking as inflection or derivation.

The OM derivational tagset is summarised in Table 3.8 . In OM, Turkish valency changes are treated as derivational, and classified into four voice categories, namely passive, causative, reflexive, and reciprocal; all are expressed by overt morphemes. Active voice is expressed by a null morpheme and is not included in OM. This is inconsistent with OM's treatment of nominative case, for which a tag is assigned (Table 3.7), even though the nominative is expressed by a null morpheme, like active voice.

Voice +Pass = Passive +Caus = Causative +Reflex = Reflexive +Recip = Reciprocal	Modality +Able =Able to verb +Repeat= verb repeatedly +Hastily= verb hastily +EverSince= have been verbing ever since +Almost =Almost verbed but did not +Stay =Stayed/frozen while verbing +Start= Start verbing immediately
Adverbial derivation (converb) +AfterDoingSo = After having verbed +SinceDoingSo = Since having verbed +As = As ... verbs +When = When . . . is done verbing +ByDoingSo = By verbing +AsIf = As if verbing +WithoutHavingDoneSo = Without having verbed	Others +Ly = Manner +Since =From temporal noun +With/+Without =modifiers derived from noun (in general) +Ness = nominaliser from adjective +Become = Inchoative +Acquire = Reception of something +Dim = Diminutive +Agt = Agentive

Table 3.8. Derivational features in OM (Oflazer 2018:46-51)

Oflazer (2018:49-50) uses the term *semantic twist* to characterise the meaning of many derivational suffixes. The tags for modality and subordinating suffixes are based on English translations of their meanings (see Table 3.8). The remaining derivational suffix analyses (under the heading of *other*) vary; some use common glossing conventions (e.g. +Dim, +Agt); others use the English translation pattern (e.g. +Ly, +Since, +Ness).

3.3.1.2 Output format

The OM output format consists of two lines. The first contains the word's morphs separated by explicit boundaries. The second contains the surface form of the root and its POS tags followed by morpheme tags. The same symbol (+) serves to demarcate both morphemes and tags on the two lines, but there is no transparent link between each morpheme and the corresponding analyses. In example (3.20), there are three morphemes (*ok, -um, -a*) whose tags are respectively ok+Noun, +A3sg+P1sg and +Dat, but nothing in the format indicates which tags apply to which morpheme.

(3.20) *Okuma* 'my arrow' (Oflazer, 2018:33)
ok+um+a
ok+Noun+A3sg+P1sg+Dat

In OM, derivational suffixes are assigned tags consisting of ^DB plus the suffix's outcome POS plus functional analytic category/ies. Example (3.21) exemplifies a word whose analysis includes multiple derivations; the final derivation(^DB+Adverb+While) determines the POS of the word.

(3.21) *Öldürülürken* 'while he is being caused to die' (Oflazer, 2018:33)
öl+dür+ül+ür+ken
öl+Verb^DB+Verb+Caus^DB+Verb+Pass+Pos+Aor^DB+Adverb+While

The roots of lexicalised collocations are treated as single sequences in OM, which therefore only receive single analyses. They are connected by an underscore as shown in (3.22). The multiword expression *hiç olmazsa* 'at least' is considered a lexicalised collocation in OM.

(3.22) *Hic olmasza* "become mentally deranged" (literally "eat head") (Oflazer, 2018:39)
hic_olmaza+Adj

In this review of OM, my main finding is that it is a word-level, rather than morpheme-level, analysis scheme . This is evident in the non-association of tags to specific morphemes and the varying use of tags for categories represented by null morphemes (see (3.21)and (3.22): nominative case and active voice are both null-marked but differently treated). Morpheme boundaries are present in the OM output, but they do not allow the user to link a tag to the morpheme that marks the function in question.

3.3.2 Coltekin's MAS

Another Turkish MAS is that used in TRmorph (Coltekin 2010). Compared to OM, Coltekin's MAS (henceforth, CM) makes improvements in certain areas, but in essence, is no different from OM in terms of specifying word-level rather than morpheme-level analysis, as the following review will show.

3.3.2.1 Treatment of roots

CM uses two POS categories not present in OM, namely *existence* and *symbol alphabet*. In addition, CM has more detailed coverage of the negation category than OM. In OM, negation is present only as a functional label associated with a given suffix. In CM, negation is present in both suffix and POS analyses, with distinctive tags. The negation suffix is tagged <neg>. The POS <NOT> consists only of the negator *degil* . Coltekin argues that, as *degil* has a special function, it should be assigned a distinctive tag (Coltekin 2013:3). The word *degil* is not mentioned in Oflazer (1994).

CM augments six POS categories (noun, pronoun, numeral, determiner, conjunction, postposition) with subcategories. The hierarchy in CM can be up to three levels deep. For instance, abbreviation is a subcategory of proper noun, and proper noun is a subcategory of noun; so the tag for abbreviations is in full <N:prop:abbr>. This stands in contrast to OM, where abbreviation is an independent major POS category.

3.3.2.2 Treatment of suffixes

OM's and CM's treatments of inflectional suffixes share many similarities. One significant difference is that CM has tags for copulative functions of verbal suffixes that form predicative clause constructions (Coltekin 2013: 9-10), such as past <cpl:past>, evidentiality <cpl:evid>, and conditional <cpl:cond>.

Coltekin (2013: 14-15) follows Goksel and Kerslake (2005:85-86) in using the term *subordinating suffix* for the suffixes listed as *converbs*[20] in OM; verbs so derived are placed in three subcategories, i.e. verbal noun, participle, and converb, which in turn have subcategories labelled either according to their TAM or with self-labelled functional tags. So, for instance,

[20] Roughly equivalent to English adverbial participial clauses.

infinitive verbal noun suffix <vn:inf> is the analysis of *-mak*, past participle suffix <part:past> is the analysis of *-Dik*, and converbial suffix <conv:cesine> is the analysis of *-cesine* .

3.3.2.3 Output format

CM's output format is a line which contains, in order, the root form of the word, the root's POS, and a collection of tags representing suffix analyses. Each tag is enclosed by angle brackets. Examples (3.23) to (3.25), taken from Coltekin (2010: 13, 7, 14), illustrate this

(3.23) *ev-ler-i* 'houses (accusative)'
ev<N><pl><acc>

(3.24) *ev-de-ki* 'in the house'
ev<N><loc><ki><Adj>

(3.25) *doktor-lar* 'they are doctors'
doktor<N><0><V><cpl:pres><3p>

Example (3.23) illustrates an inflection, but (3.24) and (3.25) involve derivational suffixes. The tag for *-ki* is self-labelled functional category <ki> plus outcome category <Adj>. Unlike OM, CM does not explicitly encode derivational boundaries, but they are implicit in the sequential suffix-and-outcome-POS tag pairs. The tag <0> prior to the outcome POS <V> indicates derivation with a null morpheme (zero derivation).

3.4 Some Arabic MASs

3.4.1 Background to Arabic

Arabic is an Afro-Asiatic language, written in the Arabic alphabet, which is laid out from right to left. In this review, for sake of readability, I render all Arabic words in the Latin alphabet following Buckwalter's transliteration scheme (Habash et al. 2010:15-22).

Arabic stems are derived by the nonconcatenative combination of *roots* and *patterns*. Typically, in Arabic, a root is a set of consonants with empty slots for vowels, while a pattern is a set of vowels that can potentially fill those slots. Arabic is well known for this *root-pattern* morphology.

Ryding (2005:46) compares this nonconcatenative phenomenon to ablaut in English, albeit the equivalence is inexact. We might analyse the inflection of *sing* as involving a root *s_ng* to which four different vowel patterns *i*, *a, u,* and *o* may be inserted to create stems *sing, sang, sung,* and *song*. This system is, however, much more prominent in Arabic; for instance the single root *k_t_b* yields many stems including *kaatib* 'writer', *-ktubu* 'write (present tense)', *kitaab* 'book' and *kutuub* 'books'.

An Arabic inflectional morpheme can mark multiple features, as is characteristic of inflectional languages. An example is suffix *-at* in *katab-at* 'she wrote' and *takallam-at* 'she spoke' (Ryding 2014:101). This one suffix marks the word as a feminine singular third person past tense verb[21]. Thus, at the same time, it marks four features (gender, number, person, and tense).

3.4.2 Buckwalter's MAS

One of the most well-known automatic morphological analysers for Arabic is the Buckwalter Arabic Morphological Analyzer (Buckwalter, 1999; 2001), or BAMA. BAMA's MAS is utilised by many other Arabic NLP systems, such as MADA (Habash et al. 2009), AMIRA (Diab et al. 2007), and MADAMIRA (Pasha et al. 2014). The purposes of these systems vary from POS tagging, to syntactic parsing or word-sense disambiguation, to higher-level applications such as information retrieval and named entity recognition. Interestingly, BAMA's main purpose is lexical tagging (i.e. lemmatisation and POS tagging), not morphological analysis, as Buckwalter explains:

> My primary goal in building a morphological parser was lexical tagging or identification—for use in lexicography, especially lemmatization—rather than morphological analysis *per se*. (Buckwalter 1999)

[21] Tense is also indicated by the vowel patterns.

Despite BAMA's popularity, its MAS is not documented in the published literature. (The citations I give here to Buckwalter are mostly to the different versions of BAMA distributed by the Linguistic Data Consortium.) I am thus forced to rely on the MAS documentation on Tim Buckwalter's website, the tagset file supplied with the program, and discussion of BAMA in the literature on other Arabic NLP systems. Henceforth, BAMA's MAS is referred to as BM.

3.4.2.1 Treatment of stems

In the BAMA tagset file, 135 tags for Arabic stems and affixes are listed; these can be combined to form larger tags. The examples provided imply that this MAS follows the three-class traditional POS classification for the Arabic lexicon (noun, verb, particle), plus a residual class.

BM analyses the major category of stems; these analyses are elaborated with subcategories, such as person and number, types of pronouns (demonstrative, possessive, etc.), and voice and TAM on verbs. Categories in the residual class include abbreviations, interjections, and foreign, Latin-alphabet and dialect words (Habash 2010:81).

3.4.2.2 Treatment of affixes

BM analyses a large number of inflectional affixes that co-occur with the two major stem categories of noun and verb. All affixes in BM are analysed into subcategories, as well as being assigned tags for morphosyntactic features such as TAM, gender, and voice.

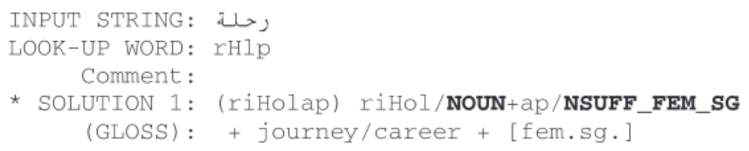
In BM, all suffixes are analysed by form, and then by function; these analyses combine to form larger tags. These larger tags include, for instance, NSUFF_FEM_SG, NSUFF_MASC_PL_NOM, and IVSUFF_SUBJ:D_MOOD:I. All three of these join a formal analysis (stem type plus SUFF) to functional analyses. This stands in contrast to the MASs reviewed in the preceding sections, in which affixes are not explicitly labelled according to their forms as prefix or suffix; very likely, this is because in Finnish and Turkish, the use of suffixes is prevalent. In Arabic, both prefixes and suffixes are used.

The fact that Arabic inflectional affixes can mark multiple features (see 3.4.1) is reflected in BM's combinations of feature tags. For instance, NSUFF_MASC_SG_ACC_INDEF encodes four different inflectional features: masculine, singular, accusative, and indefinite. All these values are expressed by a single morpheme, *-an*, for which NSUFF_MASC_SG_ACC_INDEF is the analysis.

3.4.2.3 Output format

BM's native output is complex; for readability, I use the more compact representation from the BAMA-tagged Penn Arabic Treebank (Maamouri et al. 2004), as shown in Figure 3.1; the morphological analysis is the same regardless of the representation used.

The full analysis extends over five lines: input string, lookup word, comment, solution and gloss. For words that may have multiple analyses, the alternate analyses differ merely in the last two lines (solution and gloss).



```
INPUT STRING: رحلة
LOOK-UP WORD: rHlp
     Comment:
* SOLUTION 1: (riHolap) riHol/NOUN+ap/NSUFF_FEM_SG
     (GLOSS):  + journey/career + [fem.sg.]
```

Figure 3.1. Buckwalter tags in the Penn Arabic Treebank (reproduced from Sawalha et al. 2013:83)

Let us focus on the third line, the solution, which contains the morphological analysis. Each morpheme is separated by a slash (/) from its tag; the plus (+) symbol demarcates morphemes in polymorphemic words. In the example in Figure 3.1. the word *riHolap* is analysed as being composed of the noun stem *riHol* and the nominal feminine singular suffix *-ap*. Tags composed of more than one value utilise a number of separators internally.

While BM covers almost all features of Arabic morphology an, one problem is inconsistency in how the tags represent its categories. The tag assigned to *riHolap*, NSUFF_FEM_SG, includes elements for three distinct features: affix type, gender and number. The first is a feature of form, while the other two are features of function. In this tag, the underscore (_) symbol is used as the demarcator. Here, all three features are independent; one is not a subcategory of another. However, in other cases, e.g. NOUN_PROP, the underscore is used to indicate a subcategory (proper noun is a subcategory of noun).

The NOUN_PROP example suggests that the tags are organised in a descending hierarchy, from left to right. But this directionality is not always preserved. For instance, in DEM_PRON_F, for feminine demonstrative pronoun, pronoun (PRON) is the highest level category. If the descending hierarchy were presented left-to-right, the tag should have begun with PRON instead of DEM.

3.4.2.4 Null allomorphs

It is common for morphemes to be manifest not only in the form of actual allomorphs, but also as a null morpheme or zero allomorph, as previously discussed for Finnish and Turkish; this is also true of Arabic. In BM, zero allomorphs are explicitly presented as *(null)* and are associated with analyses just as nonzero allomorphs are, as example (3.26)shows.

(3.26) *yaHotawiy* 'he/it contains'[22]
ya/IV3MS + Hotawiy/IV + **(null)**/IVSUFF_MOOD:I

In (3.26), the null morpheme is added to the tokenised word, and linked to a suffix tag that analyses it as marking indicative mood. The presence of this concrete link keeps BM's analysis at morpheme level. If the indicative mood value was not attached to (*null*), it could only be associated with the full word token, giving a word-level instead of morpheme-level analysis. Alternative mood morphemes in the same context are realised as actual non-zero forms, for instance, subjunctive suffix *-a*; for these, of course, the actual form is shown, as in (3.27).

(3.27) *yaHotawiya* 'he/it contain (subjunctive)'
ya/IV3MS + Hotawiy/IV +a/IVSUFF_MOOD:S

[22] https://catalog.ldc.upenn.edu/desc/addenda/LDC2004T27.xml (26/05/2021)

Although BAMA is the earliest morphological analyser for Arabic, its MAS can still be considered state-of-the-art, because its resources are used by many cutting-edge Arabic NLP systems. BAMA's strength lies in its wide coverage of Arabic morphology features and the link between each morpheme and its tag – despite a number of inconsistencies in the organisation of the MAS's tags, which are likely to concern linguists more than they do NLP researchers.

3.4.3 Sawalha et al.'s MAS

3.4.3.1 Overview

Another MAS for Arabic is that used by the SALMA morphosyntactic tagger (Sawalha et al. 2013). While it ultimately generates word-level morphosyntactic analysis, internally, SALMA performs analyses at morpheme level. Here, I do not consider SALMA's final morphosyntactic tagging output, but rather focus on its morpheme-level analysis.

The MAS used in SALMA will be referred to as Sawalha et al.'s MAS, or SAM. The practical and theoretical details of SAM are comprehensively documented by Sawalha et al. (2013). This stands in contrast to BM, whose documentation is sparse (see 3.4.2). The layout of SAM's analytic features is detailed in Sawalha et al. (2013:85-97), and the values for every feature are fully listed in the paper's appendix.

Analytic categories in SAM are represented as a feature-value matrix, in which numbered columns represent the features, and the row contains the corresponding values (see Sawalha et al. 2013:67). For instance, the major POS categories (noun, verb, particle, residual[23], and punctuation) are possible values of feature 1, *Main POS*. Feature 2 is *POS Noun*, with 34 values for noun subcategories. Feature 3 is *POS Verb*, with three verb subcategories.

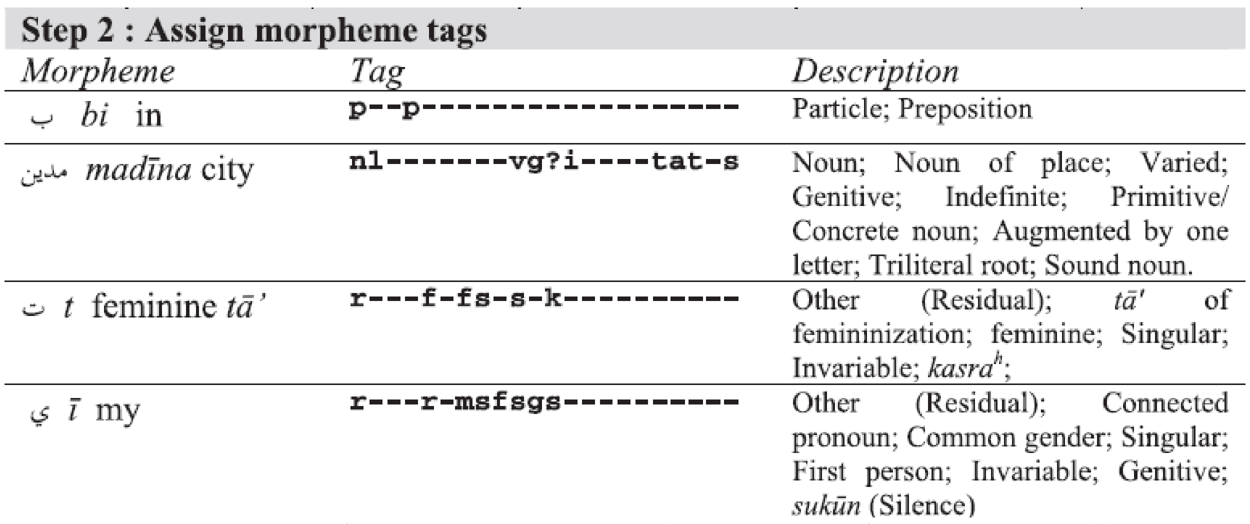
The features for properties other than POS begin at the 7th position. Feature 7, for instance, is *Gender*; feature 8 is *Number*. In this matrix representation, feature-values do not have to be organised as a hierarchy. For example, the subcategories of N (position 2) and of V

[23] The term 'residual' has broader meaning in SAM than other MASs discussed in this review, and includes some affixes and clitics as well as the usual borderline-word elements.

(position 3) are separate features rather than branches of a hierarchy of categories for the POS feature. There are 22 features, with more than 200 distinct values in total.

3.4.2.2 SAM's output format

In SAM, polymorphemic words are divided into morphemes; to each morpheme, a fixed-length 22-character tag is assigned. Crucial to this tag are not only the characters that encode particular feature-values, but also the *positions* of these characters, since the same characters are used in multiple positions. Each character position represents one column of the matrix, i.e. a single feature. Therefore, interpreting SAM tags involves two points, the analytic code plus its position in the feature matrix. This is illustrated by Figure 3.2.



Step 2 : Assign morpheme tags

Morpheme	*Tag*	*Description*
ب *bi* in	`p--p------------------`	Particle; Preposition
مدين *madīna* city	`nl-------vg?i----tat-s`	Noun; Noun of place; Varied; Genitive; Indefinite; Primitive/ Concrete noun; Augmented by one letter; Triliteral root; Sound noun.
ت *t* feminine *tā'*	`r---f-fs-s-k----------`	Other (Residual); *tā'* of femininization; feminine; Singular; Invariable; *kasra^h*;
ي *ī* my	`r---r-msfsgs----------`	Other (Residual); Connected pronoun; Common gender; Singular; First person; Invariable; Genitive; *sukūn* (Silence)

Figure 3.2. Morpheme tags from the word *bimadīnatī* 'in my city' as a feature matrix in SAM (reproduced from Sawalha et al. 2012:68)

Values are assigned only to positions whose features are relevant to the morpheme under analysis. Hyphens are assigned to positions whose features are irrelevant. The first example in figure 3.2 shows a tag for proclitic preposition *bi* 'in'. The tag begins with *p* (in first position) indicating that it is a particle. Another *p* (for preposition) occurs in fourth position, the column for particle subcategories. Hyphens appear in all the other positions, their features not being relevant for this morpheme. For instance, as a preposition, *bi* has no values for gender, number and person, so the corresponding positions (7th, 8th and 9th) contain hyphens. Other morphemes in Figure 3.2 do have values at these positions.

Three final points may be made. First, the documentation of SAM is meticulous and much more systematic than the other MASs considered so far; thus, it is more linguistically motivated. Second, its feature matrix organisation results in 22-character tags which are lengthy and challenging for human readers to interpret, though easily manipulable by computer programs. Finally, that there is a separate tag for each morpheme means that SAM's intermediate analysis is at morpheme level – even though it is not the end product, which is a morphosyntactic analysis.

3.5 Some Indonesian MASs

Earlier sections of this review began by overviewing the morphology of the language under discussion. However, Indonesian morphology has been covered in Chapter 2, and therefore need not be introduced now.

3.5.1 Pisceldo et al.'s MAS

3.5.1.1. Overview

Pisceldo et al.'s MAS (henceforth, PM) is a scheme used by the Two-Level Morphological analyser for Indonesian (Pisceldo et al. 2008). This system performs two distinct NLP tasks: synthesis (generating words from roots) and analysis; this review addresses only analysis. I will treat PM briefly, before dealing with subsequent work at greater length. This is because Pisceldo et al. acknowledge that their analysis is oversimplified (Pisceldo et al. 2008:145), which means there is little to discuss

The PM documentation extensively describes Indonesian morphology, the scheme's morphological tags, and its technical implementation and evaluation. However, only a limited number of output examples are presented. Thus, unless indicated by citation, examples in this section are not from Pisceldo et al. (2008), but rather, attested textual examples to which I have manually applied PM analysis.

3.5.1.2. POS categories

Pisceldo et al. (2008) classify the Indonesian lexicon into only four categories: noun (+Noun), verb (+Verb), adjective (+Adjective), and other (+Etc). The remaining major POS, such as adverbs, prepositions and conjunctions, are included within +Etc, even though they are actually major categories. Particles and cliticised pronouns are left unanalysed in PM.

While a three-division POS classification is commonly found in analysis schemes for Arabic (see 3.4.3.1), it is contrary to the usual tradition of Indonesian grammar, which typically places other major POS categories such as adverbs, prepositions or conjunctions on the same level of the hierarchy as verbs, nouns and adjectives. Pisceldo et al. do not explain why they organise the categories this way, but do acknowledge that it is an oversimplification to be addressed in the future (Pisceldo et al. 2008:146).

The POS tags for monomorphemic and polymorphemic words are slightly different. The POS tags for monomorphemic words (unaffixed roots used as words) are preceded by +Bare., This mechanism is illustrated in examples (3.28) and (3.29), showing a monomorphemic verb tagged as +BareVerb and a polymorphemic verb tagged as just +Verb.

(3.28) *pukul* 'hit' (monomorphemic word)
*pukul+*BareVerb

(3.29) *mem-(p)ukul* 'hit' (polymorphemic word)
*pukul+*Verb+AV
(reproduced from Pisceldo et al. 2008:150)

3.5.1.3. Affixation and reduplication

PM incorporates labels for two functional grammatical features marked by affixes: voice (active (+AV), passive (+PASS), undergoer (+UV), causative (+Caus_I and +Caus_kan), applicative (Appl_I and Appl_kan)); and nominalisation (agentive(+Actor) or instrumental (+Instrument)). PM also takes into account polysemous grammatical morphemes by integrating formal and functional labels. The tags +Caus_I and +Appl_I illustrate this; *-i* can mark either applicative or causative voice in Indonesian, and PM has a category for each function (see 2.1.3). PM also has a +Redup category, applied to all reduplications at word level.

The interaction of reduplication and affixation is left unanalysed in PM. For instance, the interaction of prefix *meN-* and full reduplication of a transitive verb results in reciprocal voice (see 2.1.3.4.3 and 4.2.4.2). In PM annotation, this interaction is analysed just as verb reduplication, without specifying the reciprocal function, as shown in (3.30).

(3.30) *pukul-mem(p)ukul* 'hit one another'.
pukul+Verb+Redup

3.5.1.4. Output format

PM output is a single line consisting of, in order, the root of the word; the POS of the word; and all the morphological tags. These elements are separated by the plus symbol. For example, *pukul+*Verb+AV is the analysis of *memukul* 'hit', composed of active prefix *mem-* and verbal root *pukul* 'hit' (Pisceldo et al. 2018:150). Example (3.31) has more complex morphological tags, due to the suffixation of causative *-kan*.

(3.31) *mem-(p)eriksa-kan* 'have (oneself) examined'
*periksa+*Verb+AV+Caus_kan

Although PM claims to be a system for morphological analysis (Pisceldo et al. 2008:146), I would argue that the analysis is at word level, for two reasons. First, in the output, the POS tag after the root is not that of the root, but that of the full word. Second, PM does not link morphological tags to the morphemes they analyse; for instance, in (3.31) +AV is not linked to *mem-* (which is, in fact, absent from the analysis). This link, absent in PM, is indispensable to morpheme-level analysis.

3.5.2 Larasati et al.'s MAS

3.5.2.1 Overview

Larasati et al.'s (2011) MAS is used in MorphInd, the state-of-the-art automatic morphological analysis system for Indonesian. Chapter 5 is dedicated to discussion of MorphInd and an evaluation of its performance. This section focuses on Larasati et al.'s MAS (henceforth LM). In addition to Larasati et al.'s paper, I draw on the more up-to-date documentation for MorphInd on Larasati's website (Larasati 2011).

In LM, each word is given two tags of different kinds. The first is a shallow POS tag for the word's root; Larasati et al. incorrectly use the term *lemma* for this tag. The second is the full-word tag (or fine-grained POS). Larasati et al. term this a morphological tag, whereas it is actually a morphosyntactic tag. Affixes are not separately tagged. Thus, LM fully implements word-level morphosyntactic analysis, but at morpheme level, only the root receives any analysis.

LM does not have any formal morphological tags for prefix, infix, suffix, or circumfix form or for reduplication. All MorphInd tags are functional. In the initial version of LM (used in MorphInd v.1.1), there was a tag for reduplication, but in the most recent version (MorphInd v.1.4), this has been replaced by a tag for plural, which references the *function* of the reduplication.

3.5.2.2 Lemma analysis

In the LM documentation (Larasati 2011), the term *lemma* is used for the unit on which shallow POS categories are annotated. However, this terminology is apparently incorrect (or at least highly idiosyncratic) because the unit to which this term is applied in LM is actually the *root,* in free or clitic form. One significant difference between PM and LM is that there are 18

POS categories in LM, plus 1 residual category (punctuation); Larasati et al. (2011) class all these as "lemma tags". All the major POS categories are at the same hierarchical level (see Table 3.9).

Noun <n>	Modal <m>
Personal pronoun <p>	Determiner <b>
Verb <v>	Adverb <d>
Numeral <c>	Particle <t>
Adjective <q>	Negation <g>
Coordinating conjunction <h>	Interjection <i>
Subordinating conjunction <s>	Copula <o>
Foreign word <f>	Question <q>
Preposition <r>	Unknown <x>
	Punctuation <z>

Table 3.9. LM's "lemma" tags for major POS category (adapted from Larasati 2011)

Although LM is more fine-grained than PM, the organisation of root tags does not fully reflect the usual organisation of POS categories in the Indonesian lexicon, just as with PM. All 16 categories (excluding foreign, unknown, and punctuation) are major categories, even though some are obviously subcategories, such as interrogative (in their terms, "question") pronouns and personal pronouns (subcategories of pronoun); subordinating and coordinating conjunctions (subcategories of conjunction); and modal and negative adverbs (subcategories of adverb) (for evidence in support of these judgements see Kridalaksana 2007:51-121).

LM's use of one-letter tags is of note. Mostly, the tag is the initial letter of the full category name, such as *n* for noun or *v* for verb. However, when two or more categories begin with the same letter, all but one category must use a different letter. So, for instance, out of pronoun (*p*), preposition (*r*) and particle (*t*), only pronoun is represented by its initial. Unfortunately, Larasati et al. do not explain the choice of *r* and *t* as labels for preposition and particle. This lack of systematicity may cause issues for users, by making it harder to memorise the tags.

3.5.2.3 Morphological analysis

In LM, word-level tags for nouns, pronouns, verbs, conjunctions, and adjectives consist of the single letter for the POS plus extra letters for further analytic categories, whereas other classes of word have tags that consist only of the single letter.

A number of features of LM that are claimed to be *morphological analyses* are actually word-level analyses. This is evident from these analyses' encoding, as well as their content. For instance, LM includes number and person as features in the morphological analysis of personal pronouns (see Table 3.10). While this is appropriate for Finnish, Arabic, or Turkish (because person and number may correspond to a suffix), in Indonesian, these are always features of certain *words*, namely pronouns; no suffixes express these features.

Similarly, PM treats number as a property of verbs. This is inaccurate because, in contrast to English, verbs do not exhibit subject-verb agreement in Indonesian (see Chapter 2). However, it is possible that Larasati et al. confuse the term *plural* with *iterative.* For example, *pukul-pukul* 'hit iteratively' (reduplicated from *pukul* 'hit') is analysed as an active plural verb, tagged VPA (P means plural) in parallel to e.g. *buku-buku* 'books' tagged as NPS. Since Larasati et al. have no category label for iterative (see Table 3.10), it seems likely that they mean 'plural' verbs in the sense of iterativity, not in the sense of subject-verb agreement.

First character	Second character	Third character
Noun <N>	Plural <P> Singular <S>	Feminine <F> Masculine <M> Non specified <D>
Personal pronoun <P>	Plural <P> Singular <S>	First person <1> Second person <2> Third person <3>
Verb <V>	Plural <P> Singular <S>	Active <A> Passive <P>
Numeral <C>	Cardinal <C> Ordinal <O> Collective <D>	
Adjective <A>	Plural <P> Singular <S>	Positive <P> Superlative <S>
Coordinating conjunction <H> Subordinating conjunction <S> Foreign word <F> Preposition <R> Modal <M> Determiner <B> Adverb <D> Particle <T> Negation <G> Interjection <I> Copula <O> Question <Q> Unknown <X> Punctuation <Z>		

Table 3.10. Morphological analyses in LM tags (reproduced from Larasati 2011)

Like the POS labels, each LM morphological analysis element is represented by a letter, and the full tags are decomposable strings of one letter per analysis. For instance, the tag VSA decomposes to Verb–Singular–Active. This encoding style is distinct from the other MASs I have reviewed, where values are separated by a demarcator. However, decomposable tags like this are commonly found in morphosyntactic tagsets, as in CLAWS (Garside 1987) or the Penn Treebank (Marcus et al. 1993). This style of tagset seems concise and easy to remember, but also has some drawbacks.

First, I earlier noted that it is common for single-letter labels to derive from the first letter of the appropriate grammatical term. But when multiple categories begin with the same letter, another letter has to be chosen, reducing memorability. Second, when single-letter labels are organised in decomposable strings, users have another factor to memorise, namely the position. For instance, P in first position encodes *personal pronoun* but P in second and third position encodes *plural* and *positive* respectively. For the MASs previously considered, users only need to memorise the labels (except for SAM, where users must also memorise positions; see 3.4.3).

The organisation of LM's decomposable tags is neither fully hierarchical nor entirely flat. Given hierarchical organisation, one would expect that the A in VSA represents a subcategory of S, and the S represents a subcategory of V. In fact, however, A is not subcategory of S; both are subcategories of V and signify different features at the same level.

A major problem with LM's so-called morphological analysis is that only four functional features of Indonesian morphemes are analysed. The first is voice, but it is limited to active and passive; the second is adjective degree, limited to superlative. Other constructions (applicative, causative, reciprocal, iterative; see 2.1.3.4.2) are present in Indonesian, but absent from LM.

As noted in section 3.5.2.1, reduplication was analysed as a feature of form in the initial version of LM, but was later replaced with a functional analysis, namely plural. I would argue that analysing Indonesian reduplication by form is more reasonable, because not all reduplications (even those with the same phonetic pattern) express plurality. Some mark similarity, variation, or reciprocality; some are metaphorical. Taking it for granted that all reduplication indicates plural number is prone to lead to incorrect analysis. Thus, analysing reduplication by form is a safer option. The LM approach also leaves the interaction of reduplication and affixation (explained in 2.2.3) unaddressed, even though this interaction can mark important functions such as reciprocal voice. With LM, users have less ability to retrieve such functions.

3.5.2.4 Output format

LM's output format is distinct from the previously discussed MASs, even PM. It presents all morphemes within polymorphemic words in citation form, with plus symbols as morpheme breaks. Thus, for instance, the word *kumengirimkannya* in (3.32) receives the analysis in (3.33).

(3.32) *ku=meng-(k)irim-kan=nya*
1s=ACV-send-APPL=3s
'I send him it'

(3.33) aku<p>_PS1+meN+kirim<v>+kan_VSA+dia<p>_PS3

The citation form of *meN-* is shown instead of the actual allomorph *meng-*; the root's canonical form *kirim* is given even though it has undergone initial consonant loss after *meN-*. The verb is surrounded by first person proclitic and third person enclitic pronouns; the LM analysis presents their full forms.

A fundamental concern with this segmentation is that it does not distinguish prefix-suffix combinations from circumfixes, a distinction crucial in Indonesian (see 2.1.3.4.2). The polymorphemic word *kejatuhan* 'fall (n)' is segmented in exactly the same way as *mengirimkan* 'send (something)' in example (3.33). However, *ke—an* is a circumfix whereas *meng-* and *-kan* are a prefix-suffix combination.

After each word is an underscore and the word's morphosyntactic tag, VSA 'verb singular active' in (3.33) and NSD 'noun singular not determined' in (3.34).

(3.34) *kejatuhan* 'fall (n)'
ke+jatuh<v>+an_NSD

In LM, there are two types of POS tags. The first is applied to root morphemes (Larasati et al. term this the *lemma* tag; see 3.5.2.2), and presented within angle brackets. Both *kirim* 'send' in (3.33) and *jatuh* 'fall (v)' in (3.34) are labelled <v> for the POS of the root (i.e. verb).

The second POS tag is a morphosyntactic tag (which they term a *morphological analysis* tag), given following an underscore symbol after the chain of morphemes in canonical form. The category of *not Determined* represented by D in the tag NSD is a value of the gender feature. Including gender as a feature is not really useful, because Indonesian lacks grammatical gender. Two suffixes borrowed from Sanskrit, *-wan* and *-wati* (see 2.1.3.4.2.2), were used historically to create gendered nouns, but these suffixes are no longer productive, and in some cases are falling out of use. For instance, *wartawan* 'male reporter' is used to refer to both female and male reporters in contemporary Indonesian. LM's treatment of this word and its feminine counterpart are shown in examples (3.35) and (3.36).

(3.35) *warta-wan* 'male reporter'
warta<n>+wan_NSM

(3.36) *warta-wati* 'female reporter'
warta<n>+wati_NSF

LM annotates clitics as monomorphemic words. Both the clitics in example (3.33), that is *ku=* 'I' and *=nya* 'him/her', are given a root tag (in angle brackets) as well as a morphosyntactic tag (following an underscore). But while clitics thus receive separate treatment, all affixes are left untagged. This is a fundamental concern about the linguistic adequacy of LM as a morpheme-level analysis. In (3.33), for instance, the tag component A (active voice) is only present as part of the morphosyntactic tag VSA. It is not linked to the prefix *meN-* which expresses active voice.

The LM output format for a given word, and the same word with reduplication, are identical; both are analysed as single tokens. Only the tags are distinct, the tag for the reduplicated word will include P (plural).. For example, the output for non-reduplicated *buku* 'book' is *buku<n>_NSD,* while the output for reduplicated *buku-buku* 'books' is *buku<n>_NPD.*

Overall, then, I have observed that LM confuses the terms *lemma* and *root* (by using the former to label the POS of the root) as well as *morphosyntactic* and *morphological* analysis. The final product of LM is a morphosyntactic analysis (that is, a word-level POS tag), even though the output of an LM analysis does present polymorphemic words divided into morphemes. Moreover, while roots (not lemmas) are annotated for POS, affixes, the major element of Indonesian morphology, are left unannotated in LM. The fact that affixes are not separately analysed, and that functional categories are not linked to the morphemes that express them, are two critical drawbacks to LM. In the novel MAS for Indonesian which I will propose in chapter 4, one of my aims is to address these drawbacks

3.6 The Universal Dependencies MAS

In section 3.1, I briefly introduced Universal Dependencies (UD) and explained why the UD MAS (henceforth, UDM) is encompassed in this review. I will now expand on this issue.

3.6.1. UDM Components

UD is a system for multiple layers of corpus annotation. Its format follows the earlier CoNLL-U format (Computational Natural Language Learning - U) (Buchoolz & Marsi 2006). In this format, analyses are presented in tabular form, so the word tokens flow vertically. Each column stores different type of annotation; these are referred to as UD *components*. Columns 1, 2 and 3 contain a token ID, the word form, and a lemma annotation; columns 4, 5 and 6 contain analyses largely relevant to morphosyntax (and to small extent also morphology); columns 7, 8, and 9 are reserved for syntactic annotation; and column 10 can be used to store any annotation. Table 3.11 exemplifies the values that these columns would take for an Indonesian word annotated by MorphInd. (The LEMMA annotation given in this example is actually incorrect, but we will ignore this for the moment.)

1	ID	7	Token number 7
2	WORD	mengakomodasi	Word form of token
3	LEMMA	menakomodasi	Lemma of the word (all lowercase)
4	UPOS	VERB	UD Universal POS tag[24], in this case VERB as an expansion of V part of VSA in XPOS
5	XPOS	VSA	Language specific POS tag; in this case MorphInd's word POS tag, VSA
6	FEATURE	Number=Sing|Voice=Act	UD features, in feature=value form; in thisn case expanded from SA part of VSA in XPOS
7	HEAD	5	Token number of syntactic head of the current token
8	DEPREL	xcomp	UD relation, in this case open clausal complement[25]
9	DEPS	_	Enhanced dependency graph in the form of a list of head-deprel pairs, not available in this case
10	MISC	MorphInd=^meN+akomodasi<n>_VSA$	Any other annotation, in this case MorphInd's full output

Table 3.11. MorphInd's analysis for the word *mengakomodasi* 'accommodate' using CoNLL-U format (adapted from UD data example)[26]

In this example, column 10, MISC, contains the raw MorphInd (LM) analysis of the word token. This example actually represents Larasati et al.'s (2011) conversion of MorphInd's output to UDM, as part of which they preserve the original LM analysis in column 10. The morphological features in column 6 are expressed in terms of the UD *universal feature inventory* (Nivre 2015). These are additional lexical and grammatical properties of word not covered by the universal POS tags in column 4.

UD defines 17 universal major/coarse POS tags, a list which is fixed and cannot be expanded or customised[27]. The UPOS will normally be the tag among the 17 that is the best match for the word POS indicated by XPOS – as in Table 3.11.

This list of features is expandable, not fixed. The UDM documentation states that "Users can extend this set of universal features and add language-specific features when necessary"[28]. In Table 3.11, the features have been expanded from the morphosyntactic tag VSA. Column 4 expands V, while column 6 expands S and A. Column 5, XPOS, contains a language-specific morphosyntactic tag. It may use a non-universal tagset. Usually the XPOS tag reflects the

[24] https://universaldependencies.org/u/pos/index.html (last accessed 26/05/2021)
[25] https://universaldependencies.org/u/dep/index.html (last accessed 26/05/2021)
[26] https://github.com/UniversalDependencies/UD_Indonesian-GSD/blob/master/id_gsd-ud-dev.conllu (last accessed 26/05/2021)
[27] https://universaldependencies.org/u/overview/morphology.html (last accessed 26/05/2021)
[28] https://universaldependencies.org/u/overview/morphology.html (last accessed 26/05/2021)

FEATURE In this case, the XPOS comes from the LM word-level tag VSA, and the values in FEATS derive from the XPOS.

3.6.2. UDM as word-level analysis

UDM is morphological analysis performed at word level, not morpheme level. Moreover, it is designed for morphosyntactic annotation, not morphological annotation. This is explicitly mentioned in the UDM documentation (my emphasis in bold):

> The UD scheme allows the specification of **a complete morpho-syntactic representation** that can be applied cross-linguistically. This effectively means that grammatical notions may be indicated via word forms (morphologically) or via dependency relations (UDM web documentation) [29]

3.6.3. Limitations of UDM for morphological annotation

Linguists who wish to use UDM for morphological analysis at morpheme level may face difficulties. First, UDM does not include any tokenisation of morphemes. We saw earlier that words are lemmatised, but this is not sufficient for morphological analysis at morpheme level.

Second, UDM's morphological features are in practice limited to *inflectional* features only, as per the UDM documentation[30]. The fact that that UDM allows for the extension of the list of morphological features makes it obvious that UDM cannot be used *as is* for all languages. And the fact that the features are only inflectional is a further indication that UDM analyses are morphosyntactic, not morphological.

Third, the restricted possible values for UPOS tags mean that any category absent from the list of 17, such as classifier, must still be forced into one of the 17 possibilities, however poor the fit. This is a problem not only for morphological annotation but also for morphosyntactic annotation.

[29] https://universaldependencies.org/docsv1/u/overview/morphology.html (last accessed 26/05/2021)
[30] https://universaldependencies.org/ (last accessed 26/05/2021)

Fourth, the CONLL-U format does not allow each morpheme to be linked to the corresponding analysis. But this is simply an obvious consequence of the fact that the CONLL-U format is designed for morphosyntactic analysis not morphological analysis.

3.7 Best practices for morphological annotation schemes

The best practices for MASs identified here are those that will produce the best possible result for linguists, particularly corpus linguists who want to search a corpus based on morphological criteria, that is, to search for particular morpheme forms and/or particular morphological tags. Therefore, linguists whose research objectives are not of this kind, or non-linguists such as Information Retrieval (IR) practitioners, may find these best practices less relevant or indeed counterproductive.

Both IR practitioners and corpus linguists need to be able to 'search' a corpus in a generic sense. But the types of search, the degree of detailed annotation that they require, and the nature of the analysis of what is retrieved substantially differ. A number of the best practices that will be identified here could even be considered to promote excessive detail, from certain perspectives. But the goals of this thesis overall justify the definition of these best practices in a manner targeted to the needs of linguists studying morphology, particularly Indonesian morphology.

3.7.1 Morpheme-level analysis and word-level analysis

The MASs reviewed in this chapter exhibit two different levels of morphological analysis. The first is morphological analysis performed at word level (which is closely akin to morphosyntactic tagging); the second is morphological analysis performed at morpheme level.

I wish to argue here that morpheme-level analysis, as exemplified by BM (3.4.2), SAM 3.4.3), and Hutmegs (3.2.2), represents the best practice. Such MASs are characterised by explicit links between morpheme tokens and the tag(s) that encode their analysis. This can be established as a best practice for two reasons. First, users of an MAS are likely to need to need to search their annotated corpus based on analysis of morphemes. This requires the words to be *tokenised into morphemes* using *obvious demarcation symbols* that *delineate one morpheme token from the next.* This is a key distinction that characterises morpheme=level analysis, and is not a feature of either word-level morphological analysis or morphosyntactic tagging.

The examples in Table 3.13 present two hypothetical English MASs applied to the same words, one morpheme-level MAS and one word-level MAS, illustrating how the level of analysis would affect the tags and thus any corpus query system. The morpheme-level MAS follows the format of BM, while the word-level MAS follows the format of Fintwol.

Word	Morpheme-level MAS	Word-level MAS
painters	paint/V+er/NOMR+s/PL	paint, V NOMR PL
buses	bus/N+es/PL	bus, N PL
dogs	dog/N+s/PL	dog, N PL
cats	cat/N+s/PL	cat, N PL
oxen	ox/N+en/PL	ox, N PL
children	child/N+ren/PL	child, N PL
sees	see/V+s/SG	see, V SG
pass	pass/V	pass, V
energies	energy/N+es/PL	energy, N PL
operators	operate/V+or/NOMR+s/PL	operate, V NOMR PL

Table 3.12. Comparing morpheme- and word-level analysis with hypothetical English MASs

The morpheme-level MAS tokenises words to morphemes, clearly demarcating them with the plus symbol. Conversely, the word-level MAS, like many such MASs, presents separately only the root morpheme, followed by a series of morphological tags for the full word form; morphemes other than roots are usually left untokenised. Moreover, in this MAS, all the features tagged are inflectional (i.e. they are features which would be relevant to morphosyntactic tagging) as tends to be the case for such word-level MASs.

The absence of affix tokenisation in word-level MASs means that users cannot search a corpus by morpheme form. For instance, it would not be possible to search directly for the literal suffix *-s*. Querying the tag PL (within word-level bags of tags such as V NOMR PL or N PL) as an alternative would not be directly equivalent; this would return not only *-s* as in *painters*, but also *-es* in *buses,* as well as *-en* in *oxen* and *-ren* in *children*.

Performing underspecified searches on raw word forms is another alternative, but equally unsatisfactory. For example, using a wildcard to underspecify all letters prior to a final *s* (*s) would retrieve plurals such as *buses or energies* which contain allomorphs other than *-s,* as well as monomorphemic words such as *pass*. On the other hand, with a morpheme-level MAS, searching for suffix *-s* accurately retrieves *painters,* but excludes *buses* and *pass*. Even if the software is not annotation-aware, morpheme forms can be searched using literal queries that utilise the scheme's delimiter symbols. In the example at hand, a query such as +s/ uses + (morpheme boundary) and / (token-tag demarcator) around *s* to ensure that only tokens of the exact morpheme that the user has targeted will be retrieved.

Another reason why morpheme-level analysis is best practice is the anticipated need of users to perform retrievals based on criteria on morphological tags. Morphological analysis at morpheme level ensures that *every morphological tag corresponds to a morpheme*. If this principle is not followed, users cannot search for a tag that does not necessarily correspond to any single morpheme. This is best illustrated by contrasting two KKM examples. The monomorphemic word *katto* in (3,37) is tagged by Fintwol with tags *S NOM SG*: noun root, nominative case, singular number. A query for the NOM tag would allow users to retrieve *katto*. However, the tag NOM is not associated with any morpheme in *katto*, since the nominative in Finnish is a feature of word-level analysis (as discussed in 3.2.1.5). This is a breach of the principle established earlier: *each analysis must correspond to a morpheme*.

(3.37) *katto* (Koskenniemi 1983:157)
katTo
Roof S NOM SG

(3.38) *katon* (Koskenniemi 1983:158)
katTo$+n
Roof S GEN SG

This is underlined by the fact that unlike Fintwol, Hutmegs, a morpheme-level Finnish MAS (Creutz et al. 2004), has no tags for nominative or singular. On the other hand, if users of a Fintwol-tagged corpus perform a query for the tag GEN (genitive), one of the words returned will be *katon* as in example (3.38). Here, GEN corresponds to the suffix token *-n*; thus, it is a morpheme-level analysis which complies with the principle established earlier. In this case, a morpheme-level MAS like Hutmegs would not be different in principle from Fintwol.

So far, two best practices have been identified. First, each individual morpheme must be tokenised (using unambiguous demarcation symbols in the output), and second, each analysis (consisting of one or more tags) must correspond to an actual morpheme token. Fulfilling these two best practices allows users of the MAS to build corpus queries using either or both of morphological forms and tags.

For instance, returning to the English examples in Table 3.12 with the morpheme-level MAS, users could search for all noun roots that are followed by suffix *-s*, to retrieve *cats*, *dogs*, etc., by combining the morphological tag N and the morpheme form *s*, via a query which might be expressed as: */N+s/*. The first part of the query (*/N) matches any morph tagged with N, and the second part (s/*) ensures that the root is followed by a token with the form *s* without specifying any morphological tag. These two elements are connected by a plus marking the morpheme boundary; this instructs the software to retrieve all instances of two morphemes matching the respective conditions occurring in succession. Corpus annotation can be presented in many different ways, but morpheme level-analysis ensures that these two features, morpheme token-tag links and morpheme boundaries, will be expressed in any presentation style. The symbols used to represent these two elements may vary from one MAS to another, but it is best practice for the symbols to be consistently used and unambiguous. In BM, for example, a morpheme token and its tags are always separated by a slash. This symbol unambiguously links the token and tag and shows them to be paired. One token-tag pair and the next are separated by a plus (i.e. the morpheme boundary); see example (3.39). Similarly in Hutmegs, morphemes and their tags are linked by a vertical bar, and the space is reserved to mark morpheme boundaries; see example (3.40).

(3.39) *yaHotawiya* 'he/it contain (subjunctive)' BM
ya/IV3MS + Hotawiy/IV +a/ IVSUFF_MOOD:S

(3.40) *arvoamme* Hutmegs
arvo:arvo|N a:PTV mme:1PL

As long as this consistency and non-ambiguity are maintained, annotated output that is reformatted (for instance, for use in another search tool) still preserves the morpheme token-tag links and boundaries. That such reformatting can be achieved without breaching the best practices under discussion is shown by the example in Figure 3.3. Instead of using punctuation symbols to represent token-tag links and morpheme boundaries, like the original BM output format, the new format indicates links and boundaries using distinctive colours in a browser display, on the website for the Quranic Arabic Corpus (Dukes et al. 2009).

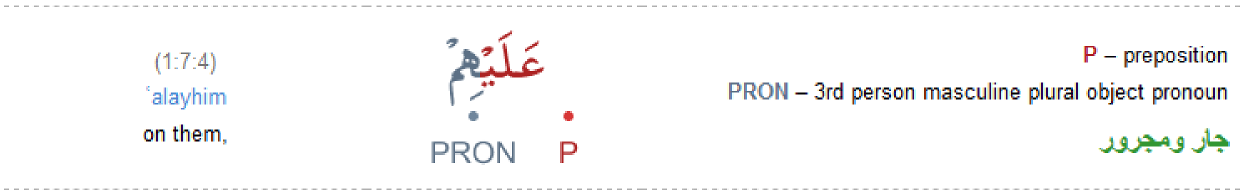

(1:7:4)
'alayhim
on them,
عَلَيْهِمْ
PRON P
P – preposition
PRON – 3rd person masculine plural object pronoun
جار ومجرور

Figure 3.3. BM morpheme-level analysis visualised in the online interface to the Quranic Arabic Corpus (Dukes et al. 2009)

Figure 3.3. shows that the word *'alayhim* 'on them', written in Arabic characters, is composed of two morphemes, rendered in different colours (red for *'alay* 'on' and grey for *him* 'them'). Under each morpheme appear the category analyses (P for preposition and PRON for pronoun). The vertical alignment reflects the link between morpheme tokens and their tags (textual English descriptions on the right side of the interface give more details of the analysis).

The clarity of the visual representation of BM in Figure 3.3 is sound evidence, I would argue, that morpheme-level analysis is indeed the best practice for MAS design. If implemented correctly, it offers many benefits for users, not only more precise queries, but also the possibility of flexible corpus data visualisation.

3.7.2 Orthographic and citation form

The previous section discussed the importance of tokenising all morphemes. It is also important to consider what forms the morphemes are tokenised into. In this section, I argue that tokenising each morpheme to both orthographic and citation form is best practice.

Searches based on orthographic and citation forms are functionalities that are likely to be required by users. For instance, users might want to retrieve all cases of English plural suffixes *-s* and *-es* in a single search, as they are allomorphs of one morpheme. In a corpus where morphemes are tokenised in orthographic form, one way to do this is by combining the two suffixes with a disjunction ("or") symbol, often the vertical bar. In that case, the query would be s|es. To accomplish this, users need to know all the orthographic forms that are variants of the morpheme for which they want to search. But in fact, this is not always the case.

The search can be more effective, particularly for morphemes with many allomorphs, if the citation form of the variant is encoded in the annotation scheme. So, for instance, if the citation form of *-s* and *-es* is represented as *S*, a query for S will retrieve all words that contain either suffix. (There would also need to be some mechanism in the query software for the user to specify whether they are targeting the orthographic or citation form, e.g. distinct query delimiters.) The presence of the two distinct representations of the tokenised units (orthographic and citation forms) affords flexibility of corpus searching.

Instead of tokenising all morphemes, a number of MASs (such as Fintwol, CM, and PM) tokenise only each word's root morpheme, and present it only in citation form, as in examples (3.41) to (3.43).

(3.41)	*koirillannekaan* "koira" N ADE PL 2PL kAA	Fintwol
(3.42)	*mem-(p)ukul* 'hit' pukul+Verb+AV	PM
(3.43)	*ev-de-ki* 'in the house' ev<N><loc><ki><Adj>	CM

This breaches the best practice of morpheme-level annotation (3.7.1). Tags for non-root morphemes cannot be linked to their morphemes due to lack of morpheme tokenisation. Obviously, since they do not tokenise *at all*, they equally do not tokenise to both forms, as I am arguing is best practice.

Other MASs, such as LM, BM and Hutmegs, do require all morphemes to be tokenised. But only Hutmegs fulfils the additional best practice argued for in this section, that tokenised morphemes should be presented in the output in both orthographic and citation forms.

(3.44)	*meng-(k)irim-kan* 'send (sth)' meN+kirim<v>+kan_VSA	LM
(3.45)	*yaHotawiya* 'he/it contain (subjunctive)' ya/IV3MS + Hotawiy/IV +a/ IVSUFF_MOOD:S	BM
(3.46)	*arvoamme* arvo:arvo|N a:PTV mme:1PL	Hutmegs

An LM analysis includes only citation forms. For instance, in (3.44), we see the prefix *meng-* presented as *meN+* instead of its orthographic form *meng*. Likewise, the root is presented in the analysis in its citation form, *kirim,* even though the orthographic form is *irim*[31]. Conversely, a BM analysis only presents orthographic form (or rather, a transliteration thereof) as shown in (3.45). But a Hutmegs analysis presents both orthographic and citation form, linking them with a colon as in example (3.46).

I argue that *presenting the morphemes of the words in both orthographic and citation form,* as in Hutmegs, *is the best practice*. To justify this, in Table 3.13, I present LM analyses of five Indonesian words, alongside a Hutmegs-style analysis of the same words (Hutmegs itself being Finnish only).

[31] The k is lost due to a morphophonemic process which I have described in the overview of Indonesian morphology in section 2.1.2.

	LM	Hypothetical Hutmegs-style MAS
mengirim	meN+kirim<v>_VSA	meng:meN|PFX+A irim:kirim|V
mengambil	meN+ambil<v>_VSA	meng:meN|PFX+A ambil|V
mengeras	meN+keras<v>_VSA	meng:meN|PFX+A ambil|V
menanam	meN+tanam<v>_VSA	men:meN|PFX+A anam:tanam|V
merebut	meN+rebut<v>_VSA	me:meN|PFX+A rebut|V

Table 3.13. Analyses of Indonesian verbs using LM versus a Hutmegs-style MAS

The LM analysis of *merebut*, for instance, does not allow an orthographic-form-based query for *me-*, because all allomorphs are presented only in their citation form *meN+*. Querying *me-* with an underspecified query on the raw word form (e.g. *me**) would result in many false positives – including every other word in Table 3.13, since all *meN-* allomorphs begin with *me*.

Using the Hutmegs-style MAS, however, searching for the specific allomorph form is feasible, because prefix *me-* is presented in orthographic form, *me*, followed by colon, and then the citation form. This enables users to retrieve all and only these instances of the *me-* allomorph of *meN-*. Equally the presence in the analysis of the citation form supports another anticipated need: queries to find all allomorphs of one morpheme. In practice, retrieving all the allomorphs (*meng-, men-*, *me-* in Table 3.13) would be achieved by a query for *meN+* in LM: or *meN*/ in the hypothetical Hutmegs-style MAS.

3.7.3 Formal and functional analysis

I have shown that the annotations encoded by MASs can be classified into two categories: formal and functional. Identifying each morpheme as root, prefix, suffix, infix, circumfix, enclitic or proclitic is an example of *formal* analysis, and we can refer to the tags that encode them as *formal* tags. On the other hand, such analytic categories as noun, verb, passive voice, agentive nominaliser, and many others mentioned in my review, exemplify *functional* grammatical analyses, encoded as functional grammatical tags. A number of the MASs that I reviewed include only functional tags, such as Fintwol, OM, CM, PM, and LM. By contrast, BM includes both formal and functional tags.

Formal analysis is, comparatively, neglected by MASs in the former group. I wish to argue that the approach taken by BM, where *both formal and functional analysis is captured by the MAS* is a best practice. Formal analysis is indispensable for the anticipated needs of linguist users. Queries based on formal categorisation are a foreseeable requirement for corpus querying in languages like Indonesian, where a variety of formal types of morphemes are present.

Formal tags can potentially be useful for improving both retrieval precision and the accuracy of quantitative analyses. Retrieval precision is reduced by morphological homographs, e.g. the locative preposition *di* 'at' and the passive voice prefix *di-* in Indonesian. But while these do not differ in orthographic or citation form, the former is a root and the latter is an affix, specifically a prefix. Here the distinction of the root versus prefix being made explicit by the tags is crucial. If a user searches for all instances of root *di*, they would expect prefix *di-* to be excluded from the results. Prefix *di-* would be a false positive. Avoiding these false positives and thereby improving precision cannot be achieved by a query for raw word forms that begin with *di,* as this would retrieve *di* as a preposition, *di* as a prefix, plus monomorphemic words that happen to begin with *di* such as *diam* 'quiet', *dia* 's/he', or *diri* 'self'*.* But if a ROOT tag is used, it can be included in the conditions of the query for *di*, and the false positives excluded. The same would work in reverse using a PFX formal tag to retrieve only prefix *di-*. Eliminating false positives to improve query precision is the first benefit of implementing formal as well as functional tags.

Formal tags are also useful for quantitative analyses. Let us consider a situation where a linguist wishes to calculate the morpheme per word ratio from a corpus or across corpora. For the sake of argument, let us assume that our corpus consists of only three words, those in (3.47) to (3.49) – repeated from the discussion of LM in section 3.5.2.

(3.47) *meng-(k)irim-kan* 'send (it) to'
meN+kirim<v>+kan_VSA

(3.48) *pukul-an* 'hit (n)'
pukul<v>+an_NSD

(3.49) *ke-jatuh-an* 'fall (n)'
ke+jatuh<v>+an_NSD

We observe that the LM analysis tokenises the three words into a total of 8 analytical units (each demarcated by plus): *meN-, kirim, -kan, pukul, -an, ke-, jatuh,* and *-an*. If each segment were a morpheme, the morpheme/word ratio would be 8/3, equal to 2.66. However, this calculation is inaccurate. There are actually only 7 morphemes in the above three words: *meN-, kirim, -kan, pukul, -an, jatuh,* and *ke—an*. Unlike the *+an* in (3.48), the *+an* in (3.49) is not a unit on its own. Rather, it is the second element of circumfix *ke—an.* LM does not distinguish circumfixes from combinations of prefix and suffix, and so cannot capture this subtlety.

Making the distinction between a circumfix and a homograph prefix-plus-suffix combination necessitates each affix being tagged for its formal category, using tags such as PFX (prefix), SFX (suffix) and CFX (circumfix). With this information, the correct ratio can be calculated. This information being present not only allows morpheme per word ratios to be calculated, it also makes possible many other quantitative analyses. For example, studying the productivity of affixes requires the same kind of fine distinctions for accurate quantification; looking at the productivity of suffix *-an* will be easier and more accurate when instances of circumfix *ke—an* are excluded from query results and frequency counts.

3.7.4 Tag encoding

Although the encoding of tags is theoretically independent from the definition of analytic categories, in practice users access the categories only via the tags. Thus, the fitness of the tags for this purpose is in fact of considerable importance. My review has identified a number of different practices regarding how the analyses in a MAS are encoded, namely that grammatical (and formal) categories may be represented by single letters, longer abbreviations, or full analytical labels.

For morphological analysis, I would argue that *schemes where analytic categories are encoded as abbreviations or full analytical labels* represent best practice, as opposed to schemes where each tag is a decomposable string of single-letter values encoding different analytic categories. I argued in section 3.5.2.3 that the decomposable tags used in LM can potentially cause memorisation problems for users

Decomposable tags are also usually hierarchical. For instance, the tag for singular common noun is NN1 in CLAWS (Garside 1987), more precisely the CLAWS-7 tagset[32], where the letters code respectively noun, common noun, singular. Each analysis is a subcategory of that before it.

There are three mechanisms by which letters to represent each analysis can be chosen. Some values are encoded as their initial letter, such as N for nouns, V for verbs and P for pronouns. Other values are not encoded as their initial letter, but as some other letter present in the full analytical label, as in R for adverb. The third way is for some otherwise unused letter to be chosen arbitrarily. An example might be C for numeral. The letter C is nowhere in the full analytical label *numeral* (it may perhaps abbreviate *cardinal*). These different ways of choosing letters makes the task of memorisation harder, as which of the mechanisms is in use must also be memorised.

The second memorisation problem of single-letter labels is position, that is, the letter's position within a decomposable tag. To illustrate this, consider three simple decomposable tags in LM: CC (cardinal numeral), CO (ordinal numeral) and CD (collective numeral). The letter in first position indicates the category at the top of the hierarchy, i.e. C alone means any numeral. The letters in second position encode subcategories of numeral. The letters C and O have been chosen by the first mechanism, and the letter D by the third mechanism (as D is not present anywhere in the full analytic label *collective*). When these letters are joined into a decomposable-hierarchical tag, their position is fixed. Users cannot reverse the order of CD to DC, because D in the first position signifies the adverb category in LM. This constraint affects how users must build queries: all letters of a tag being searched for have to be typed in the order that reflects the hierarchy – even if the query is underspecified with one or more wildcards. The query *D (wildcard prior to D) would find collective numeral morphemes, for instance. But collective numerals would not be retrieved by D* (wildcard following D) because specifying D in the first position would retrieve adverbs instead of collectives.

Users of a morphologically annotated corpus might need to retrieve items based on only one feature of the analysis, without specifying other features. This requires the query system to

[32] http://ucrel.lancs.ac.uk/claws7tags.html (last accessed 26/06/2021)

be flexible. Effectively applying positional restrictions (which, as noted, will often imply hierarchical restrictions) to all letters representing categories is an adverse effect which, for the sake of users, we might well wish to avoid.

In morphosyntactic (POS) tagging, a decomposable expression of the tag system's hierarchy of POS categories is often useful (Leech & Wilson 1999; Calzolari 1996; Sinclair 1996); but that this concept is equally applicable to morphological annotation is a position with which I do not completely agree. My counterargument to that position is presented in the next section.

3.7.5 Organisation of analytic categories

In any MAS where an analysis may consist of more than one tag, each tag indicating a different analytic category or categories, the question of how these tags are organised must be addressed. My view, based on my review of MASs, is that the best practice is to *organise tags using a semi-hierarchical approach*. Some analytic categories can be freely organised (using what is informally called a "bag of tags" approach). Other categories are hierarchically linked, and then the hierarchy of analytic categories must be expressed using decomposable-hierarchical tags within the morphological analysis.

The term *bag of tags* refers to the free organisation of independent analytic values that apply to a single token (here, morpheme token). For instance, the hypothetical tags SUFF (suffix), NOMZR (nominaliser), and PL (plural) encode values for features which are completely independent from one another. SUFF is an analysis of form, NOMZR is a derivation feature, and PL is a value for the grammatical analysis of number.

Any combination of these tag values therefore does not require the three values to be placed in any particular order; due to their independence, there is no hierarchy to reflect. None of the three is a subcategory of any other of the three. For this reason, how these categories are presented does not matter; their tags can be arranged like items placed in a *bag* in random or arbitrary order (thus the term *bag of tags*). SUFF_NOMZR_PL, or PL_NOMZR_SUFF, or SUFF_PL_NOMZR would all be acceptable renderings for the conjunction of these analytic categories.

The bag of tags organisation affords a degree of flexibility to queries using the annotation. For instance, users do not have to consider the position of each value in the sequence of the analytic elements; they are, thereby, spared the problem which I discussed in 3.7.4. To extract all plural morphemes, for instance, users would need only to run a query for morphemes whose annotation includes PL, regardless of order within the annotation.

Hierarchical analytic categories, which express some dependency, are a different matter. For instance, if we have formal categories AFX (affix) and SFX (suffix), then we have a dependency: suffix is a subcategory of affix. When a morpheme receives both, the tag's left-to-right order should reflect that, so AFX_SFX makes sense, but SFX_AFX would not.

This impacts how the MAS must be used in a query system. For instance, to devise a query to extract all suffixes, a user must remember that SFX is at the lowest level of the hierarchy. Thus, if they cannot remember the subcategorisation of the values, they would need to use a wildcard search to underspecify as the higher-level category, as for instance **SFX*; but *SFX** would be incorrect, as it underspecifies the hierarchy below rather than above SFX. In some cases, this may become cumbersome for little value (that all suffixes are affixes is true by definition), and higher level categories, such as AFX, might appropriately excluded from the MAS. In other cases, the hierarchical relations amongst analytic categories are necessary and *should* be reflected in the MAS and the tags that encode it. For instance, root morphemes may be analysed not only on the basis of major POS, but also on the basis of that major POS's subcategories. In CM, for example, when the POS analysis of a root includes two or more hierarchically-linked categories, the category and subcategory tags are connected by a colon, as in for instance <N:prop>. Here, proper noun is a subcategory of noun; the hierarchy cannot go the other way, and the features are not independent (see 3.3.2). As the very term suggests, "proper noun" is an analysis relevant to nouns but not verbs, adjectives or other POS categories. This hierarchy of categories is, I would suggest, useful (proper nouns should be found *both* by a search for proper nouns *and* by a search for all nouns), and must be preserved in the encoding of categories as tags.

The semi-hierarchical approach which I argue for combines the two approaches discussed so far, because it is possible for the categories relevant to a single morpheme token to be neither completely independent of one another, nor fully hierarchically dependent. To illustrate this, let us take an example from Indonesian, the clitics discussed in section 2.1.3.4.5. The personal pronoun *aku* 'I' can be cliticised into enclitic *=ku* (or proclitic *ku=*, but this is not relevant to the present point). This clitic has two independent features: its POS category of pronoun (functional) and its status as a clitic (formal). However, for each of the independent features, there is an additional category hierarchically dependent on the main category (for pronoun: personal pronoun; for clitic: enclitic or proclitic). POS category and morpheme form can thus be organised via the bag of tags approach, as they are independent, but the subcategories of each are dependent, and thus must be organised and encoded hierarchically.

Let us assume that PRON, Pers, CLI and ENCL represent, respectively, pronoun, clitic and enclitic. For the first person enclitic *=ku,* valid analyses would thus be PRON:Pers+CLI:ENCL and CLI:Encl+PRON:Pers. The relative ordering of PRON and CLI is arbitrary, but :Pers and :Encl must be placed following their superordinate categories, PRON and CLI. In this example, the colon serves to mark this as an explicitly hierarchical relationship.

Where tags represent hierarchical categories combined together, consistency of order is highly important. In section 3.4.2.3, I showed that BM breaches this principle. For instance, one BM tag NOUN_PROP (noun, proper) indicates a left-to-right hierarchy, while another BM tag DEM_PRON (demonstrative pronoun) implies a right-to-left hierarchy. The former order is, I would argue, generally more intuitive, and my hypothetical examples have used it. But either order is better than an inconsistent mixture.

To make full use of a semi-hierarchical MAS combining these two approaches (bag of tags and hierarchical), corpus search software requires a query mechanism that can accept different orderings of values – so that queries for, say, PRON+CLI and CLI+PRON are both accepted and yield the same results – but can also respect instances where the order of elements must strictly be maintained – so pers:PRON is rejected, or runs but matches nothing, whereas PRON:Pers works as expected.

The best practices that I have laid out in this section do not duplicate the approach of any one MAS that I reviewed. Rather, they synthesise current best practices across a range of MAS projects, addressing more languages than just Indonesian. The two most important benefits of a

MAS that implements these best practices are, in my view, linguistic accuracy (particularly, in my case, with regard to accurate description of Indonesian morphology) and practicality (for use in corpus searching systems allowing queries on morphological annotation as well as word form).

3.8 Summary

The best practices for morphological annotation schemes that I have identified in this chapter on the basis of a detail, cross-linguistic review of relevant prior work are as follows:

- The analysis should be performed at morpheme level
- Each morpheme should be given a separate analysis
- Morphemes should be tokenised within word forms and unambiguously linked to their analytic labels (or tags)
- Both the orthographic and citation (or canonical) forms of morphemes should be captured in the output
- A wide range of formal types of bound morphemes – prefixes, suffixes, infixes, circumfixes, proclitics and enclitics – should be incorporated
- Both formal and functional analyses should be taken into account
- The MAS should be expressed using tags based on abbreviations or full analytic labels and not on single letter labels
- Analyses whose values are independent from one another should be combined using the bag of tags approach
- Analyses whose values are dependent on one another should be combined hierarchically

My identification of, and arguments in support of, this list thus fulfil the aims established at the outset of this chapter. These best practices emerged from my review of a number of MASs for individual languages as well as one universal MAS (sections 3.2-3.6), MASs selected for review according to principles laid out in 3.1.

The level of detail of analysis that I argue to be best practice might well be considered to an over-analysis, or to include irrelevant fine distinctions – especially when seen from another field, such as information retrieval. However, these best practices are defined with the interests of corpus linguistic research in view, particularly into morphology, and specifically into Indonesian morphology, given that the purpose of this exercise was to develop a foundation for the creation of a new MAS for Indonesian. Defining this new MAS's analytic categories and the tagset that encodes them is the task of the next chapter.

CHAPTER 4

A NEW MORPHOLOGICAL ANNOTATION SCHEME

4.1 Principles of annotation scheme design

The objective of this chapter is to devise a novel Morphological Annotation Scheme (MAS) for Indonesian, to be applied by the automatic morphological annotator. One preliminary issue is that underlying principles to guide creation of this new MAS must be defined. To formulate these principles is this section's objective.

Few published studies discuss the principles of designing annotation schemes, as compared to studies presenting some particular tagset or tagger(s) which applies that tagset. Leech (1997) and Cloeren (1999) do discuss tagset design principles; however, both focus on morphosyntactic rather than morphological annotation.

Therefore, my approach is based primarily on the best practices summarised in 3.8. In what follows, these best practices are translated into guiding principles for the creation of a MAS for Indonesian.

4.1.1 Principle 1: To devise the MAS independently from technical implementations

The novel MAS will be constructed independently from considerations of technical *implementation*, that is, some automated system to apply the MAS. Some studies argue that annotation scheme and system are not independent, for instance that tagset granularity may affect a system's performance (Veronis & Khouri 1995; Mille et al. 2012). This first principle requires me not to take this issue of granularity into account at this stage. Rather, the MAS will be designed from the perspective of linguistic adequacy (particularly with respect to Indonesian morphology) and practical utility in corpus analysis. The MAS will be as granular as this objective requires. In consequence, it may or may not be feasibly implemented in full; changes may well be needed to facilitate implementation. However, if so, the unamended MAS in this chapter will serve to valuably inform any future work on implementations.

4.1.2 Principle 2: To focus the analysis at morpheme level

Treating morph(eme) tokens as the locus of the analysis is a best practice for which I have argued in section 3.8. This implies that, first, only morphologically marked categories are to be analysed; any functional grammatical categories expressed beyond the morphological level are excluded. Examples (4.1) to (4.4) illustrate this principle using the active and passive voices in Indonesian.

(4.1)	*dia*	*kena*	*pukul*	Analytic passive, syntactically/
	3s	get	hit	periphrastically marked
	'S/he got hit'			
(4.2)	*saya*	*mem-(p)ukul*	*dia*	Active, morphologically marked
	1s	ACV-hit	3s	
	'I hit him/her'			
(4.3)	*saya*	*pukul*	*dia*	Active, morphologically unmarked
	1s	hit	3s	
	'I hit him/her'			
(4.4)	*dia*	*di-pukul*		Passive, morphologically marked
	3s	PSV-hit		
	'S/he was hit'			

Active and passive will be captured by the MAS in cases like examples (4.2) and (4.4), where the voices are morphologically marked (by *mem-* and *di-* respectively). Conversely, the active and passive in (4.1) and (4.3) go beyond the morphological level. Determining that (4.1) is passive necessitates noting that *dia* 's/he' has the semantic role of patient despite being the subject in order to recognise the periphrastic construction formed by two root-form verbs: that is, syntactic, not morphological analysis. This excludes the analytic passive from the purview of this MAS. Ultimately, users must be aware that, when they search for morphemes tagged as passive, they will retrieve morphological passives as in (4.4), not analytic passive as in (4.1). The same will apply to any parallel cases involving other functional categories.

This principle secondly implies that the MAS will annotate only morphemes with non-zero form. In Buckwalter's MAS for Arabic, null allomorphs are introduced in some cases; for instance, a null suffix may be inserted and tagged as marking indicative mood (see 3.4.2.4). This approach is useful for Arabic, where the null allomorph is part of the verbal paradigm; the unmarked verb base has imperative mood, whereas verb bases with an affix in the same position have other moods. However, identifying null allomorphs would not be appropriate in Indonesian, as such paradigms are not a feature of Indonesian verbal morphology.

4.1.3 Principle 3: To unambiguously link each morpheme to its analysis

This principle reflects the best practice discussed in 3.7.1 and is one of the key distinctions between word-level and morpheme-level morphological analyses. I argued in 3.7.1 that breaching this practice can result in the inaccurate linking of one morpheme to the analysis of another morpheme. To avoid any potential for such errors, I adopt the principle that no such breaches will be permitted. This implies that the demarcation among morph(eme) tokens must be presented consistently and annotated explicitly .

To satisfy this purpose, polymorphemic words must be tokenised into morphemes to permit eventual users to perform morphological searches of annotated text, that is, queries based on the forms of, or analytic tags applied to, morph(eme)s.

This implies three requirements. First, morpheme boundaries must be explicitly marked. This ensures that (for instance) when users search for a word composed of three unspecified morphemes, words with two morphemes will not be returned. As such, explicit boundaries make morpheme counts more accurate, assisting quantitative analyses.

Second, morpheme forms as well as annotations must be presented for each bounded element. In this MAS, morpheme boundaries are indicated by angle brackets, such that each pair of angle brackets encloses one morpheme's form(s) and formal and functional tags. So for instance, the word *dipukul*, glossed in (4.5), is tokenised as in (4.6).

(4.5) *di-pukul*
PSV-hit
'be hit'

(4.6) <di,PFX+PSV>
<pukul,ROOT+VER>

In (4.6), the forms of *di-* and *pukul* 'hit' are given after the opening angle bracket. Tags for analytic categories follow. Labels PFX and ROOT encode the formal analyses of *di-* and *pukul* as a prefix and a root, respectively. Labels PSV and VER encode functional analyses (passive marker, verb root). Finally, a closing angle bracket completes the annotation of each morpheme.

4.1.4 Principle 4: To present morphemes in both orthographic and citation forms

The principles that morphemes' orthographic and citation forms must both be present in analysed output is among the best practices that I identified (see 3.7.2). In effect, the citation form corresponds to the *morpheme* whereas the orthographic form corresponds to one of its *allomorphs*. Incorporating both forms allows users to search a corpus based on *either*, enabling corpus analysis of both morphemes and allomorphy. The two existing Indonesian MASs reviewed in sections 3.5.1 and 3.5.2 both fail to comply with this principle. They therefore do not permit this kind of dual query. Accommodating *both* forms in this MAS will avoid this serious limitation of earlier research.

Practically speaking, if a morpheme's orthographic and citation form differ, the format is as follows: orthographic form, comma, citation form, comma, as in (4.7). If the orthographic and citation forms do not differ, only one form is presented, followed by a comma, as in (4.8).

(4.7) <**meng,meN,**PFX+DRV:VER+ACV>

(4.8) <**di,**PFX+DRV:VER+PSV>

Example (4.7) illustrates a morphophonemic alternation: *meng* is the orthographic form of the allomorph, whereas *meN-* is the morpheme's citation form. In (4.8), only *di* is given, as *di* does not exhibit any allomorphy.

For presentation of orthographic form, this MAS does not take into account whether the morph is realised in upper or lower case in the actual text. For instance, English *undo* might be written as *Undo* (so-called title case, perhaps sentence-initially) or *UNDO* (uppercase, perhaps for emphasis) or even *UnDo* (mixed case). The orthographic form given in this MAS's output ignores these variations, collapsing to a single case form. It is expected that users who desire case sensitivity would therefore query raw rather than annotated text.

4.1.5 Principle 5: To present formal and functional analyses in the annotations

I argued in section 3.7.3 that best practice is for a MAS to incorporate analysis of a range of formal types of bound morphemes (roots, prefixes, suffixes, infixes, circumfixes, proclitics and enclitics), as well as formal types of reduplication, in anticipation of users wanting to query texts and corpora based on these categories. Another user need that this MAS anticipates is searches based on grammatical functions of morphemes, e.g., voice or root POS category.

In this MAS, both types of analytic label are presented after the forms, delimited by a plus, as in (4.9) and

(4.10). Here, and throughout this thesis, formal analyses (PFX i.e., prefix and ROOT) precede functional analyses (PSV i.e., passive and VER i.e., verb). However, in implementation terms, no order is prescribed: each analysis is an unordered set of tags.

(4.9) <di,PFX+PSV>

(4.10) <pukul,ROOT+VER>

4.1.6 Principle 6: To use reference grammars as the main basis for analytic model

This MAS will draw its analytic categories mainly from existing reference grammars of Indonesian. Two such grammars are well-known. The first is *Tata Bahasa Baku Bahasa Indonesia* (TB3I), the 'Standard Indonesian Reference Grammar' (Alwi et al. 1998), written in Indonesian. This grammar is published by *Badan Pengembangan dan Pembinaan Bahasa* (BPBB), an official Indonesian government body dedicated to development and cultivation of languages and literature in Indonesia. The *Indonesian Reference Grammar* (Sneddon et al. 2010), by contrast, is written in English. Sneddon et al.'s reference grammar is not a translation of Alwi et al.'s, however; they were independently written. The categories in my MAS will be defined primarily on the basis of these two grammars.

Textbooks on Indonesian morphology, which cannot be considered reference grammars, but whose contents nevertheless are directly relevant to the MAS, will be considered as auxiliary sources. Kridalaksana (1989) and Chaer (2008) are examples of such publications.

A number of recent studies have offered new insights into Indonesian grammar, some of which might be relevant to the MAS. For instance, Nomoto (2013) studies prefixes active *meN-* and passive *di-* from an aspectual perspective and argues that they are markers of eventiveness and telicity. On principle, work of this kind will not be considered in the creation of this MAS. The aim of the MAS is to represent a morphological model of Indonesian that captures a broad, established picture of the overall system, not detailed cutting-edge research of specific facets of the grammar. Basing the analytic categories on published reference grammars achieves this aim.

4.1.7 Principle 7: To synthesise categories from the reference grammars based on relevance and genericness

Sneddon (2010:65) argues correctly that there is considerable variation in the categories presented by different authors of Indonesian grammars – and even more in Indonesian morphology textbooks. As indicated above, I draw on several of these sources, whose accounts of any given analytic category may differ from one another. How can such disagreements be resolved?

I will not attempt to incorporate all categories documented in all sources utilised. Instead, I will synthesise different accounts on the basis of *relevance* and *genericness*. The principle of relevance is derived from principle 2, i.e., morphologically marked categories are relevant, and others will not be considered.

In some cases, eliminating non-relevant categories is sufficient to synthesise the accounts of multiple sources. For instance, prefix *peN-* is usually discussed as a nominaliser prefix, but in some sources the discussion also includes its semantic categorisation as agentive or instrumental nominaliser (not directly relevant to morphology). This analysis can be discounted. In this case, the synthesis is a nominaliser prefix without any further semantic categorisation: the most relevant analysis in morphology.

In other cases, it is necessary to apply the genericness principle, as follows. The genericity principle refers to the use of the most generic property when a morpheme is ambiguously categorised. For instance, *ber-* is described by Sneddon (2010: 66-69) as marking four different categories when prefixed to a noun base (for present purposes, it does not matter exactly what these are). But Alwi et al. (1998:138) describe the same prefix as marking only three functions. Chaer (2008:106) argues that *ber-* with a noun base marks 12 categories; Kridalaksana (1989: 44) counts 19. The number of categories given by the latter two sources reflects the fact that morphology textbooks tend to be more fine-grained in their categorisation than the reference grammars.

However, I have found that all these sources have one thing in common: analysis of *ber-* as a verb-forming derivational prefix (Alwi et al. 1998: 137-142). The various other functions suggested by different authors are secondary or special cases of this, using subclassification based largely on the semantics of the root, and sometimes not adequately described. The genericness principle therefore dictates including in the MAS only one analysis of *ber-*: deriving verbs from roots (of any kind; see further □3.5.1.1.4.2.3.3.1).

4.1.8 Principle 8: To use the bag of tags approach to combine independent analytic categories

Analytic categories can be dependent on, or independent of, one another. In section 3.7.5, I argued that the *bag of tags* approach should be used to combine tags within an analysis whose values are independent of one another (that is, values for different features that are not subcategories of each other). An example is the combination of *plural* a category whose feature is number) and *suffix* (whose feature is affix type): these are *independent* values (see 3.7.5).

This approach fits the morphology of Indonesian, particularly for affixes and roots, whose analyses include multiple categories from unrelated features, formal and functional. The two Indonesian MASs reviewed in 3.5 do not attempt such a comprehensive analysis. However, since my MAS *does* attempt that, the bag of tags approach is required. For these reasons, this best practice is adopted here as a principle.

The symbol that delimits tags representing independent analytic categories in this MAS is the *plus* (+) symbol. Thus, 'PFX+PSV' is a combination of two tags, a formal category 'PFX' (prefix) and a functional category 'PSV' (passive).

(4.11) <di,PFX+PSV> correct use of connecting symbol(+)

The order of tags within a bag of tags is not meaningful. Therefore, the analysis 'PSV+PFX' would be 100% equivalent and acceptable. However, for the sake of consistency and easy reading, formal categories are always presented before functional categories in this chapter.

4.1.9 Principle 9: To hierarchically combine analytic categories that are dependent

Values that depend on one another will be combined hierarchically in this MAS's tagset, as per the best practices discussed in 3.7.5. An example given there is the relationship of noun and proper noun, the latter being a subcategory of the former; thus, *proper noun* is an analytic value that is dependent on *noun*. If sets of dependent values form a hierarchy, it is best practice for tags to represent the hierarchy.

To comply with this best practice, tags for dependent categories need to be linked to each other by a consistent symbol, which in this MAS will be the *colon* (:). Let us examine the derivational outcome POS of *peN-* (Alwi et al 1998:225). This is functionally a nominaliser, represented by the tag DER:NOU.

DER:NOU reflects the hierarchical relationship between two analytic categories; it is a derivational morpheme (DER) which derives a noun (NOU) from its base. In contrast to a bag of tags, the order cannot be swapped: NOU:DER would not reflect the hierarchy. A group of dependent units joined hierarchically is then, in turn, joined to other tags (or linked sets of tags) using +, as in example (4.12). The uses of : (dependent) and + (independent) are thus mutually exclusive.

(4.12) <peng,peN,PFX+DER:NOU>

4.1.10 Principle 10: To devise this MAS's tags using the most widely accepted terminology

This MAS will encode its analytic categories into tags using the most widely accepted terminology. The need for this principle can best be illustrated by the terms *active* and *passive*, as applied to Indonesian. A number of scholars use alternative terms, such as *subject-object focus* (Johns & Stokes 1996), *agent-object orientation* (Prentice 1987:193) or *agent-patient voice / actor-patient voice* (Mistica et al. 1999). Each of these scholars argues that these new terms capture relevant differences between this Indonesian phenomenon and the active and passive voices found in various European languages.

In this thesis, I am in no position to examine their arguments further or to come to a conclusion. However, I would argue that these terms are not yet widely recognised; conversely, even among linguists who do not engage with theoretical discussions of voice, the terms *active* and *passive* are very widely recognised. This is evident, for instance, from Djawanai's (1999:28-37) survey of Indonesian *diathesis* (another term for the active-passive distinction). Using novel terminology for tags for such categories could potentially cause users who are not familiar with the terms to interpret a category as absent when it is actually present, but under a different label. The principle of avoiding all but widely accepted terminology is adopted to prevent this from happening.

4.1.11 Principle 11: To encode tags using multi-letter or full analytical labels, not single letters

In section 3.7.4, I argued that single letter tags have a number of potential drawbacks, particularly when combined to form larger tags; and that multi-letter and full analytical labels are easier to remember and fit better with search systems for morphologically annotated corpora. The present MAS follows this best practice.

4.1.12 Principle 12: To encode the MAS's tagset in both Indonesian and English

In any corpus search interface for texts annotated with this MAS, users will access the annotation via the category labels. For this reason, it is important that these labels, the tags, should be understandable. One aspect of the understandability of a tagset is the language from which the labels are drawn; this is almost always English.

All the MASs discussed in section 3.2 to 3.6 use English-based tags, including Fintwol for Finnish (section 3.2), OM for Turkish (section 3.3), BAMA for Arabic (section 3.4), and PM for Indonesian (section 3.5), although all analyse a non-English language. None reports how usable English tags were found to be for the users that the creators of these MASs and associated implementations may have had in mind.

One possible reason for this omission is that English is a language with which the target audience is already familiar. For most of the schemes reviewed, the target audience is not explicitly stated, but is likely to be NLP practitioners or linguists already used to working with English as the language of scholarly communication. No such assumption will be made for this MAS. The main audience for this MAS is expected to be speakers of Indonesian who may or may not be familiar with English. Thus, I will encode the tagset for this MAS in *both* Indonesian *and* English. Users more familiar with Indonesian can rely on the Indonesian tagset, and users who speak little or no Indonesian, e.g. foreign scholars engaged in multi- or cross-lingual research, can rely on the English tags. In the account of the MAS later in this chapter, Indonesian-language labels and English-language labels for a category are referred to as that category's *I-tag* and *E-tag*, respectively.

4.1.13 Principle 13: To analyse loan and foreign words as roots

In this MAS, loan and foreign words will be tokenised as if they were monomorphemic, regardless of their morphological structure in the source language. Some foreign words have been fully adapted to Indonesian, such as *marketing* 'marketing' and *fenomena* 'phenomenon' (English) or *ulama* 'Islamic cleric' (Arabic). The internal structures of such words are evidently lost in Indonesian; for instance, *-ing* (in *marketing*) would not be considered a morpheme in Indonesian even though it is a suffix in the source language. Likewise, *fenomena* and *ulama* are plural in their source languages (English, Arabic). However, as loans in Indonesian, they are singular. For this reason, foreign words and loanwords will not be analysed internally, but treated as unaffixed root morphemes, tokenised following the procedure for monomorphemic words. Consequently, loan affixes such as *-isme, -isasi,* and *-logi* in words such as *paleontologi* 'paleontology'*, komunisme* 'communism' and *grafologi* 'graphology' will not be not treated as affixes by this MAS. Loanwords with these affixes are considered monomorphemic.

4.1.14 Principle 14: To not treat multiword expressions differently from sequences of single words

This MAS will not apply any special treatment to multiword expressions. Each morpheme of each word of multiword expressions will be analysed as-is. This includes morphemes within compounds and idiomatic or fixed expressions. While the identification of multiword expressions, and their internal structure in terms of the component words' interrelations, is an interesting and valuable analysis, it is not a matter of morphology. Even if it were desirable to represent multiword expressions within morphological annotation, doing so would drastically complicate any attempt to build a MAS – potentially confusing the important distinctions between an MAS and a morphosyntactic annotation scheme or semantic annotation scheme (due to the non-compositional meaning in fixed expressions and idioms).

4.1.15 Principle 15: To provide categories that disambiguate homographs at morpheme level

Often, annotation of analytic categories can disambiguate homographs (units with the same form, but more than one function/meaning). In English, for instance, suffixes *-er* in *learn-er* and *smart-er* mark distinct functions (agentive nominaliser and comparative degree adjective marker, respectively). Identifying the base POS suffices to determine which function is in use: nominaliser *-er* applies to a verb base while comparative *-er* applies to an adjective base. Examples (4.13) and (4.14) illustrate this concept in Indonesian.

(4.13) *ter-ambil* — *teR-* with verb base
PSV.Accidental-take
'be taken accidentally'

(4.14) *ter-cantik* — *teR-* with adjective base
SPV.Adj-beautiful
'The most beautiful'

Ter- marks the accidental passive (see 2.1.3.4.2.2) in (4.13) and the superlative in (4.14), respectively with a verb base and with an adjective base. To allow for disambiguation of this polysemous morpheme, the MAS must include both categories.

Another polysemous Indonesian affix is *ke-* (Sneddon 2010:61-62), but this is a counterexample to the principle above. Prefixed to a cardinal numeral such as *dua*, *ke-* can indicate either an ordinal numeral, as in *orang ke-dua* 'second person', or a collective numeral, as in *ke-dua orang* 'both people'. Disambiguating *ke-* requires information beyond the morphological level: in this case, whether *ke-dua* appears as a determiner or modifier of its head. Following the head, *ke-* forms an ordinal numeral modifier ("*N*th *X*"); prior to the head, prefix *ke-* forms a collective numeral determiner ("all *N Xs*").

Since this disambiguation operates at the syntactic level, this MAS will not include different analytic categories for these two uses. Rather, both uses of *ke-* will receive the same analysis. This is a point of contrast to the LM annotation scheme (see 3.5.2) which disambiguates this polysemy, with tags O and D for ordinal and collective numerals respectively (Larasati et al. 2011:123). In this MAS, unlike LM, any disambiguation which requires linguistic information from beyond the morphological level (morphosyntactic, syntactic, semantic, or any combination thereof) will not be taken into account.

4.2 A novel morphological annotation scheme for Indonesian

4.2.1. Tokenisation

4.2.1.1. Tokenisation of affixes and roots

Following principle 4, each morpheme's orthographic and citation forms will be included within analyses in this scheme . When orthographic and citation forms do not differ, only one form is presented (see xample (4.8) in 4.1.4). When they differ, for instance due to nasal assimilation of prefix *meN-* as discussed in 2.1.2, principle 4 dictates that the MAS should represent both, as shown by example (4.7) in 4.1.4.

The question now is how the *meN-* root boundary should be tokenised so as to fit best with users' anticipated needs for orthographic search. In example (4.7), the phonological process merely selects an allomorph, without affecting the root. However, there are cases when the phonological process removes the root's first consonant (in the phonological environments outlined in 2.1.2), as in example (4.15). Here, the root-initial consonant /p/ is lost through the same process that selects allomorph *mem-*. In example (4.15), the transcription of *memakai* as *mem-(p)akai* reflects these processes. But this representation is not appropriate for a morphological annotation, due to its *ad hoc* complexity.

(4.15) *mem-(p)akai*
ACV-use
'use'

In this MAS, in the orthographic transcription, I retain the allomorph *mem-* for the prefix, and *akai* for the root. That this segmentation is widely accepted is evident in reference grammars (Alwi et al. 1998:111), in which *memakai* is segmented as *mem-akai*. To capture the root's initial consonant loss, an additional analytic label, +*Luluh* or +LOST, is added to any root whose first consonant is omitted due to morphophonemic changes, as in (4.16). Users can thus search for +LOST and obtain all tokens of roots whose first consonant is lost, without having to specify the lost consonant.

(4.16) <**mem**,**meN**,PFX+DER:VER+ACV>
<**akai**,**pakai**,ROOT+UNCLT+VER+LOST>

From the perspective of generative morphology, polymorphemic words are constructed by the operation of a sequence of word formation rules (see 6.4), applied in a set order. Some Indonesian scholars adhere to this generativist characterisation. For instance, Chaer (2009:33) manifests this view while discussing the verb *ber-pakai-an* 'dress (oneself)'. He argues that the rule sequence begins with suffixation of nominaliser *-an* to verb root *pakai* 'wear'. Subsequently, the resulting word *pakaian* 'dress' serves as base for prefixation of verbaliser *beR-*, which finalises the orthographic word form, *berpakaian*.

The view of word formation through an ordered sequence of morphological processes is not used as an ordering principle by this MAS. If a suffix is deemed to be added by an earlier rule than a prefix, then annotation encoding an ordered generative analysis might reflect this in the arrangement of the two affixes' tags. But in this MAS, rather, all morpheme tokens are presented *linearly*, with no indication of which processes precede or follow any other. This approach is demonstrated by the tokenisation examples in sections 4.2.1.1 to 4.2.1.3, which do not exhibit any effect of sequence of affixation. Chaer's sample word *berpakaian* is linearly tokenised as shown in (4.17).

(4.17) *ber-pakai-an*
<ber,beR,PFX+DRV:VER+RFLX>
<pakai,ROOT+VER>
<an,SFX+DER:NOU>

Why does this MAS adopt linear presentation style rather than a style reflecting a sequence of processes? I argue that presenting the order of morphemes linearly is a more neutral approach, as it simply mirrors how they are presented in text. Thus the concept is easier for users to grasp, whether or not they are generativists. Moreover, linear presentation means that when querying the annotation, users do not need to specify the ordering of different analyses. For example, a style of annotation that included sequencing would require examples like (4.17) to be found using queries in which the *-an* is explicitly prior in the derivation to the *ber-*. The linear presentation adopted by MAS allows users to ignore this issue, and build queries that refer only to the orthographic sequencing of morphemes; a query for <ber><pakai><an>, or the same query with one or more underspecified elements, would find *berpakaian*. There is only one exception, namely treatment of circumfixes, which section 4.2.1.2 will address.

4.2.1.2. Special remarks on infixes and circumfixes

Principle 3, that each morpheme must be unambiguously linked to its analysis and presentation, is problematic when we deal with discontinuous morphemes. This issue arises for circumfixes and infixes.

In existing glossing conventions (as in Comrie et al. 2008; Lehmann 2004), a circumfix is usually presented as a discontinuous sequence *around* its base, as in example (4.18), following Comrie et al.'s proposal; and an infix is usually presented *within* its base, as in example (4.19), following Lehmann (2004:1858).

(4.18) *pem-betul-an*
NOMZR-correct-NOMZR
'correction'

(4.19) *t<el>unjuk*
<AGNR>point
'forefinger'

Tokenising the elements of a circumfix as a discontinuous sequence is ambiguous. The sequence can be read as one morpheme or two morphemes. For human linguists, this ambiguity is relatively unimportant. However, when this glossing convention is used within machine-readable rules, it becomes an issue.

Intervening an infix within its root makes the root discontinuous. The elements on the first line and the elements on the second line no longer line up unambiguously. Again, the ambiguity might not be problematic for human linguists, but *can* be for automated annotation.

In this MAS, despite the potential problems, I follow the glossing convention. A circumfix is annotated as two discontinuous parts around a base. The advantage of this is that the order of annotation is *natural*, in the sense that the presence of an opening and a closing analysis for each circumfix, each with orthographic/citation form plus tags, mirrors the actual order of the elements of the root-plus-circumfix combination.

The drawback is that the tokenisation of a circumfix is then similar to that of a prefix and suffix combination. But this can be compensated for by using tags to explicitly mark the opening and closing parts of the unit, here +A and +Z, as in <*OpeningCircumfixUnit*,CFX+A> and <*ClosingCircumfixUnit*,CFX+Z>, where to be valid the two elements must occur in the same word. This distinguishes the circumfix from the corresponding prefix and suffix combination, in which the latter elements would be annotated using the prefix (PFX) and suffix (SFX) category labels, rather than the circumfix label (CFX).A correct annotation of circumfix would be as in (4.20).

(4.20) *pem-betul-an*
<pem,peN,CFX+A+DER:NOU>
<betul,ROOT+ADJ>
<an,CFX+Z+DER:NOU>

An infix's annotation precedes that of its root, as (4.21) illustrates. Unlike the practice for circumfixes, this representation is not natural, although the treatment of both follows glossing convention. But this approach has the advantage of coding the root intact, making its representation in the annotation less ambiguous. This serves the needs of users, as it allows queries for citation form *tunjuk* 'point at' to return all instances of *telunjuk* as well as of orthographic *tunjuk.*

(4.21) *t-el-unjuk*
<el,IFX>
<tunjuk,NOU+ROOT>

Why should the annotation or gloss for the infix precede that for the root, and not the other way around? Lehmann (2004:1858) does not give any reason. For either glossing examples or devising a MAS, arguably whatever layout is used, the choice of order could always be questioned. Yet still, it is necessary to make a decision in order to proceed. So, the need for this arbitrary technical decision is not a major drawback.

4.2.1.3. Tokenisation of reduplications

There are three types of reduplication, full, partial, and imitative, as discussed in 2.1.3.4.3. The question is, how are these reduplications to be tokenised in this MAS? The first alternative is to tokenise the whole reduplicated word or word-part as one unit (as previous Indonesian MASs did; see 3.5). But this implies word-level morphological analysis. Thus, this alternative must be ruled out. The second alternative is to tokenise reduplicated parts as units at morpheme level. This prevents the MAS from breaching principles 2 and 4. This is exemplified by examples (4.22) to (4.26).

(4.22) *buku-buku* full reduplication
<buku,ROOT+NOU>
<buku,RED:FULL+DER:NOU+PLUR>

Any non-reduplicated morphemes must not be tokenised as a part of the reduplication. For instance, *meN-* in (4.23), which intervenes between *pukul* 'hit' and its copy, is not reduplicated, and thus must be separately analysed.

(4.23) *pukul-mem-ukul* full reduplication with affixation
<pukul,ROOT+NOU+VER >
<mem,meN,PFX+DER:VER+ACV >
<ukul,pukul,RED:FULL+DER:VER+ITRV+RECP>

Conversely, in full-word reduplication, the whole word is copied. In this case, all the copies of the morphemes that make up the reduplicated word are treated as distinct tokens. For example, the compound *es krim* 'ice cream' is composed of two roots; see (4.24). Its reduplicated form is annotated as if it consisted of four elements: two separate copies for each of the two morphemes. I am aware that some scholars may disagree with this, arguing instead that the whole-word reduplication should be considered as a single morpheme (for three tokens in all), but I will argue (in 4.2.4 and 4.2.5) that analysing four morpheme units here is a *useful fiction*.

(4.24) *es krim-es krim*
<es,ROOT+NOU> full word reduplication
<krim,ROOT+NOU>
<es,RED:FULL+DRV:NOU>
<krim,RED:FULL+DRV:NOU>

The citation forms of imitative and partial reduplication morphemes are considered to be identical to the roots which they duplicate; see (4.25), (4.26).

(4.25) *sayur-mayur* imitative reduplication
<sayur,ROOT+NOU>
<mayur,sayur,RED:IMTV+DER:NOU+PLUR>

(4.26) *te-tangga* partial reduplication
<te,tangga,RED:PART+DER:NOU>
<tangga,ROOT+NOU>

By adhering to this approach, whether or not non-reduplicated morphemes occur within the reduplicated word, the annotation of each morpheme is accessible (in addition to the annotation of the reduplication morpheme itself). Conversely, the word-level approach to tokenising reduplication would produce annotation of the sort in

(4.27), where the polymorphemic word *pukul-memukul* 'hit one another' is not annotated at morpheme level. This blocks access to morphological tokenisation and annotation, prohibiting searches from retrieving morpheme tokens within reduplicated words.

(4.27) <pukul-memukul,RED:FULL+VER+ITRV+RECP>

4.2.2. Roots

4.2.2.1. Formal analytic categories for root morphemes

The first analysis of a root morpheme is the identification of its formal category as a root, encoded as ROOT (E-tag) or *AKR* (I-tag) (see Table 4.1); this distinguishes roots from affixes and reduplication morphemes. Some examples are given in Table 4.2, where the tag under discussion is boldfaced (the other tags present will be discussed later).

	I-tag	E-tag	Examples
Root	*AKR*	ROOT	*layar* 'screen', *jatuh* 'fall'

Table 4.1. The formal analytic category for root morphemes

Description	Full I-tag	Full E-tag
Nominal root morpheme	*<layar,**AKR**+NOM>*	<layar,**ROOT**+NOU>
Verbal root morpheme	*<jatuh,**AKR**+VER>*	<jatuh,**ROOT**+VER>

Table 4.2. The root category within full analyses

4.2.2.2. Formal analytic categories for cliticised root morphemes

A number of Indonesian roots can be cliticised (see 2.1.3.4.5.). Users might want to search based on the cliticised-root category to avoid also retrieving the uncliticised root (free form roots), or the other way around. Thus, it is important for the MAS to distinguish cliticised and uncliticised root morphemes as separate categories. Examples (4.28) and (4.29) illustrate the enclitic and proclitic forms of first person pronoun *aku* 'I/my'.

(4.28)	*rumah=ku* house=1s 'my house'	pronominal enclitic (possessor)
(4.29)	*ku=ambil* 1s=take 'I take'	pronominal proclitic (subject)

In some cases, the cliticised and uncliticised forms are identical. This is the case for emphasis particles (*lah, pun*), one question particle *kah*, and one pronoun *(eng)kau* 'you', an alternative to the main second person pronoun *kamu* 'you'[33]. When these roots morphemes are cliticised, their forms are not different; they are simply attached to their host without any space between (*=lah, =pun, =kah, kau=*, etc.).

The analytic categories for cliticised and uncliticised roots are as follows. If a root is not cliticised, its formal tag is ROOT as per 4.2.2.1. If a root is cliticised, an additional category is given: *PKLT* or PCLT for proclitic, *EKLT* or ECLT for enclitic. This means that if a user should wish to search for a clitic without specifying its position, they can use a wildcard search on the clitic tag (e.g. *KLT).

In the literature on Indonesian grammar, the particles which can be written identically as free forms or as bound forms are never categorised explicitly as clitics. However, by describing these elements as possibly free or bound, scholars implicitly acknowledge that they exhibit the behaviour of clitics. For this reason, such "attached" particles are categorised as clitics in this MAS. (Special considerations apply to enclitic *=nya*; see section 4.2.6.)

[33] The sociolinguistic differences among alternative second person singular pronouns (*kamu, (eng)kau, Anda*) are discussed in Alwi et al. (1998:250-260).

Independent form	Clitic form	Type	I-tag	E-tag
Pronouns (personal/possessive)				
aku (1s)	*ku=*	Proclitic	*PKLT*	PCLT
aku (1s) *kamu* (2s) *dia* (3s)	*=ku* *=mu* *=nya*	Enclitic	*EKLT*	ECLT
Numeral				
satu	*se=*	Proclitic	*PKLT*	PCLT
Particle				
pun	*=pun*	Enclitic	*EKLT*	ECLT
lah	*=lah*			
kah	*=kan*			

Table 4.3. Formal categories for cliticised roots, organised based on POS

Clitic	I-tag	E-tag
ku=	*<ku,aku,AKR+**PKLT**+PRO>*	<*ku,aku,*ROOT+**PCLT**+PRO>
=ku	*<ku,aku,AKR+**EKLT**+PRO>*	<*ku,aku,*ROOT+**ECLT**+PRO>
=mu	*<mu,kamu,AKR+**EKLT**+PRO>*	<*mu,kamu,*ROOT+**ECLT**+PRO>
se=	*<se,satu,AKR+**PKLT**+NUM>*	<*se,satu,*ROOT+**PCLT**+NUM>
=pun	*<pun,AKR+**EKLT**+PKL>*	<*pun,*ROOT+**PCLT**+PAR>
=kan	*<kan,AKR+**EKLT**+PKL>*	<*kan,*ROOT+**PCLT**+PAR>
=lah	*<lah,AKR+**EKLT**+PKL>*	<*lah,*ROOT+**PCLT**+PAR>
=kah	*<kah,AKR+**EKLT**+PKL>*	<*kah,*ROOT+**PCLT**+PAR>

Table 4.4. Formal categories for cliticised roots within full analyses

4.2.2.3. Functional analytic categories for root morphemes

Part-of-speech is a category which applies to both roots and entire words; it is therefore a matter of morphology as well as a matter of morphosyntax. It is very common for researchers to search in POS-tagged corpora for particular word-level POS categories. It follows that users of a *morphologically* annotated corpus are likely to wish to search for particular categories of root morpheme: noun roots, verb roots, and so on. The difference between such a search and word-level POS queries is best illustrated by considering words based on the same root but with different POS tags. For instance, in English, a word-level POS query for nouns would retrieve *danger* but not *dangerous* (adjective) or *endanger* (verb); but a query for words containing a morpheme tagged with noun as its root POS would retrieve all three of these words, since the morpheme *danger* would have the same root POS annotation in all three. This justifies including root POS in morphological annotation: it is distinct from – and provides additional possibilities for analysis to – the word-level POS provided by morphosyntactic annotation.

Why, then, use only major POS categories and not fine-grained POS categories? Much fine-grained POS analysis is within the domain of morphosyntax, operating at word level, because fine distinctions such as "singular noun" versus "plural noun", or "present-tense verb" versus "past-tense verb" (to cite two distinctions common in POS tagsets for English), reflect features encoded not in the root but in affixes. For this MAS to use such fine-grained distinctions would thus be counterproductive, as the analysis could not then be clearly assigned to a specific morpheme – breaching principle 3, *to unambiguously link each morpheme to its annotations*.

Fine-grained features which *are* properties of root morphemes, such the distinction between common and proper nouns, tend not to be structural features, but rather features of (lexical) semantics. Encoding such features is not part of morphological annotation (even though it could well be useful for users). Thus, the analysis of root POS will not go beyond the major POS category. There are 12 root POS categories, in addition to a *Foreign* category (foreign words).

For many Indonesians, the term *foreign* refers to entities from outside Indonesian; thus, a language such as Javanese, which is spoken within Indonesia, is not referred to as a *foreign language*. I do not use the term *foreign* in this sense. In this MAS, *foreign* refers to *any non-Indonesian* word, whether from a language spoken outside Indonesia (e.g. English, French) or from a language spoken in Indonesia (e.g. Javanese, Sundanese).

Finally, although the category of *Article* exists, Indonesian articles are atypical compared to articles in most languages, because they occur only in limited circumstances; definiteness/indefiniteness is usually implicit.

POS	I-Tag	E-Tag	Examples
Noun	*NOM*	NOU	*nasi* 'rice', *jagung* 'corn', *London* 'London'
Pronoun	*PRO*	PRO	*aku* 'I' (personal), *kenapa* 'why' (interrogative), *sini* 'this place' (demonstrative)
Numeral	*NUM*	NUM	*satu* 'one' (cardinal), *pertama* 'first' (ordinal)
Classifier	*PGL*	CLA	*ekor* 'animal class', *orang* 'human class'
Verb	*VER*	VER	*pergi* 'go', *makan* 'eat', *lari* 'run'
Adjective	*AJE*	ADJ	*cantik* 'beautiful', *cepat* 'quick', *lama* 'long'
Adverb	*ADV*	ADV	*selalu* 'always', *jarang* 'seldom', *hanya* 'only'
Preposition	*PRE*	PRE	*di* 'at', *ke* 'to', *dari* 'from'
Conjunction	*KNJ*	CON	*dan* 'and', *atau* 'or', *ketika* 'when'
Interjection	*ITJ*	INT	*hai* 'hi', *aduh* 'ouch', *astaga* 'oh my god'
Article	*ART*	ART	*si* 'the (derogatory)', *sang* 'the (honorific)'
Particle	*PKL*	PAR	*kah, lah, pun* (all emphasis)
Foreign	*ASG*	FRG	*revenue* (English), *aqua* 'water' (Latin), *monggo* 'please' (Javanese)
Punctuation	*TDB*	PUNC	colon (:), question (?), exclamation (!)

Table 4.5. Major POS categories for roots

	I-tag	E-tag
nasi	*<nasi,AKR+**NOM**>*	<nasi,ROOT+**NOU**>
satu	*<satu,AKR+**NUM**>*	<satu,ROOT+**NUM**>
cantik	*<cantik,AKR+**AJE**>*	<cantik,ROOT+**ADJ**>

Table 4.6. Major POS categories within full analyses

4.2.3. Affixes

4.2.3.1. Formal analytic categories

The four categories of Indonesian affix are prefix, suffix, infix, and circumfix (see 2.1.3.4.2.1). Prefix and suffix are common formal morphological categories, and thus should obviously be included in this MAS. What about circumfix and infix? The importance of circumfixes in Indonesian has been argued in section 2.1.3.4.2.1, and they are thus included in this MAS. As for the infixes, Sneddon (2010:28-29) argues that they are no longer productive, and that therefore infixed words are often treated as monomorphemic words. While I agree with this for word-level analysis, I argue that for morpheme-level annotation, infixes are still worth analysing separately, as they function to mark functional categories (e.g. nominaliser, plural) which users are likely to wish to retrieve from annotated corpora.

It is tempting to encode *affix* as a super-category of these four, either organising the two categories hierarchically (following principle 9), e.g. AFX:PFX and AFX:SFX, or merging the concepts directly as one category, e.g. APFX and ASFX. But this would have little user benefit, since the category of affix is already retrievable using wildcards in an underspecified query pattern such as *FX. So no explicit super-categorisation of these four categories is incorporated into the MAS.

I-Tag	E-Tag	Category	example	In word
PFS	PFX	Prefix	*di-*	*di-bakar* PSV-burn 'be burnt'
SFS	SFX	Suffix	*-an*	*tembak-an* shoot-NOMZR 'shot'
IFS	IFX	Infix	*-er-*	*g-er-igi* PL-tooth 'teeth'
KFS	CFX	Circumfix	*ke—an*	*ke-raja-an* NOMZR-king-NOMZR 'kingdom'

Table 4.7. Formal categories for affixes

	I-tag	E-tag
di-	*<di,**PFS**+DER:VER+PSV>*	<di,**PFX**+DER:VER+PSV>
-an	*<an,**SFS**+DER:NOM>*	<an,**SFX**+DER:NOU>
-el-	*<el,**IFS**+DER:NOM>*	<el,**IFX**+DER:NOU>
ke—an	*<ke,**KFS+A**+DER:NOM>* *<an,**KFS+Z**+DER:NOM>*	<ke,**CFX+A**+DER:NOU> <an,**CFX+Z+**DER:NOU>

Table 4.8. Formal categories for affixes within full analyses

4.2.3.2. Functional analytic categories

4.2.3.3.1. Outcome POS

The first functional category relevant to affixes is *outcome POS*. This is an important factor for derivational processes. In glossing convention, labels such as nominaliser (NOMZR) or verbaliser (VBZR) include an indication of the POS category which a derivational morpheme produces when applied to some base; the annotation of an affix's outcome POS captures this information. An affix's outcome POS may be identical to that of the root it applies to, but does not have be . Likewise, an affix's outcome POS may be the same as the broad POS tag of the overall word (as is the case for examples *di-bakar* and *tembak-an* in Table 4.7, but does not have to be. Being able to search for outcome POS separately from both root POS and (in possible accompanying morphosyntactic annotation) word POS is therefore potentially beneficial to users.

For instance, verb *berkebutuhan* in example (4.30), annotated in (4.31), is composed of three morphemes: prefix *ber-,* circumfix *ke—an,* and root *butuh.* Of these three, the first two morphemes are annotated with outcome POS. Prefix *ber-* derives a verb from a noun base (i.e. *kebutuhan*). Circumfix *ke—an* derives a noun from a verb root (i.e. *butuh*). The POS that each derives, verb and noun respectively, is its outcome POS. Only the outcome POS of *ber-* is equivalent to the POS of the full word (verb). But since this MAS aims at morpheme-level annotation, morpheme outcome POS is the relevant information.

(4.30) *ber-ke-butuh-an*
VBZR-NOMZR-need-NOMZR
'have needs'

(4.31) *berkebutuhan*
<ber,beR,PFX+DER:VER>
<ke,CFX+A+DER:NOU>
<butuh,ROOT+VER>
<an,CFX+Z+DER:NOU>

Inflectional affixes should not be given outcome POS as the POS of the word they generate is the same as that of the base. But any derivational affix may potentially change the POS of its base, and therefore should have outcome POS tagging. I wish to argue that all affixes in this MAS must be given outcome POS because Indonesian is exclusively derivational. The argument is as follows.

In 2.1.3.2, I noted that the presence of any inflection in Indonesian is a matter of debate. While Alwi et al. (1998) do not *explicitly* adopt a stance on this issue, they consistently use the term *penurunan* 'derivation' for affixation processes. By contrast, Sneddon et al. (2010) suggest that two affixes only (active *meN-* and passive *di-*) are inflectional. In this, they seem to follow Prentice's (1987) argument, but Prentice in fact argues that *meN-* is ambiguous, and can be inflectional or derivational. Following principle 6, the view of the Indonesian reference grammars (Sneddon et al. 2010; Alwi et al. 1998) should be prioritised, but here they differ. While Sneddon et al. assert the existence of active and passive inflection, they give no account of the distinction between inflectional and derivational affixes beyond this, implying that this distinction is marginal. This being the case, I follow Alwi et al. and consider all affixation to be derivation for purposes of this MAS.

As previously observed, an affix's outcome POS may be but is not always the same as the POS of the word overall. Specifically, in the case of a word formed via multiple derivations, the last-added derivational morpheme's outcome POS determines the POS of the word. Other affixes' outcome POS have no effect on word-level POS. POS is a property of words, while outcome POS is a property of derivational affixes, so they are conceptually distinct even for words where they happen to coincide.

The tags for outcome POS begin with a super-category label DER (derivation), whose I and E tags are identical. Then follows the major POS which is the outcome of the affix annotated, such as noun (NOU), verb (VER), or adjective (ADJ). These are the same labels used for root POS (see 4.2.2.3). Thus, the two categories are organised hierarchically, the POS being dependent on the category of derivation. Following principle 9 on combining tags for dependent categories yields the tags DER:NOU, DER:ADJ, DER:ADV, DER:VER and DER:NUM (see Table 4.9 to Table 4.16).

I-tag	E-tag	Affix	Example
DER:NOM	DER:NOU	*pe-*	*pe-suruh* PAT-command 'person to be commanded'
		pel-	*pel-ajar* PAT-teach 'student'
		peN-(1,2)	*peny-(s)uruh* AGNR-command 'commander'
		peR-(2,3)	*pe-tani* AGNR-agriculture 'farmer'
		teR-(5)	*ter-sangka* PAT-suspect(verb) 'suspect'
		-an	*makan-an* eat-NOMZR 'food'
		-wan	*warta-wan* news-male 'male reporter'
		-wati	*warta-wati* news-female 'female reporter'
		-nya(1,2)	*buku-nya* book-DEF 'the book'
		ke—an(1)	*ke-jahat-an* NOMZR-evil(adj)-NOMZR 'crime'
		peN—an	*peny-(s)atu-an* NOMZR-one-NOMZR 'unification'
		peR—an	*per-satu-an* NOMZR-one-NOMZR 'union'
		-el-	*t-el-unjuk* <NOMZR>point 'index finger'
		-em-	*j-em-ari* <PL>finger 'fingers'
		-er-	*s-er-uling* <NOMZR>-flute 'flute'
		-er-	*g-er-igi* <PL>-tooth 'teeth'

Table 4.9. Noun derivation outcome POS

I-Tag	E-Tag	Affix	*Example*
DER:AJE	DER:ADJ	*se-(1)*	*se-mahal* EQU-expensive 'as expensive'
		se-(2)	*se-kantor* COLL-office 'whole office'
		ke—an(3)	*ke-merah-an* ADJZR-red 'Reddish'
		teR-(4)	*ter-cantik* SUP-beautiful The most beautiful

Table 4.10. Adjective derivation outcome POS

I-Tag	E-Tag	Affix	*Example*
DER:ADV		*nya-(3)*	*biasa-nya* usual-ADV 'usually'
		se—an	*se-hari-nya* ADV-day-ADV 'full-day'
		se—nya	*se-cepat-nya* ADV-quick-ADV 'as quickly as possible'

Table 4.11. Adverb derivation outcome POS

I-tag	E-tag	*Affix*	*Example*
DER:VER		*beR-(1)*	*ber-pistol* VBZR-gun 'have a gun'
		beR-(2)	*ber-cermin* RFLX-mirror 'look at oneself on a mirror'
		di-	*di-ambil* PSV-take 'be taken'
		meN-	*meng-ambil* ACV-ACV 'take'
		peR-	*per-kuat* CAUS-strong 'strengthen'
		teR-(1,2,3)	*ter-bawa* PSV-bring 'be brought by accident'
		-i(1)	*kepala-i* head-VBZR 'lead'
		-i(2)	*kirim-i* send-APPL 'send sth'
		-i(3)	*panas-i* hot-CAUS 'make hot'
		-i(4)	*ketok-i* knock-ITRV 'knock over and over'
		-i(5)	*pukul-i* punch-APPL.ITRV 'punch sth over and over with sth'
		-kan(1)	*lurus-kan* straight-CAUS 'straighten'
		-kan(2)	*kirim-kan* send-APPL 'send sth'
		-kan(3)	*periksa-kan* examine-CAUS 'have sth examined'

Table 4.12.Verb derivation outcome POS (part 1, prefixes and suffixes)

I-tag	E-tag	*Affix*	*Example*
DER:VER		*beR—an(1)*	*ber-salam-an* RECP-shake.hand-VBZR.RECP ‘shake hands one each other’
		beR—an(2)	*ber-pukul-an* RECP.ITRV-punch-RECP.ITRV ‘punch one another again and again’
		beR—an(3)	*ber-jatuh-an* VBZR.RAND-fall-VBZR.RAND ‘fall randomly’
		beR—kan	*ber-senjata-kan* POSS-weapon-POSS ‘have weapon’
		ke—an (2)	*ke-curi-an* PSV-steal-PSV ‘be robbed of’
		peR—i	*per-baik-i* VBZR-good-VBZR ‘repair’
		peR—kan	*per-malu-kan* VBZR-ashamed-VBZR ‘embarrass’
		-em-	*g-em-etar* <VBZR>-vibration ‘tremble’

Table 4.13.Verb derivation outcome POS (part 2, circumfixes and infixes)

I tag	E-tag	Affix	Sample
DER:NUM		*ke-*	*ke-dua* ORD/COLL-two ‘second/both’

Table 4.14. Numeral derivation outcome POS

	I-tag	E-tag
Adjective		
se-(1)	<se,PFX+**DER:AJE**+EKTF>	<se,PFX+**DER:ADJ**+EQTV>
se-(2)	<se,PFX+**DER:AJE**>	<se,PFX+**DER:ADJ**>
teR-(4)	<ter,teR,PFX+**DER:AJE**+SUPF>	<ter,teR,PFX+**DER:ADJ**+SUPV>
ke—an (3)	<ke,KFS+A+**DER:AJE**> <an,KFS+Z+**DER:AJE**>	<ke,CFX+A+**DER:ADJ**> <an,CFX+Z+**DER:ADJ**>
Adverb		
-nya (3)	<nya,SFX+**DER:ADV**>	<nya,SFX+**DER:ADV**>
se—an	<se,KFS+A+**DER:ADV**> <an,KFS+Z+**DER:ADV**>	<se,CFX+A+**DER:ADV**> <an,CFX+Z+**DER:ADV**>
se—nya	<se,KFS+A+**DER:ADV**> <nya,KFS+Z+**DER:ADV**>	<se,CFX+A+**DER:ADV**> <nya,CFX+Z+**DER:ADV**>
Noun		
pe-	<pe,PFX+**DER:NOM**>	<pe,PFX+**DER:NOU**>
pel-	<pel,PFX+**DER:NOM**>	<pel,PFX+**DER:NOU**>
peN-(1,2)	<pen,peN,PFX+**DER:NOM**>	<pen,peN,PFX+**DER:NOU**>
peR-(2&3)	<pe,peR,PFX+**DER:NOM**>	<pe,peR,PFX+**DER:NOU**>
teR-(5)	<ter,teR,PFX+**DER:NOM**>	<ter,teR,PFX+**DER:NOU**>
-an	<an,SFX+**DER:NOM**>	<an,SFX+**DER:NOU**>
-nya (1,2)	<nya,SFX+**DER:NOM**+DEF>	<nya,SFX+**DER:NOU**+DEF>
ke—an (1)	<ke,KFS+A+**DER:NOM**> <an,KFS+Z+**DER:NOM**>	<ke,CFX+A+**DER:NOU**> <an,CFX+Z+**DER:NOU**>
peN—an	<peN,KFS+A+**DER:NOM**> <an,KFS+Z+**DER:NOM**>	< pen,peN,CFX+A+**DER:NOU**> <an,CFX+Z+**DER:NOU**>
per—an	<per,peR,KFS+A+**DER:NOM**> <an,KFS+Z+**DER:NOM**>	<per,peR,CFX+A+**DER:NOU**> <an,CFX+Z+**DER:NOU**>
-el-(1)	<el,IFS+**DER:NOM**>	<el,IFX+**DER:NOU**>
-em-(1)	<em,IFS+**DER:NOM**>	<em,IFX+**DER:NOU**>
-er-(1)	<er,IFS+**DER:NOM+PLUR**>	<er,IFX+**DER:NOU+PLUR**>
-er-(2)	<er,IFS+**DER:NOM**>	<er,IFX+**DER:NOU**>

Table 4.15. Outcome POS categories within full analyses (part 1)

Numeral		
ke-	<ke,PFX+**DER:NUM**>	<ke,PFX+**DER:NUM**>
Verb		
beR-(1)	<ber,beR,PFX+**DER:VER**>	<ber,beR,PFX+**DER:VER**>
beR-(2)	<ber,beR,PFX+**DER:VER**+RFLX>	<ber,beR,PFX+**DER:VER**+RFLX>
di-	<di,PFX+**DER:VER**+PSV>	<di,PFX+**DER:VER**+PSV>
meN-	<men,meN,PFX+**DER:VER**+ACV>	<men,meN,PFX+**DER:VER**+ACV>
peR-(1)	<per,peR,PFX+**DER:VER**+CAUS>	<per,peR ,PFX+**DER:VER**+CAUS>
peR-(4)	<per,peR,PFX+**DER:VER**>	<per,peR,PFX+**DER:VER**>
teR-(1,2,3)	<ter,teR,PFX+**DER:VER**+PSV>	<ter,teR,PFX+**DER:VER**+PSV>
-el-(2)	<el,IFS+**DER:VER**>	<el,IFX+**DER:VER**>
-em-(2)	<em,IFS+**DER:VER**>	<em,IFX+**DER:VER**>
-i (1)	<i,SFX+**DER:VER**>	<i,SFX+**DER:VER**>
-i (2)	<i,SFX+**DER:VER**+APLI>	<i,SFX+**DER:VER**+APPL>
-i (3)	<i,SFX+**DER:VER**+CAUS>	<i,SFX+**DER:VER**+CAUS>
-i (4)	<i,SFX+**DER:VER**+ITRF>	<i,SFX+**DER:VER**+ITRV>
-i (5)	<i,SFX+**DER:VER**+APLI+ITRF>	<i,SFX+**DER:VER**+APPL+ITRV>
-kan (1)	<kan,SFX+**DER:VER**>	<kan,SFX+**DER:VER**>
-kan (2)	<kan,SFX+**DER:VER**+APLI>	<kan,SFX+**DER:VER**+APPL>
-kan (3)	<kan,SFX+**DER:VER**+CAUS>	<kan,SFX+**DER:VER**+CAUS>
beR—an (1)	<beR,KFS+A+**DER:VER**+RECP> <an,KFS+Z+**DER:VER**+RECP>	<ber,beR,CFX+A+**DER:VER**+RECP> <an,CFX+Z+**DER:VER**+RECP>
beR—an (2)	<ber,beR,KFS+A+**DER:VER**+RECP+ITRF> <an,KFS+Z+**DER:VER**+RECP+ITRF>	<ber,beR,CFX+A+**DER:VER**+RECP+ITRV> <an,CFX+Z+**DER:VER**+RECP+ITRV>
beR—an (3)	<ber,beR,KFS+A+**DER:VER**+RAND> <an,KFS+Z+**DER:VER**+RAND>	<ber,beR,CFX+A+**DER:VER**+RAND> <an,CFX+Z+**DER:VER**+RAND>
beR—kan	<ber,beR,KFS+A+**DER:VER**+POSS> <kan,KFS+Z+**DER:VER**+POSS>	<ber,beR,CFX+A+**DER:VER**+POSS> <kan,CFX+Z+**DER:VER**+POSS>
ke—an (2)	<ke,KFS+A+**DER:VER**> <an,KFS+Z+**DER:VER**>	<ke,CFX+A+**DER:VER**> <an,CFX+Z+**DER:VER**>
peR—i	<per,peR,KFS+A+**DER:VER**> <i,KFS+Z+**DER:VER**>	<per,peR,CFX+A+**DER:VER**> <i,CFX+Z+**DER:VER**>
peR—kan	<per,peR,KFS+A+**DER:VER**> <kan,KFS+Z+**DER:VER**>	<per,peR,CFX+A+**DER:VER**> <kan,CFX+Z+**DER:VER**>
-em-	<em,IFS+**DER:VER**>	<em,IFX+**DER:VER**>

Table 4.16. Outcome POS categories within full analyses (part 2)

4.2.3.3.2. Voice and other valency constructions

4.2.3.3.2.1. Active and passive voice

Active and passive voice can be briefly described in terms of the semantic and grammatical roles of a verb's arguments. In the active voice, the grammatical role of subject is occupied by an agent (or agent-like semantic role), whereas in the passive voice the grammatical role of subject is occupied by a patient (or patient-like semantic role). In many languages including English, the active voice is the basic form of the verb, and passives are constructed by modification of an active transitive verb (either morphologically or periphrastically), with the consequence that the object 'slot' of the active verb corresponds to the subject 'slot' of the passive verb.

The Indonesian active and passive differ from this pattern (see 2.1.3.4.2.2). In Indonesian, prefix *meN-* marks active voice, while prefixes *di-* and *teR-* and circumfix *ke—an* mark passive voice (note that I here discuss *teR-(1)* and *ke—an(2)* whose outcome POS is verb, not *teR-(5)* and *ke—an(1)* whose outcome POS is noun). Here, I diverge slightly from Sneddon et al. (2010: 17,116), who label only *di-* and *teR-* as passive, because, their analysis involves not only morphological marking, but also morphosyntactic class and syntactico-semantic context. Sneddon et al. firmly argue that *di-* is a pure morphological marker of the passive. By contrast, they argue, a verb marked by *teR-* can be transitive or intransitive, and the derivation can express one of three categories: *accidental*, *stative* and *abilitative*. They then argue that the passive reading applies only when the verb is transitive. Compare examples (4.32), with transitive root *tembak* 'shoot', and (4.33) with intransitive root *tidur* 'sleep'; the latter does not have a passive meaning.

(4.32)	*dia*	*di-tembak*	*atau*	*ter-tembak*	*polisi*
	3s	PSV-shoot	or	PSV.ACCIDENTAL-shoot	police

's/he was deliberately or accidentally shot by a cop'

(4.33)	*Latif*	*ter-tidur*	*di*	*kelas*
	PN	PSV.ACCIDENTAL-sleep	in	classroom

'Latif fell asleep in the classroom'

The categories of *stative* and *abilitative* are not linked to passive by Sneddon et al., but the examples they propose are written as passive voice constructions in the translations. Stative *teR-* verbs express the subject patient entity's ongoing state as a result of the verb's action, without specifying an agent; see example (4.34). This might be equivalent to an English agentless passive. Abilitative *teR-* adds modal meaning; see (4.35). Again, the passive construction is used in the translation.

(4.34)	*surat*	*itu*	*ter-tulis*	*dalam*	*bahasa*	*Inggris*
	letter	that	PSV.STAT-write	in	language	English

'That letter is written in English' (Sneddon et al. 2010:117)

(4.35)	*mobil*	*se-mahal*	*itu*	*tidak*	*ter-beli*	*oleh*	*saya*
	car	EQT-expensive	that	not	PSV.ABILITY-buy	by	1s

'I can't afford to buy a car as expensive as that' (Sneddon et al. 2010:121)

Sneddon et al. (2010:124) argue that circumfix *ke—an* marks an event as adversative, as shown by example (4.36) in comparison to example (4.37). Here, *Tomo* is an undergoer. Sneddon et al. use the same English passive "was stolen" in the translations for both for prefix *di-* (passive) and circumfix *ke—an* (adversative).

(4.36)	*mobil*	*Tomo*	*di-curi*
	car	PN	PSV-steal

'Tomo's car was stolen' (Sneddon et al. 2010:124)

(4.37)	*Tomo*	*ke-curi-an*	*mobil*
	PN	PSV.ADVRS-steal-PSV.ADVRS	car

'Tomo was robbed of his car' (Sneddon et al. 2010:124)

If we adhere to Sneddon et al.'s model of Indonesian voice, the categories are as follows: active, passive, accidental, accidental passive, abilitative, stative, and adversative. Passive could be made a super-category, with the subcategories of deliberate passive and accidental passive. But this would be odd, as accidental would then be both a subcategory of passive and an independent category.

The alternative to Sneddon's et al.'s approach is to distinguish two opposed categories, active and passive. In this approach, while the five distinct subtypes of passive – general passive *di-*, accidental passive *teR-*, stative passive *teR-*, abilitative passive *teR-* and adversative passive *ke—an* are acknowledged, they are not distinguished in the analysis. The reason for this approach is that determining the precise subtype function of *teR-* requires semantic context. So *teR-* is analysed in this MAS as marking one functional category only, passive.

Therefore, there are two categories: the affix that definitely marks active voice (*meN-*), and the affixes that definitely mark passive voice (*di-, teR-, ke—an*). I will not attempt to annotate the "zero" active implied by the absence of a passive marker, only the explicit marker *meN-*. This adheres to principle 4. The category of active is encoded as *AKF* or ACV. The category of passive is encoded as *PSF* or PSV.

I-Tag	E-tag	Affix	Word
AKF	ACV	*meN-*	*meng-ambil* ACV-take 'take'
PSF	PSV	*di-*	*di-ambil* PSV-take 'be taken'
		teR-	*ter-ambil* PSV-take 'be taken accidentally'
		ke—an	*ke-curi-an* PSV-steal-PSV 'is robbed of'

Table 4.17. Analytic categories for active and passive voices

	I-tag	E-tag
meng-	*<meng,meN,PFS+DER:VER+**AKF>***	<meng,meN,PFX+DER:VER+**ACV>**
di-	*<di,PFS+DER:VER+**PSF>***	<di,PFX+DER:VERB+**PSV>**
teR-	*<ter,teR,PFS+DER:VER+**PSF>***	<ter,teR,PFS+DER:VERB+**PSV>**
ke—an	*<ke,KFS+A+DER:VER+**PSF>*** *<ke,KFS+Z+DER:VER+**PSF>***	<ke,CFX+A+DER:VERB+**PSV>** <an,CFX+Z+DER:VERB+**PSV>**

Table 4.18. Analytic categories for active and passive voices within full analyses

4.2.3.3.2.2. Applicative and causative constructions

As it has been introduced in 2.1.3.4.2.2, there are some affixes in Indonesian which can be used to mark applicative and causative constructions. According to Payne (1997:186), an applicative construction promotes to core argument status an oblique with some specific semantic role such as recipient or instrument. So, for instance, in *John sent a gift to Mary*, Mary is an oblique, but in the equivalent dative shift construction (a type of applicative) *John sent Mary a gift*, Mary is promoted to the status of object, a core argument. Discussing causative constructions (Payne 1997:176) requires an understanding of two important terms, *causer* and *causee*. In a causative construction, an event is represented as being caused to happen by some entity other than its immediate agent participant. In a causative clause, the external entity responsible for the causation is called the *causer*, and is usually the subject. The immediate agent of the event, i.e. the subject of the equivalent non-causative clause is then the *causee*, and has a grammatical relation other than subject. So, for instance, in the sentence *I had him examine my eyes*, *I* is the causer and *him* is the causee, in contrast to non-causative *he examined my eyes*, where there is no causer and *he* is the agent/subject.

In Indonesian, the causative and applicative can be morphologically marked. As shown in Indonesian reference grammars (Alwi et al. 1998; Sneddon et al. 2010) and other works discussing causative constructions (such as Arka 1993), the term *causative* is defined as deriving a verb with the meaning 'make something [VERB/ADJECTIVE]' from the 'VERB/ADJECTIVE' base. Sneddon (2010:74) argues that the base of Indonesian morphological causatives can be verb, adjective, or noun. Here are some examples they proposed (Senddon et al. 2010:74-78).

(4.38)	*Siti*	*bangun*
	PN	wake

'Siti woke'

(4.39)	*Ibu*	*mem-bangun-kan*	*Siti*
	mother	ACV-wake-CAUS	PN

'mother woke Siti up'

(4.40)	*kamar*	*ini*	*bersih*
	room	DEM	clean

'this room is clean'

(4.41)	*Siti*	*sudah*	*mem-bersih-kan*	*kamar*	*ini*
	PN	PF	ACV-clean-CAUS	room	DEM

‘Siti has cleaned the room/Siti has made this room to be clean’

(4.42)	*pesawat*	*sudah*	*men-darat*
	Plane	PF	ACV-land

‘the plane has landed’

(4.43)	*Pilot*	*sudah*	*men-darat-kan*	*pesawat*
	pilot	PF	ACV-land-CAUS	plane

‘the pilot has landed the plane/the pilot has put the plane on the land’

This is slightly different from the standard understanding of causatives, as described in Payne (1997), as a valency-changing construction on verb bases. In this MAS, I adopt the extended definition of *causative* given in the Indonesian reference grammars.

Three verbal affixes indicate applicative or causative, namely *peR-*, *-kan* and *-i.* Indonesian scholars agree that *peR-* is causative and not applicative. *-kan* and *-i* are less straightforward. Sneddon et al. (2010:74-98) argue for a tendency for *-kan* to mark causative and *-i* applicative, whereas Arka (1993: 209) argues that both *-kan* and *-i* can mark both applicative and causative.

Furthermore, both *-kan* and *-i* exhibit idiosyncratic behaviour in terms of compatibility with specific bases, as Table 4.19 shows. *Jatuh* ‘fall’ combines with both, forming a causative with *-kan* and an applicative with *-i*; *lupa* ‘forget’ combines with *-kan* to form an applicative, but not with *-i*. Moreover, *-i* cannot co-occur with a base ending in *-i*, so **beri-i*, for instance, is not a word. Overall, *-kan* is more productive than *-i.*

Root	Root with *-kan*	Root with *-i*
jatuh ‘fall’	*jatuh-kan* ‘drop’ (causative)	*jatuh-i* drop (something) at (something)’ (applicative)
lupa ‘forget (intrans.)’	*lupa-kan* ‘forget (trans.)’ (applicative)	**lupa-i*
beri ‘give’	*beri-kan* ‘give (sth) to someone’ (applicative)	**beri-i*

Table 4.19. Productivity of *-kan* versus *-i*

Many authors, including Alwi et al. (1998), Sneddon (1996), and Sneddon et al. (2010), do not use the term *applicative.* Instead, they label this construction with the semantic role of the nominal promoted to become the applicative object. Thus, Sneddon et al. (2010) establish the categories of *instrumental*, *benefactive, locative,* and *recipient* illustrated in examples (4.44) to (4.51). This stands in contrast to studies focusing on voice such as Arka (1993), Arka (2009) and Shiohara (2012), which do use the term applicative. Note that unlike causative, no extended definition is proposed for applicative constructions in Indonesian.

While many examples show that suffix *-i* tends to mark applicative constructions in Indonesian, Shiohara adds that in some cases, the applicative suffix is optional. For instance, in the recipient applicative example in (4.51), it is possible to delete *-i* and still obtain an applicative reading for the sentence.

(4.44)	*dia*	*mem-(p)ukul*	*anjing*	*dengan*	*tongkat.*	
	3s	ACV-hit	dog	with	cane	

‘he hit the dog with a stick’

(4.45)	*dia*	*mem-(p)ukul-kan*	*tongkat*	*pada*	*anjing.*	Instrumental
	3s	ACV-hit-APPL	cane	to	dog	

‘He used the stick to hit the dog with. (Sneddon 1996: 78)

(4.46)	*pe-layan*	*meng-ambil*	*se=gelas*	*air*
	NOMZR-serve	ACV-take	one=glass	water

‘The waiter took a glass of water.’

(4.47)	*pe-layan*	*meng-ambil-kan*	*tamu*	*se=gelas*	*air*	Benefactive
	NOMZR-serve	ACV-take-APPL	guest	one=glass	water	

‘The waiter took the guest a glass of water.’ (Sneddon 1996: 80)

(4.48)	*dia*	*men-(t)anam*	*padi*	*di*	*sawah=nya*
	3s	ACV-plant	rice	at	ricefield=3

‘He planted rice in his field.’

(4.49)	*dia*	*men-(t)anam-i*	*sawah=nya*	*dengan*	*padi*	Locative
	3s	ACV-plant-APPL	ricefield=3	with	rice	

‘He planted his field with rice.’ (Sneddon 1996: 91)

(4.50)	*Ayah*	*meng-(k)irim*	*uang*	*kepada*	*saya.*
	Father	ACV-send	money	to	1s

‘Father sent me money.’

(4.51)	*Ayah*	*meng-(k)irim(-i)*	*saya*	*uang.*	Recipient
	Father	ACV-send(-APPL)	1s	money	

‘Father sent me money.’ (Sneddon 1996: 90)

Sneddon et al. seem to classify causatives on the basis of the POS of the root. When the root is a verb, the semantic roles of causer and causee seem to be relevant in their explanation of causatives like those in (4.53). When something other than a verb is causativised, then what is important is that the causer imposes the state coded by the non-verb root onto the object, as exemplified in (4.55). This example is from Arka (1996:2), as Sneddon et al. do not give a causative example with *–i*.

(4.52)	*Wanita itu*	*men-cuci*	*pakai-an*	*saya*	
	Female that	ACV-wash	wear-NOMZR	1s	
	'the woman washed my clothes'				

(4.53)	*Saya*	*men-cuci-kan*	*pakai-an*	*pada*	*wanita itu.*
	1s	ACV-wash-CAUS	wear-NOMZR	to	female that
	'I have that woman wash my clothes			(Sneddon et al. 2010:79)	

(4.54)	*lantai*	*itu*	*basah*
	floor	that	wet
	'that floor is wet'		

(4.55)	*Dia*	*mem-basah-i*	*lantai*	
	3s	ACV-wet-CAUS	floor	
	'she made the floor wet'			(Arka 1996:2)

peR- has causative reading (it does not mark applicative). The distribution of causative *peR-* is also more restricted; it appears on adjectival roots. *peR-* also attaches, irregularly, to certain verb roots, e.g. *per-buat* 'do', from *buat* 'make'; these do not have causative meaning. In this respect, *peR-* stands in contrast to *-kan* and *-i*.

(4.56)	*Negara-negara*	*Asia*	*men-coba*	*mem-**per**-canggih*
	Country-country	Asia	ACV-try	ACV-**CAUS**-sophisticated
	ke-mampu-an		*militer=nya.*	
	NOMZR-able-NOMZR		military=3p/s	
	'The Asian countries are trying to make their military capability (more) sophisticated.'		(Sneddon et al. 2010:104)	

As *-kan* and *-i* are polysemous, how can we disambiguate them? To identify an applicative, we need to verify whether or not the verb it forms has an applicativised object, which requires comparison of the object's semantic role, and that of any obliques or other arguments, to those expected by the verb base. To identify a causative, we likewise need to know the arguments' semantic roles, in order to know whether a causer and causee are both present.

Disambiguating these categories with purely morphological cues is not completely possible but is partially doable. I would argue that when *-kan* and *-i* are used with adjective or noun roots, these suffixes tend to mark causative, and not applicative constructions. The second argument is related to verbs with verbal roots and can be described as follows. Suffixes *-kan* and *-i* (in verbs with transitive verbal roots) are likely to mark applicative. When the verbal roots are intransitive, *-kan* tends to mark causative, while *-i* tends to mark applicative constructions.

To test this hypothesis, from the corpus that I have collected, I manually analysed 200 randomly collected sentences whose verbs ending in *-kan* and *-i*. These sentences were divided into two sets, each containing 100 sentences. The roots of the verbs of the sentences in the first set were adjective and noun roots, while the roots of verbs in the second set were verb roots. I then contextually examined these sentences to identify which ones are causative and which ones are applicatives. The results were cross-examined with my above proposal.

My proposed rule of thumb was 95% accurate for verbs with adjective and nooun roots (first set), and 71% accurate for verbs with verbal roots (second set), in this sample. These results suggest that this approach is promising, albeit without allowing an exact estimation of effectiveness. Since applicative and causative *can* be distinguished morphologically to an extent, the distinction *should* be accounted for in this MAS.

I opt not to include Sneddon et al.'s categories of benefactive, locative, instrumental, and recipient as subcategories of applicative in this MAS. While it could be useful, making the distinction requires syntactic/semantic analysis, which is beyond morphology.

There are two alternatives to Sneddon et al.'s division into four types of applicative plus causative. The first is to treat all the applicative and causative functions as one category, that is the applico-causative, as Malihah & Hardie (2014) do in work on Javanese. The advantage of using a single applico-causative category would be that the system later does not need to disambiguate these categories. The second alternative is to maintain causative and applicative as separate categories, without the four finer applicative categories. Although more natural (categories are kept separate), it requires the implementation to perform disambiguation.

I adopt this latter alternative, annotating two categories, applicative and causative, Although using a joint applico-causative category might simplify the implementation, the investigation reported above suggests that the applicative and causative *can* be distinguished morphologically in many contexts. This stands in contrast to the three different types of passive *teR-*, whose disambiguation requires context beyond morphology. The tags for causative and applicative are *KSF*/CAUS and *APLI*/APPL.

I-Tag	E-tag	Affix	Examples
KAUS	CAUS	*-kan*	*datang-kan* come-CAUS 'make someone come'
		-i	*panas-i* hot-CAUS 'heat (i.e.make sth hot)'
		peR-	*per-cantik* CAUS-beautiful 'beautify'
APLI	APPL	*-i*	*datang-i* come-APPL 'come at'
		-kan	*beli-kan* buy-APPL 'buy (someone) sth'

Table 4.20. Analytic categories for causative and applicative within full analyses

	I-tag	E-tag
-kan	*kan,SFS+DER:VER+**KAUS***	kan,SFX+DER:VER+**CAUS**
-kan	*kan,SFS+DER:VER+**APPL***	kan,SFX+DER:VER+**APPL**
-i	*i,SFS+DER:VER+**APPL***	i,SFX+DER:VER+**APPL**
-i	*i,SFS+DER:VER+**KAUS***	i,SFX+DER:VER+**CAUS**
peR-	*per,peR,SFS+DER:VER+**KAUS***	per,peR,SFX+DER:VER+**CAUS**

Table 4.21. Analytic categories for causative and applicative within full analyses

4.2.3.3.2.3. Reflexive and Reciprocal

In this MAS, reciprocal *beR—an* and reflexive *beR-* are each assigned a distinct voice category. These unproductive categories are worth including because they are criteria by which a user might want to search a corpus.

Reflexive voice indicates that the patient is the same entity as the agent, expressed as the subject. While English marks the reflexive with special '-self' pronouns, as in *she pushed* *herself*, in Indonesian, *beR-* can mark reflexive voice. Reflexive *beR-* is unproductive, limited to a handful of bases; Kridalaksana (2007:55) gives such examples as *ber-cukur* 'shave oneself', *ber-dandan* 'decorate oneself', and *ber-cermin* 'look at oneself in the mirror'. There is also a periphrastic reflexive which uses a special reflexive pronoun like those in English. However, when a root can take both the morphological and the periphrastic reflexive, usually the morphological construction is used. Compare (4.57) and (4.58).

(4.57)	*dia*	*sedang*	*ber-cukur*	
	3s	PROG	RFLX-shave	
	'he is shaving himself'			(morphological reflexive with *beR-*)

(4.58)	*dia*	*sedang*	*men-cukur*	*diri=nya*	
	3s	PROG	ACV-shave	self=3s	
	'he is shaving himself'				(periphrastic reflexive)

Reciprocal voice indicates that multiple people or entities do the same thing to each other, or that multiple people or entities stand in the same relation to each other. Thus, the subject of the verb is again both agent and patient. Reciprocal voice in Indonesian can be marked by circumfix *beR—an*. In spoken colloquial Indonesian, *–an* suffices to express the reciprocal. But since this project focuses on written Indonesian, this spoken feature is not considered further.

Reciprocal *beR—an* is unproductive and is found exclusively with a specific set of transitive verbal and nominal bases. So, in examples (4.59) to (4.61), adapted from Sneddon et al. (2010:111), *beR—an* co-occurs with *tabrak* 'hit', but new forms of this kind cannot be devised, e.g. with *beri* 'give' (**ber-beri-an*) or *makan* 'eat' (**ber-makan-an*). Some specific intransitive bases can take *beR—an*, for instance *ber-jatuh-an* 'fall randomly', but a reciprocal reading is not possible; this pattern is, again, unproductive. Example (4.62) illustrates *ber—an* with a nominal base *musuh* 'enemy'.

(4.59)	*Kapal*	*tangki*	*dan*	*kapal*	*barang*	*saling*	*tabrak*
	ship	tank	and	ship	thing	each	collide
	'The tanker and the cargo ship collided with each other'						

(4.60)	*Kapal*	*tangki*	*dan*	*kapal*	*barang*	*ber-tabrak-an*
	ship	tank	and	ship	thing	RECP-collide-RECP
	'The tanker and the cargo ship collided (with each other)'					

(4.61)	*Kapal*	*tangki*	*ber-tabrak-an*	*dengan*	*kapal*	*barang.*
	ship	tank	RECP-collide-RECP	with	ship	thing
	'The tanker collided (reciprocally) with the cargo ship'					

(4.62)	*mereka*	*bermusuhan*
	3p	RECP-enemy-RECP
	'they make enemies of each other'	

The agent-patient argument of a reciprocal can be expressed in two ways: as a single subject noun phrase as in the periphrastic reciprocal in (4.59) and the morphological reciprocal in (4.61), with agent and patient roles fused, or with one of the entities as the subject and the other an oblique nominal, as in (4.62). Another way to obtain reciprocal reading is by using reduplication, which I will explain in 4.2.4.2.

I-Tag	E-tag	Examples
RESP	RECP	*ber-peluk-an* RECP-hug-RECP 'hug each other'
REFL	RFLX	*ber-cukur* RFLX-shave(v.tr) 'shave oneself'

Table 4.22. Analytic categories for reciprocal and reflexive voices

	I-tag	E-tag
ber—an	*<ber,beR,KFS+A+DER:VER+**RESP>*** *<an,KFS+Z+DER:VER+**RESP**>*	<ber,beR,CFX+A+DER:VER+**RECP>** <an,CFX+Z+DER:VER+**RECP**>
ber-	*<ber,beR,PFS+DER:VER+**REFL>***	<ber,beR,PFX+DER:VER+**RFLX>**

Table 4.23. Analytic categories for reciprocals and reflexives within full analyses

4.2.3.3.3. Adjective degree

There are two adjective degree prefixes in Indonesian: superlative prefix *teR-*, equivalent to English *-est* or *most [adjective]*; and equative prefix *se-*, equivalent to the English construction *as [adjective] as* . Both *teR-* and *se-* have other functions, e.g. passive *teR-* and collective *se-*, but the distribution is distinct for the different functions. Superlative *teR-* only co-occurs with an adjective base, and passive *teR-* only with a verb base, so disambiguation is relatively easy. Likewise, collective *se-* precedes noun roots, while equative *se-* precedes adjective roots. Tags for adjective degree prefixes are presented in Table 4.24 and Table 4.25.

I-Tag	E-tag	Affix	Examples
SUPF	SUPV	*teR-*	*ter-cantik* SUPV-beautiful 'most beautiful'
EKTF	EQTV	*se-*	*se-cantik* EQTV-beautiful 'as beautiful as'

Table 4.24. Analytic categories for adjective degree

	I-tag	E-tag
ter-	*<ter,teR,PFS+DER:AJE+**SPF**>*	<ter,teR,PFX+DER:ADJ+**SUPV**>
se-	*<se,PFS+DER:AJE+**EKF**>*	<se,PFX+DER:ADJ+**EQTV**>

Table 4.25. Analytic categories for adjective degree within full analyses

4.2.3.3.4. Other functional categories

Some affix functions described in 2.1.3.4.2.2 are not included in the MAS, for various reasons. This may be due to the application of the principles in 4.1.2. For instance, nominaliser prefixes can be further classified as agentive, instrumental, and so on; but doing so requires analysis beyond morphology (semantics, syntax, or both). Therefore, only the most generic function is represented, which for a nominaliser prefix is the outcome POS of noun (DER:NOU).

In other cases, categories are omitted because a similar function is already covered by another reading. For instance, *se* can be described as a proclitic numeral or collective prefix. The word *sekampung* permits either reading: *se-kampung* 'the whole village' (collective prefix) or *se=kampung* 'one village'. The latter is the clitic version of *satu kampung* 'one village', but in fact *satu kampung* can have either reading: 'one village/the whole village'. Thus, the numeral clitic can be considered as also covering the meaning of the collective via polysemy, just as the full form covers both meanings. Thus, the MAS does not include a tag for the collective function of *se-/se=*.

4.2.3.3.4.1. Definite marker

Definiteness is not usually marked in Indonesian, but there do exist three definite markers: articles *si* and *sang*, and suffix *-nya.* Each receives an additional tag for the definite property (+DEF) in addition to its tagging as root or suffix. The last of these is the only bound definite marker in Indonesian. In some morphological contexts, e.g., suffixed to adverb *semua*, *-nya* can only be a definite suffix. However, in other cases, distinguishing definite *-nya* from pronominal enclitic *=nya* requires context beyond morphology; handling of this is discussed in 4.2.6.

When *nya* is a non-clitic morpheme, i.e. a suffix, its orthographic and citation form are identical. Therefore, following principle 4 (see 4.1.4), only one form is presented in the analysis, as in Table 4.26. Note that the +DEF for definite articles *si* and *sang* in 4.2.2.1 are given after the root's POS label, that is ROOT+ART+DEF.

I-Tag	E-tag	Affix	Word
DEF		*-nya*	*buku-nya* book-DEF 'the book'

Table 4.26. Analytic category for definite suffix

	I-tag	E-tag
-nya	*<nya,SFS+DER:NOM+**DEF**>*	<nya,SFX+DER:NOU+**DEF**>

Table 4.27. Analytic category for definite suffix within full analysis

4.2.3.3.4.2. Iterative

Iterative is the usual term for the aspect that expresses an action being performed repeatedly. Sneddon et al. (2010:98), however, use the term *Repetitive* instead of *Iterative* for this phenomenon in Indonesian. The affix that marks this action is *–i*, already discussed as applicative or causative marker. An analysis of this affix as iterative does not take the place of an analysis of causative/applicative, but rather adds to it, in cases such as example (4.65), in which *mem-(p)ukul-i* 'kiss iteratively' is an applicative with an additional iterative analysis. This meaning is only present when the suffix follows a transitive verb base.

(4.63)	*dia*	*men-(p)ukul*	*anjing=nya*	
	3s	ACV-hit	dog=3s	
	'He hit his dog'			

(4.64)	*dia*	*men-(p)ukul-i*	*anjing=nya*	
	3s	ACV-hit-ITRV	dog=3s	
	'He hit his dog over and over'			

(4.65)	*Dia*	*men-(p)ukul-i*	*anjing=nya*	*dengan*	*tongkat*
	3s	ACV-hit-APPL.ITRV	dog=3.poss	with	cane
	'He hit his dog over and over with a cane'				

The restriction to transitive verbal bases can be a useful cue for disambiguation. As (4.66) and (4.67) show, suffixing *–i* to adjectival base *sakit* 'sick' and intransitive base *jatuh* 'fall' does not give the iterative reading.

(4.66)	*Dia*	*meny-(s)akit-i*	*pacar=nya*	*dengan*	*kata-kata=nya*
	3s	ACV-sick-CAUS	girlfriend=3s	with	word-word=3s
	'He sickened his girlfriend with his words'				

(4.67)	*Dia*	*men-jatuh-i*	*ter-dakwa*	*dengan*	*hukum-an*	*mati*
	3s	ACV-fall-APPL	NOMZR.PAT.charge	with	punish-NOMZR	die
	'He punished the defendant (lit. person being charged) with a death sentence'					

Iterative aspect is indicated by tags *ITRV* / ITRV, applied to applicative/causative *-i* only after a transitive verb base, as illustrated by its absence in (4.68) (intransitive base) and presence in (4.69) (transitive base).

(4.68) *men-jatuh-i*
<men,meN,PFX+DER:VER+ACV>
<jatuh,ROOT+VER>
<i,SFX+DER:VER+APPL>

(4.69) *mem-(p)ukul-i*
<mem,meN,PFX+DER:VER+ACV>
<ukul,pukul,ROOT+VER>
<i,SFX+DER:VER+APPL+**ITRV**>

I-Tag	E-tag	Affix	Example
ITRF	ITRV	*-i*	*pukul-i* hit-APPL.ITRV 'hit over and over'

Table 4.28. Analytic category for iterative aspect

	I-tag	E-tag
-i	*<i,SFS+DER:VER+APLI+**ITRF**>*	<i,SFX+DER:VER+APPL+ITRV>

Table 4.29. Analytic category for iterative aspect within a full analysis

4.2.3.3.4.3. Random action

The meaning of diversely-directed activity, or *random action* (Sneddon et al. 2010:113), is conveyed by *beR—an*, which as noted in 4.2.3.3.2.3 usually marks reciprocal construction. However, *beR—an* with this random-action meaning instead of reciprocal meaning has low productivity, occurring only with a specific set of intransitive bases (listed by Sneddon et al. 2010:119), such as *jatuh* 'fall' (see Table 4.30). When its function is random action, *beR—an* is tagged *ACAK*/RAND.

I-Tag	E-tag	Affix	Word
ACAK	RAND	*beR—an*	*ber-jatuh-an* RAND-fall-RAND 'fall randomly'

Table 4.30. Analytic category for random action

	I-tag	E-tag
beR—an	*<ber,beR,KFS+A+DER:VER+**ACAK**>* *<an,KFS+Z+DER:VER+ACAK>*	<ber,beR,CFX+A+DER:VER+RAND>` `<an,CFX+A+DER:VER+RAND>

Table 4.31. Analytic category for random action within full analyses

4.2.4. Reduplication

4.2.4.1. Placement of functional analysis

Each reduplication includes two units: the *source* and *copy* (see 2.1.3.4.3). The source unit is annotated with the analytic categories of the root from which the reduplication is built (following normal root analysis). The copy is annotated with the analytic categories for the reduplication itself, as shown in (4.70) and (4.71). For discussion of how reduplication is tokenised, see section 4.2.1.3.

(4.70)	*sayur-mayur*	imitative reduplication
	<sayur,ROOT+NOU>	
	<mayur,sayur,RED:IMTV+DER:NOU+PLUR>	
(4.71)	*buku-buku*	
	<buku,ROOT+NOU+PLUR>	source
	<buku,RED:FULL+DER:NOU+PLUR>	copy

In imitative and partial reduplication, determining which unit is the source and which the copy is not difficult, as the orthographic form of the copy is obviously distinct from that of the source (*mayur* in example (4.70), not *sayur*). In full reduplication, the source and copy are identical. In this case, I opted to treat the second unit as the copy, as in (4.71), where the analytic category for reduplication, RED, is assigned to the second *buku* not the first.

What if the reduplication applies to a full polymorphemic word, such as compound *es krim* 'ice cream' composed of *es* 'ice' and *krim* 'cream'? When this two-root word is fully reduplicated, the copy mirrors the full word, giving the impression of two words and two duplicated morphemes (*es krim-es krim* 'ice creams'). Even though the reduplication affects a whole word, not a morpheme, the duplicated components of the base are annotated separately in full; the tagging of this example, presented in section 4.2.1.3, is repeated here as (4.72).

(4.72)	<es,ROOT+NOU>	source
	<krim,ROOT+ADJ>	source
	<es,RED:FULL+DER:NOU+PLUR>	copy
	<krim,RED:FULL+DER:NOU+PLUR >	copy

Both units in the copy (the second instance of *es krim*) are annotated with the analytic label for the reduplication morpheme, RED, while all units in the source are annotated according to their actual category, in this case as noun roots. While there is actually only one abstract reduplication morpheme here (the process of full-word reduplication applied to the overall word *es krim*), two reduplication morphemes are presented in the system of analysis defined by this MAS (one reduplication morpheme, i.e. copy, for each root). This is what we can call a *useful fiction*. It is indeed incorrect, but is useful in terms of keeping the overall system clear and simple, and matching up morphemes to analyses. For reasons explained in 4 2 1 3, attempting to present the reduplication as one rather than two morphemes would lead to a much more confused layout.

4.2.4.2. Formal and functional analytic categories for reduplication

All the analytic categories for reduplication begin with the formal morphological category label RED (reduplication). Full, partial, and imitative are dependent subcategories of RED, with labels FULL, PART and IMTV, respectively. After this formal tag are tags for the outcome POS of the reduplication, and then other functional tags (if any). For example, in (4.73), *buku-buku* is a full reduplication, whose outcome POS is noun, with plural function. Larasati et al.'s (2011) MAS (see 3.5.2) *also* applies the category of plural to verbs, but this would seem to be inaccurate (the meanings in question are in fact reciprocal/iterative, as discussed below). In this MAS, the analytic category plural is applied *exclusively* to the copy unit of a noun reduplication.

(4.73)	<buku,ROOT+NOU>	source
	<buku,RED:FULL+DER:NOU+**PLUR**>	copy

The two other functional categories marked by reduplication are reciprocal (RECP) and iterative (ITRV) (see 4.2.3.3.2.3 and 4.2.3.3.4.2). The reciprocal reading occurs only when active marker *-meN-* occurs between the source and copy of a verb root, as in (4.74). In this case, I opt decide to incorporate this category into the analysis of the copy, rather than the prefix *meN-* or the source. This is because the reciprocal reading is only valid in reduplication, particularly this pattern, and nowhere else. For instance, in (4.75) the verb root is fully reduplicated but there is no *meN-* between the source and copy, and thus this reduplication must not be analysed as reciprocal; rather, the correct reading is iterative[34].

(4.74) *pukul-mem-(p)ukul*
<pukul,ROOT+NOU+VER >
<mem,meN,PFX+DER:VER+ACV >
<ukul,pukul,RED:FULL+DER:VER+ITRV+**RECP**>

(4.75) *pukul-pukul*
<pukul,ROOT+NOU+VER >
<pukul,RED:FULL+DER:VER+**ITRV**>

The complete analytic categories for reduplication are given in Table 4.32 to Table 4.34.

I-Tag	E-tag	Reduplication type	Examples
RED:PNH	RED:FULL	Full	*buku-buku* book-RED.FULL 'books'
RED:PARS	RED:PART	Partial	*te-tamu* RED.PART-guest 'guests'
RED:IMTF	RED:IMTV	Imitative	*sayur-mayur* vegetable-RED.IMTV 'vegetables'

Table 4.32. Formal analytic categories of reduplication

I-Tag	E-tag	Examples
PLUR		*buku-buku* book-RED.FULL.PLUR 'books'
ITRF	ITRV	*pukul-pukul* hit-RED.FULL.ITRV 'neighbour'
RPKL	RECP	*pukul-mem-(p)ukul* hit-RED.FULL.ITRV.RECP 'hot one and each other'

Table 4.33. Functional analytic categories of reduplication

[34] Reciprocal and iterative readings can also be obtained from affixes instead of reduplication; see 3.5.1.1.4.2.3.3.2.3 and 4.2.3.3.4.2.

Reduplication	I-tag	E-tag
Full & plural	*<buku,RED:PNH+DER:NOM+PLUR>*	<buku,ROOT+NOU> <buku,**RED:FULL**+DER:NOU+**PLUR**>
Full & iterative	*<pukul,ROOT+NOM+VER>* *<pukul,RED:PNH+DER:VER+ITRF>*	<pukul,ROOT+NOU+VER> <pukul,**RED:FULL**+DER:VER+**ITRV**>
Full, iterative, and reciprocal	*<pukul,ROOT+NOM+VER>* *<mem,meN,PFS+DER:VER+AKF>* *<ukul,pukul,RED:FULL+DER:VER+ ITRV+RECP>*	<pukul,ROOT+NOU+VER> <mem,meN,PFX+DER:VER+ACV> <ukul,pukul,**RED:FULL**+DER:VER+**IT RV+RECP**>
Partial	*<te,tangga,RED:PNH+DER:NOU>*	<te,tangga,**RED:PART**+DER:NOU>
Imitative	*<mayur,sayur,RED:IMTF+ DER:NOU+PLUR>*	<mayur,sayur,**RED:IMTV**+ DER:NOU+PLUR>

Table 4.34. Analytic categories for reduplication within full analyses

4.2.5. Compounds

Compounds are analysed as a combination of roots (e.g. *kacamata* 'pair of eyeglasses' from *kaca* 'glass' and *mata* 'eye'), tokenised as two morphemes, each receiving an analysis. Compounds may also occur in combination with affixes, as in *per-tanggung-jawab-an* 'implementation of responsibility', in which the compounded *tanggung* 'carry' and *jawab* 'answer' are circumfixed by *peR—an*. Likewise, a compound can be reduplicated as in *kacamata-kacamata* 'pairs of eyeglasses', whose annotation is given in Table 4.35.

Kacamata-kacamata actually contains only one instance of reduplication (for the overall word *kacamata*), a circumstance already discussed in 4.2.4.1. There is, thus, one abstract reduplication morpheme, but in the analysis, two tokens are present, one for each root affected by the word-level reduplication. The four morphs thus collectively represent only three morphemes. As noted in 4.2.4.1, presenting the word as ha7ving four morphemes is a kind of *useful fiction*: a theoretically incorrect analysis which is useful in terms of keeping the overall system clear and simple and matching up morphemes to annotations.

Compound	Compound + affixation	Compound + reduplication
kaca-mata	*per-tanggung-jawab-an*	*kacamata-kacamata*
<kaca,ROOT+NOU> <mata,ROOT+NOU>	<per,peR,CFX+A+DER:NOU> <tanggung,ROOT+VER> <jawab,ROOT+VER> <an,CFX+Z+DER:NOU>	<kaca,ROOT+NOU> <mata,ROOT+NOU> <kaca,RED:FULL+DER:NOU+PLUR> <mata, RED:FULL+DER:NOU+PLUR>

Table 4.35. Compounds with and without other morphological processes

4.2.6. The NYA tag

The morpheme *nya* can be highly ambiguous in terms of functions and forms, and its disambiguation may require context beyond morphology. It is a homonym of 1) the pronominal enclitic (object of host verb or possessor of host noun); 2) the definite suffix, and 3) an adverbialiser suffix (see Table 2.7 in 2.1.3.4.2.2). Given the differences between clitics and derivational suffixes, for instance the latter having outcome POS annotation, these require distinct treatment.

In some cases, disambiguation can be implemented using orthographic and lexical cues. When *nya* is an anaphor for *tuhan* 'God', the N is always written in uppercase, thus: *=Nya*. Capitalised this way, it *must* be a case of the clitic, not the suffix. The adverbialiser *-nya* only occurs with a limited set of adjectives (e.g. *biasa* 'usual', *lazim* 'common'*, umum* 'general') and can be ruled out as the correct reading of *nya* in other contexts.

Otherwise, the suffix and enclitic readings are quite challenging to disambiguate, but there are a minority of cases where this can be achieved. First, there is a limited set of roots (e.g. *kata* 'word', *jawab* 'answer') with which *nya* is always the pronominal enclitic; see example (4.76). Second, if the word is marked by certain affixes (e.g. *meN-*, *peR—kan*), and the root is a transitive verb, *nya* must also be the pronominal enclitic: see example (4.77) . These affixes indicate that the host word is a transitive verb, and thus, the only possible interpretation of *nya* is as a pronominal enclitic for the verb's object, not a definite suffix.

(4.76) *kata=nya*
word=3s
'his/her word'

(4.77) *men-cari=nya*
ACV-search=3s
'search him/her'

In other cases, disambiguation requires syntactic or even pragmatic context. Attached to *buku* 'book', as in (4.78) and (4.79), *nya* can have either reading. For example, *mana bukunya?* can be translated as either 'where is his/her book?' or 'where is the book?', and only the speaker and hearer's shared situational knowledge determines which.

(4.78) *buku-nya*
book-DEF
'the book'

(4.79) *buku=nya*
book-3s
'his/her book'

Of these two means of disambiguation, one is within the scope of this thesis (orthographical, lexical, and morphological context), and one falls beyond the scope of this thesis (syntactic and pragmatic context). Looking forward to the implementation, we may say that the system should always try its best to disambiguate among the readings, annotated as explained earlier: enclitic <nya,dia,ROOT+ECLT+PRON> (capital N, if the enclitic is written as *Nya*, is preserved in the orthographic form); definite suffix/definite nominaliser suffix <nya,SFX+DER:NOU+DEF>; adverbialiser suffix <nya,SFX+DER:ADV>. Per principle 4, the annotation for the clitic includes as its citation form the independent pronoun *dia* 's/he'.

But what happens when the system cannot disambiguate the enclitic and suffix? One possibility would be to use some statistical approach, perhaps referencing context wider than adjacent morphemes. To explore this possibility, I asked a number of Indonesian-speaking linguists to manually disambiguate a few sentences containing ambiguous instances of *nya*, and found that their answers were split between enclitic and suffix analyses. This means that even for human annotators given access to considerable context, this is not an easy problem to resolve. This makes a statistical approach highly unlikely to succeed, as even the initial step of setting a gold-standard for testing would pose considerable difficulty.

The consequence of this is that many cases of *nya* will have two analyses in the output of any system implementing this MAS. But in practical terms, this is highly undesirable. This morpheme is very productive. Leaving it ambiguous will drastically reduce an annotation system's precision-recall scores, and make using annotated data a lot more complicated for the end-user.

For this reason, a special tag <NYA> is defined, as illustrated in example (4.80), for instances of *nya* that a system cannot disambiguate. This tag indicates that *nya* could be read in context as either pronominal enclitic <ROOT+ECLT+PRON> or definite/definite nominaliser suffix <SFX+DER:NOU+DEF>. The reading of *nya* as adverbialiser is not a possibility, because those cases can all be identified lexically, as noted previously.

(4.80) <buku,ROOT+NOU>
<nya,NYA>

4.3 A full-sentence morphological analysis

Table 4.36 presents a full morphological analysis, using the complete MAS, of a sample sentence from an Indonesian national news portal[35], which reads:

Presiden Joko Widodo (Jokowi) mengaku sudah selesai menyusun kabinet untuk periode kedua pemerintahannya.
'President Joko Widodo (Jokowi) admits that he has finished organising his cabinet for his second period of governance'.

The analyses are presented vertically. For clarity, only E-tags are provided; in the actual implementation, both I- and E-tags may be generated.

[35]https://news.detik.com/berita/d-4744938/jokowi-rampung-susun-kabinet-dahnil-prabowo-siap-jadi-oposisi-atau-koalisi last accessed 25/03/2021

Words (with glosses)	Annotation
Presiden 'president'	<presiden,ROOT+NOU>
Joko 'PN'	<Joko,ROOT+NOU>
Widodo 'PN'	<Widodo,ROOT+NOU>
(Jokowi) 'PN'	<(,PUNC> <Jokowi,ROOT+NOU> <),PUNC>
meng-aku ACV-admit 'admit'	<meng,meN,PFX+DER:VER+ACV> <aku,ROOT+VER>
sudah 'already'	<sudah,ROOT+ADV>
selesai 'finish'	<selesai,ROOT+VER>
meny-(s)usun ACV-organise 'organise'	<meny,meN,PFX+DER:VER+ACV> <usun,susun,ROOT+VER+LOST>
kabinet 'cabinet'	<kabinet,ROOT+NOU>
untuk 'for'	<untuk,ROOT+PRE>
periode 'period'	<periode,ROOT+NOU>
ke-dua NUM-two 'second'	<ke,PFX+DER:NUM> <dua,ROOT+NUM>
pem-(p)erintah-an=nya NOMZR-order-NOMZR 'governance'	<pem,peN,CFX+A+DER:NOM> <erintah,perintah,ROOT+NOM+LOST> <an,CFX+Z+DER:NOM> <NYA>

Table 4.36. Full analysis sample

4.4 Conclusion

In this section, I have presented a novel MAS for Indonesian. The analytic categories for roots, affixes and reduplications have been defined and justified, and their use demonstrated both in isolation and within a full analysis of a single sample sentence. In Chapter 7, we will see how annotation using this MAS is automatically implemented, and how the results are evaluated. This evaluation will include a comparison with the existing state-of-the-art morphological analyser for Indonesian, MorphInd, to which I now turn in Chapter 5.

CHAPTER 5

AN EVALUATION OF THE STATE-OF-THE-ART MORPHOLOGICAL ANALYSER FOR INDONESIAN

5.1 A brief description of MorphInd

MorphInd (Larasati et al. 2011) is a Morphological Analyser (MA) for Indonesian, which runs on UNIX-like operating systems (such as Linux or macOS). MorphInd utilises the TLM approach (Koskenniemi 1983), which operates by modelling both lexicon and grammar abstractly as *Finite State Machines*, or FSMs (Beesley & Kartunnen 2003), a topic to be discussed in Chapter 6. The FSM utilised within MorphInd is a model of Indonesian morphology which is able to recognise affixation, compounding, reduplication, cliticisation, and particle attachment. TLM systems (including MorphInd) use a *compiler* tool to create an FSM program or programs from their linguistic resources, minimally a lexicon and a set of rules describing possible configurations of morph(eme)s within words. MorphInd accepts an input text and yields an output text to which morphological annotations are given, based on the recognition of morphemes within each word by the FSM.

Briefly, MorphInd analyses each input word as follows. Initially, it generates all possible analyses of the word. If a word has only one analysis across both tokenisation and tagging, this analysis appears in the final output. If there are multiple possible analyses, MorphInd applies a sequence of disambiguation procedures. First, it applies *rule-based disambiguation*. This consists of applying 16 rules of thumb that reduce the ambiguity without considering neighbouring word tokens (Larasati et al. 2011:121 refer to this as *unigram* word context). Subsequently, MorphInd performs *statistical disambiguation*, which operates at sentence level. All remaining analyses are statistically scored according to their probability of correctness; the candidate with the highest score is generated as the final output.

The programs used to implement MorphInd are as follows. MorphInd itself is implemented in, and thus requires, the Perl programming language. MorphInd utilises foma

(Hulden 2009) as its FSM system. The parameters needed for statistical disambiguation are built using the KenLM program (Heafield 2011).

5.2 Why MorphInd is the state-of-the-art morphological analyser for Indonesian

There exist two MAs for Indonesian. The earlier was built by Pisceldo et al. (2008); I refer to this as PMA (*Pisceldo et al.'s Morphological Analyser*) in this thesis. The more recent is MorphInd[36] (Larasati et al. 2011), which, I argue, is presently the state-of-the-art MA for Indonesian. MorphInd is presented as an advance upon PMA by Larasati et al. (2011:120-121).

Why should MorphInd be considered state-of-the-art? First, MorphInd is completely functional and in continuous development. This stands in contrast to PMA, which due to some technical issue does not run on current systems; I have contacted PMA's authors regarding this matter, but to date, not received any response. The webpage from which PMA may be downloaded was last updated in 2011 and gives no indication that PMA has undergone any modification since 2008[37]. MorphInd, however, has undergone three modifications since its initial release in 2011; the current version of MorphInd is v.1.4. In MorphInd v.1.2, the author added a disambiguation module, which was then improved in version v.1.3. The most recent update (v.1.4) added a module to handle analysis of compound words.

This may explain why MorphInd, rather than PMA, is used by other Indonesian NLP systems, as the following non-exhaustive survey illustrates. Rashel et al. (2014) use MorphInd to build a rule-based POS tagger for Indonesian. Green et al. (2012) use MorphInd to build an Indonesian dependency treebank. MorphInd has been used to annotate the IDENTIC Corpus, built by Larasati (2012). It has also been used by Denistia & Bayeen (2019) to analyse a part of the Indonesian data in Leipzig Corpora Collection (LCC) (Goldhahn et al. 2012); see 2.2.3.

Another likely reason why MorphInd, instead of PMA, is reused in the abovementioned systems is that MorphInd is freely redistributable. All its software components are released

[36] https://septinalarasati.com/morphind/ (accessed 26/04/2021)
[37] https://bahasa.cs.ui.ac.id/nlp-resources (accessed 26/04/2021)

under a free licence[38]. Conversely, PMA's core NLP program, xfst (Kartunnen & Beesley 2003) is patent-encumbered, such that full access to all functionality is only available in the commercial version (according to Larasati et al. 2011:121). This could be another reason why the Indonesian NLP systems previously mentioned prefer to embed MorphInd rather than PMA. Second, MorphInd's annotation scheme represents an improvement relative to PMA in terms of tokenisation and tagging, as I have discussed in section 3.5.2.

5.3 Larasati et al.'s evaluation

The creators of MorphInd evaluated its performance, as reported in Larasati et al. (2011:126-128). That evaluation focuses on just one aspect of MorphInd's performance, that is, its *coverage* measured over word types and tokens. Although not spelt out by the authors, it is very likely that *coverage* refers to the proportion of words that are *not* analysed as *unknown* (tagged with <x>). This is the most sensible interpretation given the totality of what Larasati et al. say about coverage. Moreover, the fact that MorphInd does not leave *any* word untagged (as shown by Table 5.1) rules out the interpretation that out-of-coverage words are left completely unanalysed.

Input[39]	Output
PT KAI melarang aktivitas di sekitar jalur rel kereta api yang hampir menyebabkan kecelakaan.	^pt<n>_NSD$ ^kai<x>_X—$ ^meN+larang<v>_VSA$ ^aktivitas<n>_NSD$ ^di<r>_R—$ ^sekitar<d>_D—$ ^jalur<n>_NSD$ ^rel<x>_X—$ ^kereta<n>_NSD$ ^api<n>_NSD$ ^.<z>_Z—$ ^yang<s>_S—$ ^hampir<d>_D—$ ^meN+sebab<n>+kan_VSA$ ^ke+celaka<a>+an_NSD$ ^.<z>_Z—$
Hal itu berkaitan dengan acara Jakarta *Mystical Tour*	^hal<n>_NSD$ ^itu<b>_B—$ ^ber+kait<v>+an_VSA$ ^dengan<r>_R—$ ^acara<n>_NSD$ ^jakarta<n>_NSD$ ^mystical<x>_X—$ ^tour<f>_F—$

Table 5.1. A sample of MorphInd input and output[40]

[38] Specifically, the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.
[39] PT = *Perseroan Terbatas* (state-owned-enterprise), KAI = *Kereta Api Indonesian* (Indonesian railway system), Foreign words are written in italic.
[40] This text sample is taken from an Indonesian news website: https://news.detik.com/berita/d-4772024/kai-larang-tur-mistis-datangi-lagi-rel-lokasi-tragedi-bintaro-1987 (last accessed 26/04/2021)

Larasati et al. report MorphInd's coverage in comparison to PMA's (Table 5.2). Both MorphInd and PMA have low coverage when measured by either types (below 55%) and tokens (below 85%). This data suggests that MorphInd is slightly better than PMA in terms of token coverage, while PMA is better in terms of type coverage.

	Test set	# sentences	Token coverage	Type coverage
MorphInd	T5K	5,000	84.69±0.28 (highest)	50.77±0.70
	T10K	10,000	84.61±0.10	47.19±0.35
PMA	T5K	5,000	83.62±0.27	54.95±0.76 (highest)
	T10K	10,000	83.46±0.06	51.39±0.05

Table 5.2. Percentage coverage of word tokens and types for MorphInd and PMA (adapted from Larasati et al. 2011:128)

Larasati et al. argue that the results are affected by differences in the two systems' resources. MorphInd utilises a *root* lexicon (3,000+ entries), whereas PMA utilises a lexicon that includes both *roots* (monomorphemic words) and polymorphemic *words* (29,000+ entries in total). This is evident from both (a) Larasati et al.'s explanation that PMA's lexicon contains *affixed forms* (i.e. polymorphemic words) as well as roots and (b) direct observation of PMA's lexicon (PMA's resources are openly accessible).

Larasati et al. explain that MorphInd's better token coverage is because, unlike PMA, its rules handle clitics and particles. They argue that PMA's better type coverage is because of the great difference in lexicon size (an order of magnitude). This is not completely accurate. It seems clear that PMA's better type coverage is not only a matter of lexicon *size* but also lexicon *type*. As we have seen, MorphInd's lexicon contains only roots, while PMA's lexicon also has polymorphemic words – forms which must be dealt with by rule-based analysis, rather than lexical lookup, in MorphInd.

In an effort to demonstrate that MorphInd can outperform PMA on coverage of both types and tokens, Larasati et al. then report another experiment. This time, they modify PMA's lexicon to include only items also in the MorphInd lexicon. The results are shown in Table 5.3.

	Test set	#sentences	Tokens	Type
MorphInd	T5K	5,000	84.69±0.28 (highest)	50.77±0.70 (highest)
	T10K	10,000	84.61±0.10	47.19±0.35
modified PMA-comparable	T5K	5,000	81.91±0.18	44.60±0.66
	T10K	10,000	81.82±0.06	40.83±0.31

Table 5.3. Percentage coverage of word tokens and types for MorphInd and modified PMA (adapted from Larasati et al. 2011:128)

However, I would argue that this comparison is misleading. Using the shared lexicon means removing PMA's polymorphemic lexicon entries. This cripples PMA's performance, because unlike MorphInd, PMA is *designed* to use a lexicon of both monomorphemic and polymorphemic words. Under these conditions, MorphInd indeed exceeds the modified PMA in both type coverage and token coverage. However, this cannot justify a claim that MorphInd outperforms PMA, because the comparison is not fair. The result simply demonstrates that MorphInd and PMA use different approaches to polymorphemic words (rules and lexical entries, respectively).

5.4 Evaluation of MorphInd's morphological annotation scheme

The Morphological Annotation Scheme (MAS) proposed by Larasati et al. (2011) and used in MorphInd was evaluated at length as part of my review of such schemes in section 3.5.2. Here, I present some advantages and drawbacks of Larasati et al.'s MAS (LM) in abbreviated form as a prelude to my evaluation of MorphInd's implementation (in 5.5).

That LM's morphological analytic labels, along with the word POS labels, are linked to the whole word token, is characteristic of a morphosyntactic or POS tagger. This means that MorphInd has the advantage that it can be used for POS tagging purposes, even though it is a morphological analyser.

MorphInd can tokenise words into different kinds of morphemes (root, affix, clitic, particle). The earlier PMA does not handle clitics and particles. With its better tokenisation scheme, MorphInd enables searches for a wider variety of morphological tokens.

In LM, not only words but also roots are POS tagged. Therefore, users can perform searches based on the root's POS as well as the word's. MorphInd's output is unambiguous. With this kind of output, evaluation is relatively easier, as we never have to evaluate more than one analysis.

Despite the above-mentioned advantages, I have argued that LM, even if flawlessly implemented, cannot serve a number of anticipated user needs (see section 3.5.2).

LM indeed tokenises words into morphemes, but there are limitations. First, affixes and clitics that exhibit allomorphy (e.g. *meng-, men-, meny-*; see 2.1.2) of a morpheme are presented only in citation form (in this case *meN-*). This prevents users from searching the annotation for morph tokens based on their allomorphic (orthographic) form. Nor does LM distinguish the tokenisation of a circumfix from that of a combination of prefix and suffix. Moreover, LM tokenises reduplications identically to the corresponding non-reduplicated morpheme. The reduplicated morpheme (the copy) is not tokenised or analysed. This prevents users from searching the annotation for reduplications based on their form.

Another limitation is that instead of linking affix tags to affix tokens, MorphInd adds affix tags to the word-level POS tags. Larasati et al. (2011) refer to the combination of word POS tag and affix tag as a *morphological tag*. Combining POS and morphological features is a common characteristic of morphosyntactic tagging.

LM lacks any tags for formal categories. This means that users cannot search by formal morphological criteria (e.g. prefix, circumfix, or full/partial/imitative reduplication). Of LM's functional tags, a number are over-generalised. For instance, the label *plural* is given to all reduplications. This is inaccurate, because reduplication can also mark other functions including *iterative* or *reciprocal*. Similarly, the form *nya* is always annotated as a third person singular clitic pronoun. This is inaccurate, as in certain contexts, *nya* is a definite suffix (see 4.2.3.3.4.1). Another shortcoming is that certain common morphologically marked functions, such as equative adjective and causative voice, are absent from LM.

Certain analytic categories, such as *determiner, copula,* or *modal,* are treated as POS categories in LM but not by reference grammars of Indonesian. For instance, *modal* is a considered a subcategory of adverb by Alwi et al.'s (1998) grammar. But in LM, *modal* and *adverb* are two separate top-level POS categories. This is likely to cause difficulty for users devising searches searching based on LM tagging, particularly if they rely on knowledge acquired from reference grammars.

The benefit of MorphInd's unambiguous output has been discussed earlier. However, on the negative side, it bears noting that MorphInd actually yields multiple analyses at an intermediate stage, one of which is correct. Since no ambiguity is allowed, MorphInd has to choose one analysis at the final stage (output), but it is not guaranteed to be the correct one.

Earlier, advantages of MorphInd providing word-level annotation were discussed. However, from the perspective of morpheme-level annotation, LM is evidently lacking, not least because affixes are left untagged (see section 3.5.2.4). This is actually a major disadvantage; users cannot search the corpus based on annotation which does not exist.

The conclusion of my assessment, previously presented in section 3.5.2, is that for morphological annotation at morpheme level, a new scheme is required to avoid the disadvantages of LM. This motivated the creation in Chapter 4 of a new and, I argue, improved scheme that fulfil this goal. I shall now proceed to evaluate MorphInd's *implementation* of the analysis, independently of these problems.

5.5 Evaluation of MorphInd's performance

5.5.1 Output format

Understanding MorphInd's output format is important, not only for computational processing, but also and particularly for the purpose of the upcoming manual evaluation exercise. Table 5.4 gives a concise representation of MorphInd's procedures, and corresponding output formats, for two primary tasks: tokenisation and tagging. It shows how morphemes are delineated from one another in MorphInd's output. It also shows the location and the format of the analytic tags.

Task		Procedure	Output format
1	Tokenise words to morphemes	Tokenise all words to the corresponding morphemes' citation forms	All morphemes are presented in lowercase (e.g London > london; Margaret > margaret; NaCl > nacl; IELTS > ielts) Exception: citation forms of allomorphs, e.g *meng* > *meN*, *peng* > *peN*
		Set morpheme boundary	+ symbol
2	Assign tag	Roots are assigned coarse POS tag, but affixes are not tagged	Single letter tag, e.g. <n>, on roots
		Words are assigned morphosyntactic tags	Mnemonic letter tag (e.g. VSA for verb singular active)

Table 5.4. MorphInd's output structure and analysis procedures

A sample of MorphInd's tokenisation and tagging mechanism is diagrammed in Figure 5.1. The verb *men(t)ulis* 'write' is annotated as the combination of active prefix *meN-*, followed by verb root *tulis* 'write', whose POS is given inside angle brackets. These morphemes are delineated by +. Subsequent to the final morpheme, an underscore is given followed by the word's POS tag, VSA.

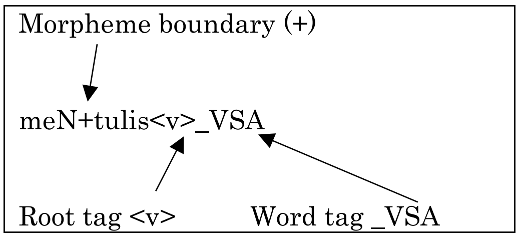


Figure 5.1. Example of assignment of morpheme boundaries, root tags and word tags by MorphInd

5.5.2 Rationale for the evaluation

Even given the problems with LM, evaluation of MorphInd's performance is still required for at least two reasons. First, we need to verify whether LM *can* be implemented in full. This is very difficult if we rely only on Larasati et al.'s evaluation, because that exercise only measures coverage. Second, we need to assess whether Larasati et al.'s (2011) evaluation is still up-to-date. That evaluation utilised the first version of MorphInd. Since then, MorphInd has undergone a number of important upgrades, which may be expected to have improved performance.

5.5.3 Testbed corpus

For a fair comparison, a new evaluation would ideally be performed against the test set of sentences used by Larasati et al. Unfortunately, this is not possible. Larasati et al.'s test set drew on two sources: the PAN localisation corpus (Adriani and Hammam 2009) and an archive of movie subtitles, later released as the IDENTIC Corpus (Larasati 2012). However, the test sets were created by random selection of sentences. A new randomly selected test set from these sources would be distinct from Larasati et al.'s previous test set. Another basis for the evaluation is needed. For this reason, I did not replicate Larasati et al.'s approach to creating a testbed. Another reason to adopt a different course is to avoid genre homogeneity. Larasati et al.'s testbed is almost all composed of newspaper data. With such test data, it may legitimately be questioned to what extent evaluations would reflect system performance on more heterogeneous texts.

As discussed in section 2.2, my 10K-word testbed was created by random selection of paragraphs from a larger text collection that was balanced by domain. This is an appropriate replacement for Larasati et al.'s test sets because all the domains in those test sets map to domains in my balanced text collection (and thus, the testbed), as Table 5.5 shows.

Text domains in Larasati et al.'s test set	Equivalent domain(s) in testbed
International	World affairs
Sport	Leisure
Science	Applied science, natural or pure science or social science
Economy	Commerce and finance
Movie subtitle	Imaginative and creative writing

Table 5.5. Mapping Larasati et al.'s domains to testbed domains

The testbed is much smaller than Larasati et al.'s test set, unavoidably, because my approach to evaluation requires me to manually examine each token in the output. Manual verification permits an advance on Larasati et al.'s (2011) evaluation, as explained in 2.2.

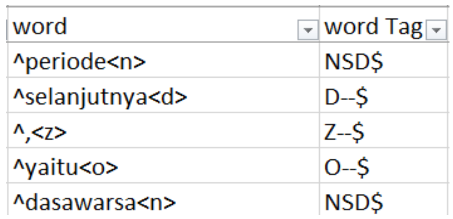
5.5.4 Procedure for evaluation

The eight-step procedure of my evaluation of MorphInd is summarised in Table 5.6, and will be explained in detail in the remainder of this section.

	Procedure	Further detail
1	Run MorphInd on the testbed	To get authentic output from MorphInd
2	Clean and adapt the output to vertical format (one token per line) and transfer to excel spreadsheet	To enable 1. clarity of manual verification on *token level* 2. calculation of MorphInd's performance
3	Define evaluation criteria	By identifying MorphInd subsidiary tasks (discussed in steps 5, 6, 7) and primary task (step 8)
4	Examine spelling and writing convention correctness	To divide words in the testbed into 1. misspelt (these words will be excluded from the evaluation; encoded as EXC for *exclude*) 2. valid words, whose total number, N, is used in the performance calculation
5	Examine morpheme boundary correctness	To determine whether morpheme boundaries are correctly assigned for each token. Possible results are as follows: 1. M = no boundary supplied to correctly analysed monomorphemic word 2. P = correct morpheme boundaries supplied to polymorphemic word 3. M/P = monomorphemic words but incorrectly analysed as polymorphemic 4. P/M = polymorphemic words but incorrectly analysed as monomorphemic 5. P/P = polymorphemic words, analysed as polymorphemic, but morpheme boundaries are wrong
6	Examine root tag correctness	Roots are coded as follows 1. CR = correctly tagged root 2. IR = incorrectly tagged root (including roots tagged as unknown (X))
7	Examine word tag correctness	Words coded as follows 1. CW = correctly tagged word 2. IW = incorrectly tagged word (including words tagged as unknown (X))
8	Calculate aggregate performance according to coverage and accuracy	These measures are defined as: 1. Coverage: the proportion of in-coverage tokens assigned tags other than X, out of N 2. Accuracy: the proportion of words receiving correct root tag, word tag and morpheme boundary placement (termed as *Correctly Analysed Tokens* (CAT)) out of N

Table 5.6. Evaluation procedure

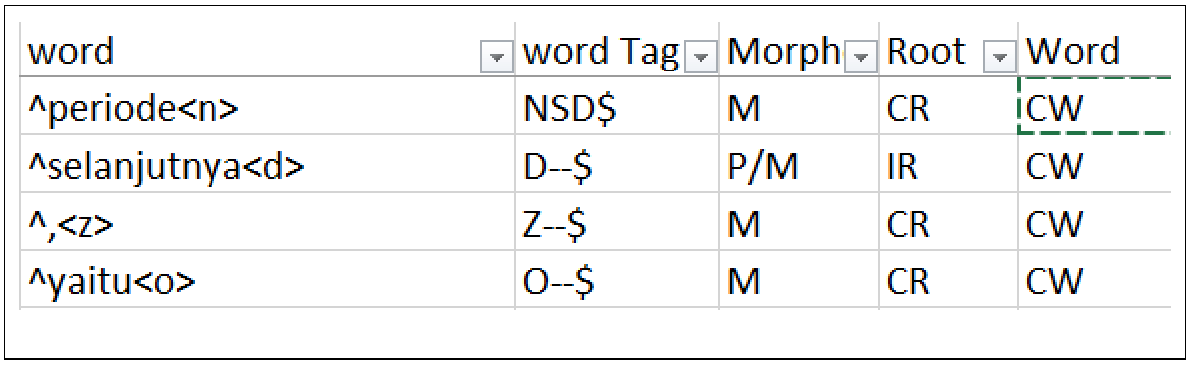
First, I ran the current version of MorphInd (v.1.4) over the testbed to generate authentic MorphInd output to be evaluated. Second, I adapted the output to one-word-per-line format, which I transferred into a spreadsheet (using Microsoft Excel). This format was optimal because I manually analysed every word, line by line, based on each word's local context. For each word, I assigned a number of evaluation categories relevant to measuring MorphInd's performance.

Vertical content output	Spreadsheet
^periode<n>_NSD$ ^selanjutnya<d>_D—$ ^,<z>_Z—$ ^yaitu<o>_O—$ ^dasawarsa<n>_NSD$	

word	word Tag
^periode<n>	NSD$
^selanjutnya<d>	D--$
^,<z>	Z--$
^yaitu<o>	O--$
^dasawarsa<n>	NSD$

Table 5.7. Transferring the output to a spreadsheet

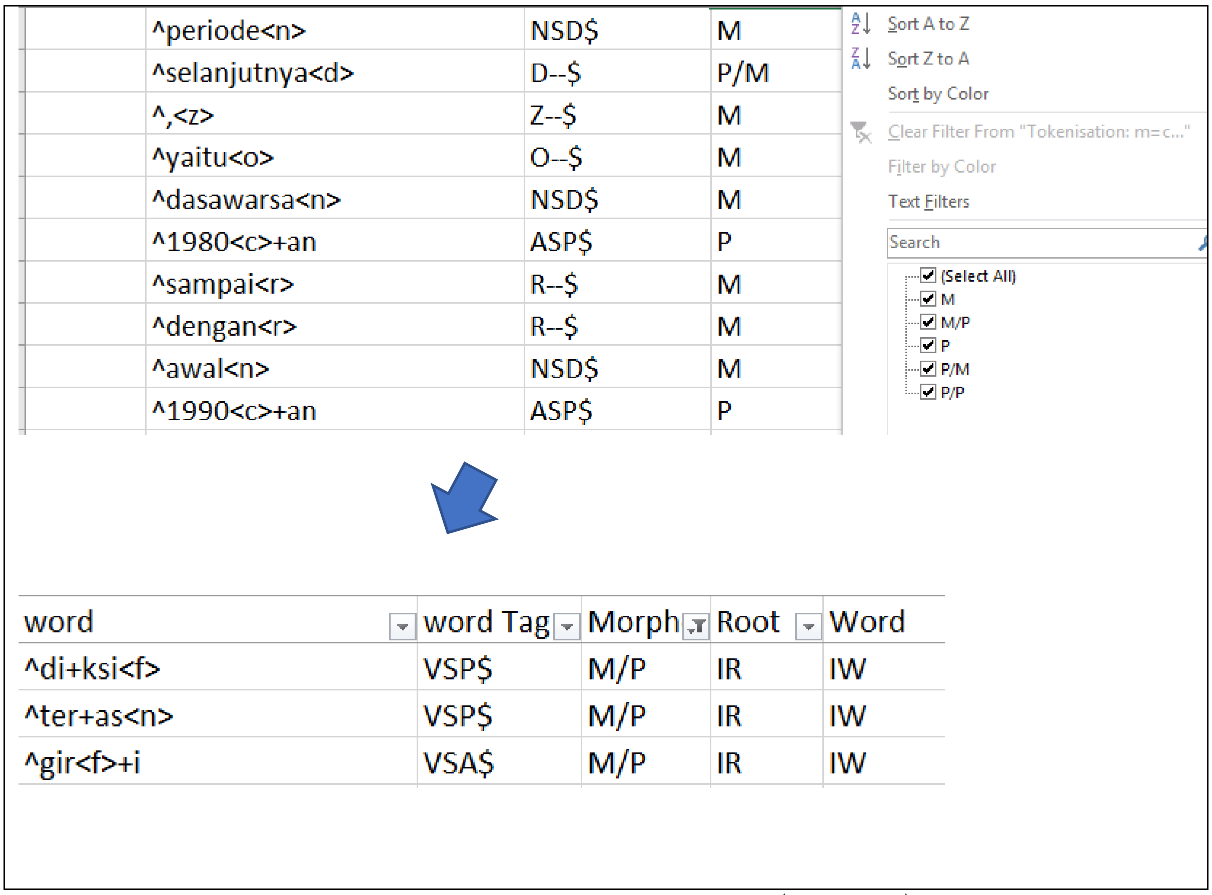
Each point of the evaluation was coded in a subsequent column of the spreadsheet row containing the word token. This allows different aspects of the evaluation (morpheme boundary, root and word tag) to be separately encoded and conveniently presented using the codes listed in Table 5.6 and explained in more detail below. For instance, the word *periode* was correctly analysed in all respects. MorphInd correctly recognised it as a monomorphemic word (code M). It supplied the correct root tag <n> (code CR). It also supplied the correct word tag NSD (code CW). On the other hand, *selanjutnya* is a polymorphemic word incorrectly analysed as monomorphemic by MorphInd (code P/M). MorphInd did not identify the root, and therefore its root tagging is incorrect (code IR). However, the word as a whole may be still be tagged correctly, and in this case it is (code CW).



word	word Tag	Morph	Root	Word
^periode<n>	NSD$	M	CR	CW
^selanjutnya<d>	D--$	P/M	IR	CW
^,<z>	Z--$	M	CR	CW
^yaitu<o>	O--$	M	CR	CW

Figure 5.2. Evaluation categories coded across three columns

Excel's *filtering* function makes it possible to focus on one or more categories of interest. Figure 5.3 shows how the underlying data may be filtered to display only words coded M/P for morpheme boundary correctness.



^periode<n>	NSD$	M
^selanjutnya<d>	D--$	P/M
^,<z>	Z--$	M
^yaitu<o>	O--$	M
^dasawarsa<n>	NSD$	M
^1980<c>+an	ASP$	P
^sampai<r>	R--$	M
^dengan<r>	R--$	M
^awal<n>	NSD$	M
^1990<c>+an	ASP$	P

word	word Tag	Morph	Root	Word
^di+ksi<f>	VSP$	M/P	IR	IW
^ter+as<n>	VSP$	M/P	IR	IW
^gir<f>+i	VSA$	M/P	IR	IW

Figure 5.3. Using Excel's filter to select specific sub-categories (here, M/P) from morpheme boundary correctness

Third, I defined criteria for the evaluation. Larasati et al. 's (2011) evaluation only considered coverage. Unlike Larasati et al., in addition to coverage, I also evaluated aggregate token accuracy. An aggregate-accurate token is defined as one given a correct output on *all* of MorphInd's three subsidiary tasks, namely: (a) assigning morpheme boundaries; (b) assigning a root tag; and (c) assigning a word tag (as explained in section 5.4). The evaluation methods for each tasks constitute steps 5, 6, and 7. Measuring aggregate token coverage and accuracy is step 8.

Every word was examined in its local context. For instance, the monomorphemic word *paling* may be either an adverb 'most' or a verb 'turn away'. In one instance in the data, *paling* modifies an adjective (local context: *paling monumental* 'most monumental'), such that the correct tag is an adverb tag. However, MorphInd produced a verb tag. This word was thus considered incorrectly tagged.

MorphInd's performance was evaluated against its annotation scheme *as is*. For instance, in a number of instances in the data, the definite marker *–nya* is incorrectly tagged as the pronominal enclitic =*nya*. However, MorphInd tags *all* uses of *nya* as pronominal enclitic. This is clearly by design (see 3.5.2; LM has no tag for the definite marker). The software is fully complying with the scheme, even though the scheme is not completely correct linguistically. Cases like these, therefore, do not count as mistakes for the purpose of the evaluation. This stands in contrast to words such as *paling*, where the correct tag (verb) *does* exist within LM but has not been supplied by MorphInd.

Fourth, I checked the validity of the input tokens as words of standard written Indonesian. The aim here is to exclude misspelt words (code EXC) from the evaluation, as MorphInd cannot be expected to produce well-formed output from badly-formed input. The total number of valid word tokens, N, is used as the denominator value for all calculations of proportions of accuracy. Invalid words include incorrectly conjoined words and spelling errors. For example, I found a number of instances of locative preposition *di*, wrongly written as joined without space to following demonstratives such as *sini* 'here' (i.e. *disini* instead of standard *di sini*). An example spelling error is *sebag<u>a</u>inya* 'etc.' being written as *sebag<u>aa</u>inya*.

Fifth, I examined whether or not morpheme boundaries were correctly assigned. I coded each word token as follows: correctly analysed monomorphemic word, with no morpheme boundaries assigned (M); polymorphemic word assigned the correct morpheme boundaries (P); monomorphemic word incorrectly analysed as polymorphemic (M/P); polymorphemic word incorrectly analysed as monomorphemic (P/M); polymorphemic word analysed as polymorphemic but assigned incorrect morpheme boundaries (P/P). The percentage evaluation measure for this tokenisation task (T) is then calculated as follows. The first two codes (M and P) represent accurate boundaries; thus, the percentage of tokens with these codes is the percentage of all tokens correctly assigned morpheme boundaries.

$$T = \frac{\sum(M+P)}{N} \text{ x } 100$$

The sixth and seventh steps in the procedure evaluate correctness of root and word tags. Roots and words tagged with *unknown* <X> are considered *incorrect*. The accuracy measures for root tagging (RT) and word tagging (WT) are then respectively the percentages of correct roots (CR) and correct words (CW) out of N.

$$RT = \frac{\sum CR}{N} \text{ x } 100$$

$$WT = \frac{\sum WR}{N} \text{ x } 100$$

At the eighth step, two further measures are calculated: token coverage and token accuracy. As noted earlier (see 5.3), I deduced that Larasati et al.'s *coverage* is defined as the proportion of word tokens not analysed as *unknown* <X>, and I follow this definition. Thus, coverage (C) is obtained through the following calculation:

$$C = \frac{N - \sum X}{N} \text{ x } 100$$

Aggregate token Accuracy (A) is calculated as the number of *fully* correct analyses of tokens as a percentage of N. To be considered correct on aggregate (code CAT), a word token must have: (a) correct morpheme boundaries (code M or P), (b) the correct root tag (code CR), and (b) the correct word tag (code WR). A is then calculated as follows:

$$A = \frac{\sum CAT}{N} * 100$$

5.5.5 Morpheme boundary assignment accuracy

MorphInd's accuracy in assigning morpheme boundaries in this exercise is 96.6 %. The system correctly identified 1,697 polymorphemic words (P) and 8,188 monomorphemic words (M) (see Table 5.8). The error rate is only 3.4 %.

Categories	Number	%	%
M = Monomorphemic (correct)	8,188	80.0%	96.6% correct
P = Polymorphemic (correct)	1,697	16.5%	
M/P = Monomorphemic analysed as polymorphemic (error)	4	0.1%	3.4% error
P/M = Polymorphemic analysed as monomorphemic (error)	321	3.1%	
P/P = polymorphemic but morpheme boundary is wrong (error)	18	0.1%	

Table 5.8. Frequency of codes for morpheme boundary identification

Most errors are of type P/M, polymorphemic words incorrectly analysed as monomorphemic. There are two possible reasons for this. First, unknown words are *always* analysed as monomorphemic. In consequence, MorphInd supplies inaccurate morpheme boundaries for unknown polymorphemic words. Second, MorphInd's rule-based disambiguation prefers monomorphemic analyses over polymorphemic word analyses. The disambiguation rules in question are not documented in the literature but are viewable in the public source code repository[41]. I quote some comments on these rules in the code in Table 5.9; Larasati explicitly mentions a preference for monomorphemic word analyses, phrased as ‘choose the full word’.

If ambiguous between having a foreign word + something or a full word, choose the full word
If ambiguous between having a morpheme + foreign word or a full word, choose the full word
If ambiguous between segmented morpheme and Foreign word, choose foreign word
If ambiguous between 'se+' and a full word, choose a full word
If ambiguous between '+i_' and a full word, choose a full word
If ambiguous between having a clitic and not, choose not
If ambiguous between having a clitic and not, choose not
If ambiguous between '+an_' and a full word, choose a full word
If ambiguous between 'di+' and a full word, choose a full word
If ambiguous between 'peN+' and a full word, choose a full word
If ambiguous between 'ter+' and a full word, choose a full word

Table 5.9. Comments in the MorphInd code on disambiguation rules relevant to tokenisation

5.5.6 Word and root tagging accuracy

Critically, Larasati et al.’s (2011) evaluation does not consider tagging accuracy. However, MorphInd’s word tagging task is equivalent to POS tagging, and accuracy is fundamental in judging a POS tagger’s success. MorphInd’s word tagging accuracy is therefore of high interest.

In the testbed, MorphInd provides 9,179 word tokens with the correct word tag; its word tagging accuracy is therefore 89.8 % (see Table 5.11). This can be considered low, since POS tagger accuracy is routinely above 90% (see Table 5.10).

POS tagger	Target language	Reported accuracy
CLAWS (Garside 1987)	English	96.97%
Stanford POS tagger (Toutanova 2003)	English	96.86%
HMM POS tagger for Indonesian (Farizki & Purwarianti 2010)	Indonesian	99.40%
HMM POS tagger for Malaysian (Mohamed et al. 2011)	Malaysian	98.60%
Joint model POS tagger for Korean (Park and Seo 2015)	Korean	96.17%

Table 5.10. Some POS taggers’ accuracy rates

[41] The repository can be accessed from this link: https://svn.ms.mff.cuni.cz/svn/MorphInd/trunk/MorphInd.v.1.4. Enter ‘public’ for both id and password when requested. The disambiguation rules are in the file *MorphInd.pl*. (last accessed 26/04/2021)

Meanwhile, MorphInd's root tagging accuracy is similar to word tagging accuracy at 89.5% . This is a novel finding, since Larasati et al. (2011) do not measure root tagging accuracy at all. Root tagging is a task not commonly performed by POS taggers, but crucial for a morphological analyser.

	Word tags	%	Root tags	%
Correct	9,179	89.8	9,523	90.3
Incorrect: analysis error	335	3.3	289	2.8
Incorrect: word unknown	714	6.9	714	6.9

Table 5.11. MorphInd's word tagging and root tagging accuracy

There are at least two reasons for MorphInd's relatively low tagging accuracy. First, the rate of unknown, i.e. out-of-coverage, words is quite high. I identified 714 words assigned the unknown tag X—; this lowers the maximum possible accuracy to 93% for both word and root tagging, as around 7% of the tokens are unknown (see further section 5.5.8).

Second, MorphInd assigns incorrect analyses to some known words. It seems that MorphInd struggles at tagging Indonesian proper nouns with foreign-word characteristics. For instance, Arabic loanwords starting with *al-,* such as *al-quran* 'Quran' or *al-jin* 'lit. the spirits; name of a chapter in the Quran', are *all* tagged by MorphInd as foreign words, even if they are clearly treated as proper nouns in the context of Indonesian. This seems to be caused by one of MorphInd's disambiguation rules, which assigns the foreign word tag (F—) to all words beginning with *al-, an-* or *el-*. This rule seems to be applied with no constraints, as in the testbed, *al-quran* and *al-jin* are tagged as foreign words (F—). But these, and many similar words, are proper nouns in Indonesian despite the morphological cue *al-*, which is the Arabic definite prefix (also transliterated *el-* or, with allomorphy, *an-*).

Other instances of words with foreign characteristics in the corpus are *Apple* and *News*. In isolation, these words do seem to be foreign words, but they are actually treated as proper nouns in Indonesian. This can be deduced from the local contexts of these examples. *Apple* (with uppercase A) is a phone brand; *News* (with uppercase N) is part of the name of an Indonesian news agency, *Antara News*. Contextually both clearly operate as normal Indonesian proper nouns. It is common, especially in the modern world, for languages to have many proper nouns that are loanwords.

MorphInd even inaccurately tags proper names like *Steve* or *Chris* as foreign words. These are very common English personal names, not commonly used to name Indonesians[42]. However, that does not change the fact that they are clearly being used as proper nouns, not foreign words, if some person called *Steve* or *Chris* is discussed in a fully Indonesian stretch of text.

It is noticeable that MorphInd consistently fails to properly analyse reciprocal voice shown by reduplication plus *meN-*, as illustrated in Table 5.12. A tag DASH, which is not defined in LM, is added to the word tag, causing the source and copy of the reduplication to be treated as distinct word tokens. In consequence, the word tag is also incorrect: the token receives two VSA tags (singular active verb), whereas there should only be one VPS tag (plural, i.e. reciprocal, active verb). The same error affects compound word reduplication. For the second example in Table 5.12, MorphInd generates four NSD tags (singular noun), whereas according to LM, there should only be one word, tagged NPD (plural noun). Such incorrect splitting-up of tokens will usually cause the word tag to be incorrect in this way. For instance, following LM, *es krim-es krim* should be analysed as *es<n>+krim<n>_NPD,* but is instead analysed as in Table 5.12.

Word	Incorrect analysis
pukul-mem(p)ukul 'hit one another'	^pukul<v>_VSA$**DASH**^meN+pukul<v>_VSA$
Es krim-es krim 'ice creams'	^es<n>_NSD$ ^krim<n>_NSD$**DASH**^es<n>_NSD$ ^krim<n>

Table 5.12. Incorrect morpheme boundary for compound word reduplication and reciprocal reduplication with *meN-*

5.5.7 Aggregate accuracy

As I outlined in 5.5.4, aggregate token accuracy requires all three analyses to be fully correct: morpheme boundary assignment, root tagging, and word tagging. Tokens to which

[42] Likewise, some country names such as Austria and Algeria are tagged as foreign words)

MorphInd supplies a partially correct analysis are considered incorrect. Here is an illustration. The word *pen-dahulu-an* ‘preview’ is a noun, composed of an adverbial root *dahulu* ‘previously (adv)’ and nominaliser *pe—an*. MorphInd annotates it incorrectly as *pendahulu<n>+an_NSD*. Here, *pendahulu* ‘predecessor’ is analysed as the root, due to incorrect morpheme boundary detection. The word tag NSD (singular noun) for this token is correct, but because MorphInd failed to recognise the root’s boundaries, the root tag is necessarily wrong. Therefore, the analysis is, in aggregate, incorrect.

	Aggregate token accuracy	%	Aggregate type accuracy	%
Correct	8,939	87.3	2,207	73.3
Incorrect: analysis error	575	5.6	226	7.2
Incorrect: word unknown	714	6.9	580	19.2

Table 5.13. Aggregate accuracy evaluation

MorphInd’s aggregate token accuracy is 87.3%. Of the incorrect aggregate analyses, slightly more than half were unknown words (tagged <X>); the remainder were known (in-coverage) words. This shows that MorphInd fails to carry out analyses more often than it incorrectly analyses words. This finding is paralleled in aggregate type accuracy, in that the number of unknown types is almost three times greater than the number of erroneously analysed types.

5.5.8 Coverage

MorphInd’s token coverage, the percentage of tokens to which it assigns an analysis other than <X>, correct or incorrect, is 93.0%. Larasati et al. (2011) report MorphInd’s coverage to be less than 85% (see 5.3). It is very likely that improvements to MorphInd (see 5.5.2) have increased its coverage since 2011. Even so, 7.0 % of tokens in the testbed are out of coverage.

	Tokens	%	Types	%
Known (in coverage)	9,514	93.0	2,356	82.7
Unknown (out of coverage)	714	7.0	493	17.3
Total	10,228		2,849	

Table 5.14. MorphInd's coverage

Classification	Tokens	%	Types	%
Known (in coverage)	9,514	93.0	2,356	82.7
Unknown (out of coverage)				
Foreign words	49	0.5	33	1.2
Proper nouns (including abbreviations, acronyms)	275	2.7	159	5.6
Others	390	3.8	301	10.5
Total	10,228		2,849	

Table 5.15. MorphInd's coverage, breaking down out-of-coverage items

I grouped the unknown words into three categories: foreign words (e.g. English *ideologi* Dutch *bestuursdwang* 'legal use of force by government officials'); proper nouns, including abbreviations and acronyms (e.g. *UMY* for *Universitas Muhammadiyah Yogyakarta*); and finally the category of *others*, i.e. anything not in the first two categories (e.g. *pesisir* 'coastal', *endap-an* 'sediment'). The first two groups in the data are all monomorphemic[43], thus segmenting the category further into polymorphemic and monomorphemic is not possible. However, *others* includes polymorphemic words; see breakdown in Table 5.16. This breakdown is useful to further classify the errors.

Beginning, however, with the three main Unknown categories, we see that MorphInd does seem to struggle with foreign words, although a large number of English words are correctly recognised. However, this is the cause of relatively few errors. I notice that MorphInd is not very good at recognising foreign words from non-English languages, such as Dutch, Chinese, or Javanese. Therefore, the error rate can be expected to rise if MorphInd is applied to texts with a larger proportion of non-English foreign words.

Surprisingly, many widely used Indonesian abbreviations and proper nouns are out-of-coverage, including *PNS* 'civil servant', *Yesus* 'Jesus' or *Yogyakarta* (name of a city). *PNS* or *Pegawai Negri Sipil* 'civil servant' is a very prevalent job in Indonesia, and this abbreviation frequently occurs in news texts. The name *Yesus* is frequently found in Christian religious texts;

[43] It is possible to create a polymorphemic word containing foreign elements, such as *didownload* 'be downloaded', in which *di-* is a passive prefix and *download* is a loan from English. But in the test data, no such words are found.

in Indonesia, Christians are the second largest religious group (after Muslims). Meanwhile, Yogyakarta is a popular tourist destination (Indonesia's second most popular in Indonesia after Bali), and its name appears quite frequently in leisure texts. Yet, all these are unknown to MorphInd.

Let us now look at the different subcategories of the Unknown third category, *others*, given in Table 5.16 (as noted earlier, for present purposes, foreign words, proper nouns, abbreviations and acronyms, that is everything under the former two Unknown categories, are considered monomorphemic[44]).

Word structure	Tokens	Types
Polymorphemic	85	57
Monomorphemic	305	230
Total	390	287

Table 5.16. Out-of-coverage polymorphemic and monomorphemic words from category *others*

The 57 polymorphemic out-of-coverage words include affixed words (e.g. *ber-ideologi* 'have an ideology', *me-mungkin-kan* 'make possible'), affixed compound words (*ke-tidak-ber-daya-guna-an* 'thing that is not useful or powerful') and affixed words with additional cliticisation (*pikir-an=ku* 'my thought').

What possible explanations are there for these words failing to be analysed? In some cases, it may be that the word's root is absent from the lexicon. Then, even if MorphInd uses rules to correctly recognise the affixes and clitics, the root cannot be recognised; in consequence, the word as a whole cannot be analysed and is tagged *unknown*. For example, *ideologi* is an unknown root; this explains why MorphInd fails to analyse *ber-ideologi* 'have an ideology' despite having a rule for prefix *ber-*.

Many of the out-of-coverage monomorphemic words in question are specialised terminology borrowed from other languages, such as English (*delinkuensi* 'delinquence', *feudal* 'feudal', *edukasi* 'education', *ideologi* 'ideology', *epistemologi* 'epistemology') Dutch (*alkoholisme* 'alcoholism') or Arabic (*dakwah* 'preach', *jama'ah* 'followers'). Some of these words have foreign

[44] Historically, many proper nouns are polymorphemic – often compound – including *Yogya-karta* and *Yesus* from Hebrew 'Yeho-Shua'. However, I do not consider this diachronic view here and treat such nouns as monomorphemic.

morphological cues, such as *-logi* in *ideologi* ‘ideology’ or *-isme* in *alkoholisme* ‘alcoholism’. However, the orthography and in some cases the actual phonetic form of the cue, or the whole word, has been adapted to Indonesian, marking them clearly as loanwords.

By examining the list of known (in-coverage) words, I found that a handful of words with *-logi* and *-isme* were analysed correctly, including *teknologi* ‘technology’ and *sosialisme* ‘socialism’. It seems that these words must be present in MorphInd’s lexicon. This supports the hypothesis that words like *ideologi* are out-of-coverage simply because they are absent from MorphInd’s lexicon. However, not all words affected by this issue are specialised loan-terminology. Some are fairly common non-technical Indonesian words. For such words to be outside MorphInd’s coverage is surprising. Some examples are *awam* ‘common’*, majemuk* ‘various’, and *pidana* ‘crime’.

Another kind of case is exemplified by *me-mungkin-kan* ‘make possible’. The root *mungkin* ‘perhaps’ occurs as a within-coverage word, and therefore is likely to be in the lexicon. So here, there is likely a problem with the rules. Other cases of this type have as roots the in-coverage words *tidak* ‘no’, *daya* ‘power’, *guna* ‘use(n)’, and *pikir* ‘think’.

5.6 Summary

In this chapter, I have argued that MorphInd can be considered the state-of-the-art MA system for Indonesian (see 5.2), being an improvement on a prior MA that remains in continuous development and is used by many other NLP systems for Indonesian. However, the only comprehensive evaluation of MorphInd (Larasati et al. 2011) is of its first version. Moreover, that exercise *solely* measures MorphInd’s coverage. For this reason, I reassessed MorphInd.

The full results of my evaluation (section 5.5) are gathered in Table 5.17. Points 1 and 2 are comparable to Larasati et al.’s (2011) evaluation. That comparison indicates several improvements in the present version of MorphInd. The other properties evaluated here (root tagging, word tagging, and morpheme boundary assignment) were not assessed by Larasati et al.

My evaluation by these criteria thus represents a novel contribution to the field of the computational analysis of Indonesian.

	Evaluation criteria	Larasati et al. (2011) %	Present evaluation %
1	Token coverage	84.6	93.0
2	Type coverage	50.7	82.7
3	Morpheme boundary assignment accuracy	Not measured	96.6
4	Root tagging accuracy	Not measured	90.3
5	Word tagging accuracy	Not measured	89.8
6	Aggregate accuracy	Not measured	87.3

Table 5.17. Full results of the present evaluation compared to Larasati et al.'s (2011) evaluation

MorphInd's word tagging, which I have argued to be comparable to POS tagging, has 89.8% accuracy, compared to typical POS-tagger accuracy of around 95% (see 5.5.6). Root tagging and morpheme boundary assignment score higher, respectively around 93% and 96%. Aggregate accuracy across all tasks is then approximately 87%.

Having manually checked every token in a sample of MorphInd annotation allowed me to adduce evidence that MorphInd struggles, perhaps surprisingly, with monomorphemic words, most noticeably loanwords and proper nouns (see 5.5.8). In theoretical morphology, the analysis of loanwords and proper nouns may be considered marginal and of little relevance to the features of the language under study. However, an MA system that focuses on handling polymorphemic words but neglects these two types of elements will be inadequate to deal with unrestricted text, as loanwords and proper nouns are commonly present in many frequently encountered kinds of text.

In light of the above, I am confident that I have fulfilled the goal of this chapter, i.e. to perform an evaluation of MorphInd. The evaluation has underlined that MorphInd, even after several years of program development, still suffers from multiple implementation problems that hinder it in its role as a state-of-the-art MA system for Indonesian. This, along with my evaluation of MorphInd's annotation scheme (in 3.5.2), makes clear the importance of this PhD project. Given all these findings, an alternative Indonesian MA system is required, one designed to address the shortcomings identified in Morphind's MAS and implementation. Thus, the next chapter moves on to review tagging methodologies, particularly for MA systems. This review underpins my subsequent choice of system architecture for the MA system to be implemented.

CHAPTER 6

REVIEW OF TAGGING SYSTEMS: RESOURCES, FRAMEWORKS, AND TECHNIQUES

6.1 Introduction to the review

In this chapter, I review typical tagging systems, and apply the findings of this review to my planned system development. I will argue that the general principles of a tagging system apply to different levels of linguistic analysis, including the morphological, morphosyntactic and syntactic levels. The outcome of this chapter is a proposed design for my Morphological Annotation (MA) system. The review in this chapter shall justify my claim that this design is (1) optimal and, more importantly, (2) feasible to implement.

This review covers the variety of tagging resources and how they are created; the different techniques for performing automatic morphological analysis; and various frameworks to implement these techniques. The review centres primarily on specific examples of implemented tagging systems, but still requires a number of preliminaries, namely a general overview of how tagging systems work, as well as some theoretical background on the computational linguistic formalisms that underlie, particularly, MA systems. The first half of this chapter (section 6.2–6.5) covers these preliminaries.

A number of key terms for Natural Language Processing (NLP) systems in general were discussed in the Introduction (see 1.2). Among these terms, *tokenisation, annotation* and *disambiguation* are highly relevant to this chapter. In section 6.2, I will show that these three terms actually correspond to common sub-tasks of standard tagging systems (see van Halteren 1999:109).

The next topic is the prominent Two-Level Morphology (TLM) formalism (Koskenniemi 1983), already discussed briefly in section 3.2.1. TLM is a computational linguistic formalism that underlies many of today's MA systems. In sections 6.3 to 6.6, formal grammar, linguistic formalisms, computational morphology prior to TLM, and the emergence of TLM will be discussed in detail. I separately discuss linguistic (6.4) and computational formalisms (6.5), considering some key differences between them, and highlighting the challenges of incorporating linguistic (particularly morphological) expressiveness into automated systems. In section 6.6, I will show how Koskenniemi's TLM managed to meet those challenges.

The aim of the second part of this chapter is to survey existing frameworks, techniques, and resources of possible utility for my new MA system. Section 6.7 considers a number of NLP resources used by existing MA systems, and how these resources are built. It also reviews the techniques and frameworks used for the three MA sub-tasks: tokenisation, annotation, and disambiguation. In section 6.8, evaluation measures to assess the success rate of taggers are discussed.

In the final part of this chapter, sections 6.9 and 6.10, I will justify the choice of tagging technique and program to utilise in the new MA system. Section 6.11 concludes the chapter.

6.2 Typical tagging systems

A tagging system usually performs three sub-tasks in order, namely *tokenisation*, *annotation* and *disambiguation* (van Halteren 1999:109). These subtasks are typical of all tagging systems, regardless of the language or linguistic level targeted. Examples presented in this section cover POS tagging (example from English) and morphological tagging (examples from Arabic and Turkish).

Tokenisation refers to the segmentation of the raw text into analytic units. *Annotation* refers to the assignment of analytic codes, or tags, to the tokens. In some cases, tokenisation and annotation may introduce ambiguities. At the tokenisation and annotation stages, all possible analyses are usually retained. *Disambiguation* refers to the removal of contextually incorrect analyses, or the selection of the contextually correct analysis, from ambiguous tokenisation and/or annotation output. Example (6.1), reproduced from Voutilanen (1999:14), illustrates the three stages of tagging. Voutilanen's illustration is of morphosyntactic tagging of English, but the procedure is typical of all tagging systems including MAs.

(6.1)

The	ART
man	N V
is	V
an	ART
agent	N

First (tokenisation), the system segments the text into tokens (*the, man, is, an, agent*) using line breaks; no ambiguity is introduced in this case. Second (annotation), the system assigns all possible analyses to each token. One token here is ambiguously tagged: *man* gets two tags, N and V, to represent the parts of speech it can have in different contexts. Third (disambiguation), the system chooses the contextually correct analysis. In this example, V on *man* would be removed, leaving N as the only tag.

However, in some tagging systems, ambiguities are not always resolved. Example (6.2)[45] is output from the well-known Buckwalter Arabic Morphological Analyser or BAMA (Buckwalter 1999; see 3.4.2 for discussion of the tagging scheme). BAMA performs tokenisation and annotation at morpheme level.

(6.2)
```
INPUT STRING: في
LOOK-UP WORD: fy
SOLUTION 1: (fiy) [fiy_1] fiy/PREP
(GLOSS):  + in +
SOLUTION 2: (fiy~a) [fiy_1] fiy/PREP+~a/PRON_1S
(GLOSS):  + in + me
SOLUTION 3: (fiy) [fiy_2] Viy/ABBREV
(GLOSS):  + V. +
```

In example (6.2), three *solutions* (i.e. analyses) are presented. Solutions 1 and 3 tokenise *fy* as one morpheme, analysed as PREP in 1 and ABBREV in 3. In solution 2, however, *fy* is split into two tokens (*fiy* and *a*), tagged PREP and PRON_1S respectively. The three solutions thus

[45] Reproduced from the BAMA output sample available online at http://www.qamus.org/30-words.txt (last accessed 26/05/2021)

exemplify ambiguities of both tokenisation (*fy* tokenised into 1 or 2 tokens) and annotation (PREP, PREP plus PRON_1S, or ABBREV). BAMA leaves these ambiguities unresolved.

By contrast, MADAMIRA (Pasha et al. 2014) is an Arabic MA system that performs all three tasks, including disambiguation. MADAMIRA's initial (ambiguous) annotations are not generated by MADAMIRA itself, but rather, obtained from BAMA or the related Standard Arabic Morphological Analyzer (Graff et al. 2009). To this, MADAMIRA adds statistical disambiguation (see section 6.8).

Let us now consider the example of morphologically tagged Turkish data in Table 6.1. Eryiğit's (2014) ITU Turkish NLP pipeline[46] is built on top of many other systems, which he terms *modules*[47]. To perform morphological annotation, Eryiğit uses Oflazer's TLM model (Oflazer 1994), implemented in the Helsinki Finite State Transducer (Linden & Pirinnen 2009). Subsequently, a hybrid morphological disambiguator is used to discard incorrect analyses.

Tokenisation	Annotation	Disambiguation
Rahat	Noun+A3sg+Pnon+Nom	~~Noun+A3sg+Pnon+Nom~~
	Adj	Adj
et	Adj	~~Adj~~
	Verb+Pos+Imp+A2sg	Verb+Pos+Imp+A2sg
musfik	Adj	Adj
	Noun+Prop+A3sg+Pnon+Nom	~~Noun+Prop+A3sg+Pnon+Nom~~
Kenter	Noun+Prop+A3sg+Pnon+Nom	Noun+Prop+A3sg+Pnon+Nom

Table 6.1. ITU Turkish NLP Pipeline sample

Table 6.1 shows how the MA initially tokenises the input into four tokens. At the annotation stage, all tokens except *kenter* are ambiguously analysed. At the disambiguation stage, incorrect annotations are removed (indicated by strikethrough in Table 6.1), so that only the analyses most likely to be correct are presented in the output.

These examples establish the relevance, in typical approaches, of the three sub-tasks to different types of computational annotation system and to diverse languages.

[46] http://tools.nlp.itu.edu.tr/ (Last accessed 26/04/2021)

[47] In addition to a morphological analyser and disambiguator, Gulsen's system includes a *named entity recogniser* and a *syntactic parser*. It also incorporates a *normaliser* for non-standard Turkish text (such Twitter data). As these programs are not relevant to present purposes, their output is omitted from Table 6.1.

6.3 Formal languages, grammars, and automata

Three interrelated concepts are discussed in this section: formal languages, formal grammars, and automata. Their description is drawn from the following sources where not otherwise specified: Chomsky (1957), Silberztein (2016), Levelt (2008), Wintner (2013) and Linz (2001, 2012).

6.3.1 Introduction to formal languages

Computers cannot work with human language directly because they are fundamentally number-manipulating machines, and thus must deal with mathematical representations of language. For this reason, natural language elements must be mathematically defined prior to computational analysis. Defining an object mathematically is a process called *formalisation* (Silberztein 2016:7). Therefore, the linguistic objects that constitute a language (e.g. letters or words) must be formalised so that they can be handled by a computer.

The formalisation of language generates a *formal language.* Similarly, formalisation of a language's grammar yields a *formal grammar*. The use of either of these terms usually implies the other, that is, the term *formal grammar* usually occurs within the study of formal language, and use of the term *formal language* implies analysis of language in terms of formal grammar. Levelt (2008:2), a linguist, states that formal languages can serve as a mathematical model for a computer, as well as a model for developing linguistic theory. In section 6.4, I will explore how *generative grammar*, an influential theory in linguistics, models grammatical descriptions and explanations using formal grammar.

Some terms, such as *word, letter* and *alphabet*, are used in both linguistics and the computational/mathematical study of formal grammars, but defined slightly differently. The following definitions of formal grammar terminology follow those of Wintner (2013:11-13), a computational linguist. Wintner begins by defining the term *alphabet*, often symbolised by a sigma ∑. In a formal language, an alphabet is a collection of symbols, where each of the symbols is called a *letter*.

Letters are the smallest input units that the formal grammar manipulates, while *words* are the *outcome units* or *outputs* of the manipulations, as illustrated by the examples in Table 6.2. To distinguish *letters* and *words* in this discussion, words are delimited by quotation marks, whereas letters are not. Sets of letters in an alphabet, or words in a language, are enclosed in braces, with their members separated by commas. Thus, the sets {a,b,c,d}, {my, your, keys}, {the man, sees, the woman} are all considered alphabets of letters, even though in natural language {my, your, keys} and {the man, sees, the woman} would be considered words/phrases. Likewise, the sets of outputs {'my keys', 'your keys', …} and {'the man sees the man', 'the man sees the woman', 'the woman sees the man', the woman sees the woman',…} are considered words of different formal languages even though in natural language terms they are phrases or clauses.

Formal language	Letters	Words
L_A	{a,b,c,d}	{'a', 'cab', 'bad', …}
L_B	{my, your, keys}	{'my keys', 'your keys', …}
L_C	{the man, sees, the woman}	{'the man sees the man', 'the man sees the woman', 'the woman sees the man', the woman sees the woman',…}

Table 6.2. Sample letters and words of formal languages.

The alphabet of L_A consists of four letters {a, b, c, d}. Let us assume that L_A's formal grammar allows these units to be combined into at least three words {'a', 'cab', 'bad'}. That of L_B consists of three letters, *my, your* and *key*; possible output words include 'my keys' and 'your keys', but not 'a key'. This is because *a* and *key* are not letters in this language. Even though *keys* contains *key*, *key* is not a valid letter because *keys*, as a letter, is one indivisible unit. A formal language may be defined with English words, but this does not necessarily mean that they acquire all the rules and internal structure they have in English.

Each set of words in Table 6.2 ends in an ellipsis because there are other possible outputs. For instance, 'bda', 'my your', 'the man the woman' would be valid words in L_A, L_B, and L_C respectively. Each language's grammar would either generate or forbid any possible sequence of letters.

We cannot rely on our natural language intuition to verify the outputs of formal languages. A formal language may easily be defined using the elements of English as its base units and yet have very different combinatorial behaviour. Moreover, as noted already, *letters* and *words* in the formal language sense are defined differently than in the context of natural language.

One of the linguistic objects which must be formalised for computer manipulation is the language's grammar or rules. The rules of a formal grammar are mathematically expressed symbol-manipulation formulae which generate, or validate as correct, the words of the language (i.e. the sequences that *are* part of that formal language's output set). A *rule* in the formal grammar sense is different from what that term means in linguistic description, where *rule* usually refers to a characterisation by linguists of some consistent behaviour that they observe in a language under study. The nature of formal grammar rules will be addressed further in 6.3.2 and 6.3.3.

The formalisation of the letters, words and grammar of a language thus generates a mathematical model of the language. This model can be used by an NLP system to generate, or verify strings as, words of the language. For instance, a system could use a formal language L_X, with an alphabet of English morphemes and a grammar for their combination modelling the rules of English morphology, to determine that 'called' and 'calling' are valid output words whereas 'calleding' and 'callinged' are not. A different system could likewise verify that 'I ate an apple' is a valid word at the syntax level whereas 'apple I an ate' is not, using a grammar that formalises the relevant rules of English syntax.

In practice, this generation/verification is performed by a physical computer. However, we can also model the process using the notion of an *automaton* (plural: *automata*), an abstract model of a digital computer (Linz 2001:25); Levelt (2008:2) refers to automata as *accepting machines*. Either way, we conceptualise an automaton as a hypothetical machine that is presented with an input and can either accept or refuse it. Let us assume that we have an automaton for L_X. If we input 'called' into this abstract machine, it would be accepted, as it is a valid word in L_X.

6.3.2 The elements of formal grammar

There are four elements of a formal grammar. Linz (2012: 33) introduces the notation G = (V, T, S, P), for a grammar (G), *variable*[48] symbols (V), *terminal* symbols (T), *production rules*[49] (P) and a *start symbol* (S). That is, a formal grammar is defined to consist of a set of variable symbols representing sequences of letters, plus a set of terminal symbols which *are* the letters, plus a set of production rules defining how the variable symbols are composed, plus the special start symbol.

To exemplify these different elements, let us discuss a tiny formal grammar called G_D (Table 6.3), with two rules, which generates a single string 'ab', the only word of language L_D. Each rule is composed of a symbol on the left-hand side and one or more symbols on the right-hand side. The sides are separated by an arrow ($\rightarrow$) which expresses 'can be rewritten as' or 'can be composed of''. Thus, the rule $S \rightarrow aB$ is a formal expression meaning that 'S can be rewritten as a then B' or 'S is composed of A then B'.

G_D				L_D
V	T	P	S	
{S,B}	{a,b}	$S \rightarrow aB$	S	{ab}
		$B \rightarrow b$		

Table 6.3. Summary of G_D that produces L_D

Variables (conventionally represented by uppercase letters) are symbols that can be rewritten into another symbol(s), such as S (can be rewritten into aB) or B (can be rewritten into b). Among the variables in a grammar, one is the *start symbol*, the variable at the top of the hierarchy created by the grammar; all operations of the grammar are assumed to begin with a single start symbol and then to continue by applying the rules to that start symbol. Other symbols are not used as the starting point, but are introduced in the outputs of different rules. In the case of G_D, as the start symbol is S, the application of the grammar begins with the rule $S \rightarrow$ aB.

[48] Equivalent terms are *vocabulary, non-terminals* or *auxiliary*.
[49] An equivalent term *rewriting rule*.

The start symbol is often, but need not be, uppercase S. In a formal grammar for human language syntax, the S is meaningful, as the starting point for the rules is the unit of the sentence. However, the start symbol (like all symbols) is an algebraic symbol, so strictly the label is arbitrary. A formal grammar will always declare its start symbol.

In G_D, a and b exemplify the other type of symbol, *terminals*. In a formal grammar, a terminal is equivalent to a letter of the language, as defined in 6.3.1. A terminal cannot be further rewritten, and is part of the output. In G_D, symbols a and b are terminals, as they cannot be rewritten (are not on the left side of any rule) and appear directly in the output 'ab'.

The process of how the rules in G_D apply to the start symbol to ultimately generate the terminal symbols in the final output can be illustrated using a tree diagram. In section 1.1.4.1, I introduced tree diagram representation for morphological examples; the same principles apply here. In all tree diagrams that may represent a formal grammar, terminals are always at the bottom. All other symbols are variables.

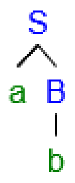


Figure 6.1. A tree diagram for formal grammar G_D

In natural language examples (such as those in Table 6.4 and Table 6.5) non-terminal symbols are given labels such as N, NP, or ROOT, devised to be meaningful for linguists (denoting noun, noun phrase, and root respectively). However, in a formal grammar, non-terminals are merely variable symbols whose labels have no effect on how the grammar works. In G_E, the terminal symbols (letters) are equivalent to natural language words (Table 6.4), while in G_F (Table 6.5), the terminals are equivalent to natural language morphemes. Regardless, the terminals are also just symbols in the formal grammar, regardless of their linguistic classification as morpheme, word, phrase, clause or sentence.

G_E				L_E
V	T	P	S	
{NP, N}	{the, cat}	NP → the N	NP	{'the cat'}
		N → cat		

Table 6.4. G_E producing L_E, representing the English noun phrase composed of words

G_F				L_F
V	T	P	S	
{W, ROOT}	{sleep, ing}	W → ROOT ing	W	{'sleep', 'sleeping'}
		W → ROOT		
		ROOT → sleep		

Table 6.5. G_F producing L_F, representing English words composed of morphemes

6.3.3 The Chomsky hierarchy

6.3.3.1 Introduction to the Chomsky hierarchy

In linguistics, it is quite common to organise grammars according to the level of linguistic unit they treat of, such as morphology (how to build words from morphemes) versus syntax (how to build phrases from words and clauses from phrases). However, regardless of the linguistic level, the *production rules* are the basis on which the type of a formal grammar is determined. The different types of grammar are organised in a hierarchy, called the Chomsky hierarchy (Chomsky 1957). The account of the hierarchy in this section draws on Chomsky (1957), Levelt (2008), Wintner (2013), and Silberztein (2016).

G_G				L_G
V	T	P	S	
{NP, D, N, P, PP}	{the, cat, in, hat, ran}	NP → D N	NP	{'the cat', 'the cat in the hat', 'the hat', 'the hat in the hat, … }
		NP → NP PP		
		PP → P NP		
		N → cat		
		N → hat		
		P → the		
		D → in		

Table 6.6. G_G that produces L_G, modelling the English noun phrase (adapted from Wintner 2013:56)

	Structure of the rule		Rules grouped by structure	
	Left hand side	Right hand side	G_D	G_G
#1	One variable	One terminal	N → cat	N → cat P → hat D → the P → in
#2	One variable	One terminal and one variable	NP → the N	-
#3	One variable	Two variables	-	NP → D N PP → PP NP NP → NP PP

Table 6.7. Comparison of production rules of G_D and G_G

Let us compare the production rules in grammar G_G (Table 6.6) with those of grammar G_D (Table 6.3 in the previous section). Table 6.7 lists the two grammars' production rules sorted by the structure of each rule. The type of each grammar can be determined by reference to restrictions on the structure of production rules, proposed by Chomsky and presented in the formulation of Silberztein (2016:120) in Table 6.8.

Grammar type		Restriction
Type 0	Unrestricted/ recursively enumerable	No restriction: any combination of variable and terminal symbols in each of the two sides is allowed (as is any Type 1, 2, or 3 rule).
Type 1	Context sensitive	Type 2 and Type 3 rules can be used. In addition, the rule may be conditioned by a context, for example the context *PLURAL* in the rule *PLURAL* SENTENCE → *PLURAL* NP see NP (PLURAL being, in this example, an abstract representation of verb agreement with a plural subject). This rewriting rule does not touch the symbol PLURAL. The context "activates" the rule and is still there after the rewriting of SENTENCE. A context can be a variable or terminal symbol and can be located before or after the main left- and right-hand side parts of the rule.
Type 2	Context free	Type 3 rules can be used. In addition, there can be rules where the left-hand side is a single variable symbol and the right-hand side is any combination of terminal and variable symbols, e.g. SENTENCE → NP sees NP, or SENTENCE → NP VP
Type 3	Regular	The left-hand side of each rule is one single variable symbol, and the right-hand side is either a single terminal symbol (e.g. NOUN → cat), or the empty string (e.g. NP → ∅), also denoted by Greek letter ϵ (epsilon), or a single terminal symbol followed by a single variable symbol (e.g. NOUN → cat ADJECTIVE).

Table 6.8. Grammar types defined via restrictions on production rules (adapted from Silberztein 2016:120)

In G_D and G_G, all rules have one variable on the left-hand side, so we can focus on the right-hand side. G_D (Table 6.3) is a *type 3* or *regular grammar*, because its production rules (in

Table 6.7) all comply with the restrictions relevant to regular grammars (type 3 in Table 6.8). Namely, the single variable on the left-hand side is rewritten on the right-hand side either as a single terminal (N → cat) or as a terminal followed by a variable (NP → the N).

Conversely, G_G (Table 6.6) is not a regular grammar. Some of its rules satisfy the restrictions of the regular grammar type. But the rules in row #3 are outside what is permitted in a regular grammar, because they have two variables on the right-hand side. However, these rules *do* satisfy the restrictions of *context free* or type 2 grammars, because context free grammars allow any combination of terminals and/or variables on the right-hand side. Thus, G_G is a context free grammar.

Seen thus, the types of grammar exemplified by G_D and G_G are not disjunct but hierarchical. The type of grammar that G_D exemplifies is a subset of the type of grammar that G_G exemplifies. This subset relationship is illustrated in Figure 6.3.

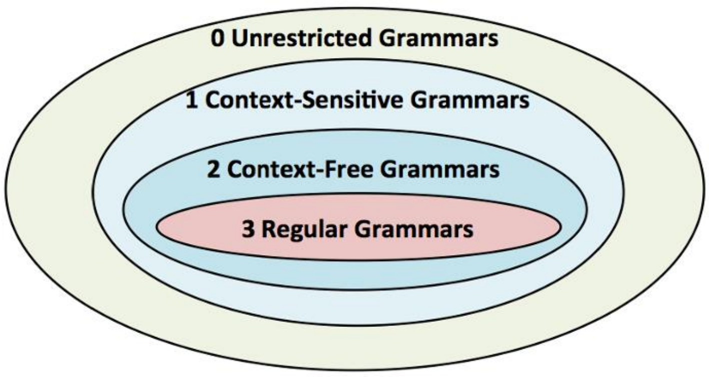


Figure 6.2. Chomsky hierarchy illustration (reproduced from Silberztein 2016:121)

Figure 6.2 illustrates the organisation of types of grammar from the least powerful type, regular, to the most powerful, unrestricted, in which no restriction is imposed on the structure of rules. The further down the hierarchy one goes, the more restrictions apply. As Table 6.8 shows, the regular grammar type has more restrictions than all other types.

Types of formal language are named after the types of grammar. Regular grammars produce regular languages; context free grammars produce context free languages; context sensitive grammars produce context sensitive languages; and unrestricted grammars produce unrestricted languages. The subset relationship also applies to the types of formal language, so that they can *also* be organised in a hierarchy, as shown in Figure 6.3.

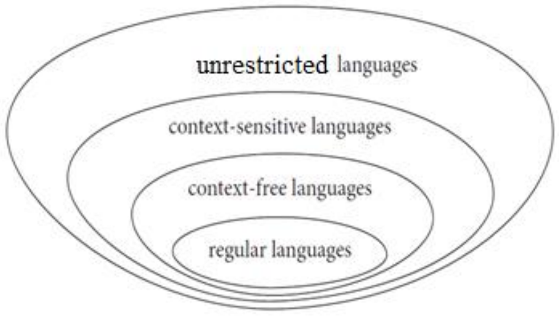


Figure 6.3. Hierarchy of formal language types[50]

In section 6.3.1, I introduced the idea of an abstract computing machine called an automaton, which can apply a formal grammar and/or verify potential formal language outputs. Each type of grammar corresponds to a different type of automaton, as listed in Table 6.9.

Grammar		Accepting machine/automaton
Type 0	Unrestricted	Turing machine (TM)
Type 1	Context sensitive	Linear bounded automaton (LBA)
Type 2	Context free	Push down automaton (PDA)
Type 3	Regular	Finite state automaton (FSA)

Table 6.9. Grammar types and corresponding accepting automata (adapted from Silberztein 2016:121-122)

Just like types of grammars, types of automata can be organised in a hierarchy. Due to this relationship, grammar types lower on the hierarchy can be processed by automata for grammar types higher up. For example, the regular grammar type corresponds to the *Finite State Automaton* (FSA). However, this type of grammar can also be processed by an automaton for any type of grammar higher in the hierarchy, that is, a *Push Down Automaton* (PDA), *Linear Bounded Automaton* (LBA) or *Turing Machine* (TM). A PDA cannot be used to process an unrestricted grammar or context sensitive grammar, but *can* process a context free grammar or regular grammar. An LBA can be used to process a context sensitive grammar, context free

[50] Adapted from https://www.tutorialspoint.com/automata_theory/chomsky_classification_of_grammars.htm (last accesed 26/05/2021)

grammar, or regular grammar, but not an unrestricted grammar. The most powerful type of automaton is the TM. It can process all types of grammar.

Power in this context refers to the capacity of each grammar type for different kinds of rules. At the high end of the hierarchy, the restrictions on rule structure are relaxed, allowing more variety of production rules and thus a more capable grammar overall. This power comes at a cost in terms of the complexity of the automaton needed. To generate or verify words of a regular language, an FSA refers to the states in its transitions (which will be explained later). To operate, the FSA only needs to remember its current state. Other automata are more complex. They must not only remember current state, but also other parameters, to process a sequence of letters. The concepts of state, transition, and other parameters are discussed in more detail later. For now, the point is that the more powerful the automaton, the more complex the parameter information it contains and uses.

FSAs implementing regular grammars serve as building blocks for computational morphology, as I will show in my review of Koskenniemi's (1983) seminal TLM, in section 6.6. Many MA programs use FSAs and regular grammars, even state-of-the-art systems. For this reason, the discussion of FSAs and regular grammars which follows will be detailed, but other types of grammar and automaton will be treated briefly, without abandoning formal rigour.

6.3.3.2 Further aspects of regular languages

The simple formal grammars introduced in section 6.3.1 are all of the regular grammar type; this introduction to regular grammar therefore will not be repeated. Rather, I will now consider other aspects of the regular languages that such grammars define.

6.3.3.2.1 Regular language operations and regular expressions

The discussion in this section is, where not otherwise specified, a synthesis of information drawn from Silberztein (2016), Linz (2012), Levelt (2008) and Wintner (2013).

Regular languages can undergo a number of operations, namely 1) *concatenation*, 2) *union* and 3) *Kleene* operation (Silberztein 2016: 120-121; Wintner 2012:13-15; Linz 2012:29-32). Table 6.10 and Table 6.11 present two regular grammars and the regular languages they generate.

G_H				L_H
V	T	P	S	
{S, B}	{a, b}	S → aB B → b	S	{'ab'}

Table 6.10. G_H that produces L_H

G_I				L_I
V	T	P	S	
{S, C}	{c, d}	S → cD D → d	S	{'cd'}

Table 6.11. G_I that produces L_I

(6.3) $L_H.L_I$ = L_J = {'ab'}.{'cd'} = {'abcd'}

Regular language operations can apply to one or more than one regular language. A *concatenation* of two regular languages chains together their outputs. For instance, concatenation of L_H and L_I produces {'abcd'}, as shown in (6.3). The concatenation result is *also* a regular language, which I label L_J. The concatenation operation is symbolised by a dot (or period/full stop) .

Being regular, the result can be described by a set of rules in a regular grammar. For instance, L_J can be generated by the regular grammar G_J in Table 6.12.

G_J				L_J
V	T	P	S	
{S, B, C, D}	{a, b, c, d}	S → aB B → bC C → cD D → d	S	{'abcd'}

Table 6.12. G_J that produces L_J

The second operation is *union*. Unlike concatenation, the union operation creates a collection of separate, or disjunct, outputs. Therefore, this operation is also known as *disjunction*.

For example, the union of L_H and L_I produces a language with two disjunct outputs {'ab', 'cd'} instead of a concatenated output {'abcd'}. The union symbol is ∪. I illustrate this operation in (6.4); an equivalent list of rules to produce these outputs is given in Table 6.13.

(6.4) $L_H \cup L_I = L_K$ = {'ab'}∪ {'cd'} = {'ab', 'cd'}

G_K				L_K
V	T	P	S	
{S, B, C, D}	{a, b, c, d}	S → aB S → cD B → b D → d	S	{'ab', 'cd'}

Table 6.13. G_K that produces L_K

The third operation is the *Kleene* operation (Wintner 2010:13), named after Michael Kleene, an influential figure in mathematics and computer science. Applied to a language, this operation outputs 1) an empty string (represented by an epsilon), 2) the outputs of the original language, and 3) an infinite set of repetitions of the original output with different lengths. For instance, applying the Kleene operation to L_H produces { ∊, 'ab', 'abab', 'abababab', …}. A Kleene operation is indicated by a star (*), as shown in (6.5) (corresponding grammar in Table 6.14.

(6.5) $L_H^* = L_L$ = { ∊, 'ab', 'abab', 'abababab', …}.

G_L				L_L
V	T	P	S	
{S, B}	{a, b, c, d}	S → ∅ S → aB B → bS B → b	S	{∊, 'ab', 'abab', 'abababab', …}

Table 6.14. G_L that produces L_L

A regular language can be represented not only by a list of rules, but also by *regular expression* notation. Regular expression is a more compact notation than a list of rules, but equally powerful as a definition of the resulting regular language (Linz 2012:77). It is common to present a formal language's outputs as a regular expression to take advantage of this compactness. Table 6.15 compares some regular expressions to the lists of regular grammar rules used to denote the same regular languages.

Regular expression	Language	Regular language	Regular grammar
abcd	L_J	{'abcd'}	S → aB B → bC C → cD D → d
ab | cd	L_K	{'ab', 'cd'}	S → aB S → cD B → b D → d
(ab)*	L_L	{∊, 'ab', 'abab', 'abababab', …}	S → ∅ S → aB B → bS B → b

Table 6.15. Comparison of regular expressions to lists of rules

A regular expression is an expression over an alphabet, augmented by special symbols (Wintner 2012:15). For instance, in the regular expression for L_K the vertical bar symbol is used to denote a disjunction, i.e. union, over two regular languages (this differs from the *operation* symbol for disjunction, which as we saw is ∪). The Kleene operation is an asterisk (or *Kleene star*) in both regular expression notation and as a regular language operator. The asterisk operator in the regular expression for L_L applies to 'ab' as one unit (grouped by brackets) to produce {∊, 'ab', 'abab', 'abababab', …}, as opposed to {∊, 'ab', 'aabb', 'aaabbb', …}, which would be the result if it was applied to *a* and *b* separately, i.e. 'a*b*'. Concatenation does not require any symbol in a regular expression; the things being concatenated are merely placed next to one another. So the regular expression for L_J 'abcd' is equivalent to the concatenation of 'ab' and 'cd'. The concatenation operator (dot) means something different in regular expressions[51].

[51] For a list of regular expression symbols and their functions, see Jurafsky (2007:22-30).

6.3.3.2.2 Finite state automata

6.3.3.2.2.1 Elements of an FSA

In addition to a list of rules and a regular expression, a regular language can be characterised by the FSA that would process it. An FSA is an automaton or accepting machine implementing a regular grammar, which can determine whether strings presented to it are or are not valid in the corresponding regular language. This machine consists of a *finite* set of *states,* represented by the symbol Q – thus the term FSA. The states are connected by a finite number of *transitions* (δ), each of which is labelled by a letter from the language's alphabet (∑). Processing a sequence of input letters involves the machine transitioning from being in one state (before the next input letter is processed) to another state (after that letter is processed).

An FSA begins operating from a *start state* (q0). For every transition from one state to another, it *prints* the label associated with that transition. The term *print* here does not mean actual printing to paper. Rather, it is part of the terminology of abstract automata, referring to the automaton emitting the letters that label the transitions it makes and accumulating the omitted letters to create the ultimate output. In FSA notation, the transition <q0,c,q1> means "when the automaton moves from q0 to q1, print c". This movement from state to state, and the printing of letters, continues until the system reaches a *final state* (or *accepting state*), at which point the accumulated output is combined into a word. Table 6.16 illustrates the elements of FSAs using an example labelled FSA_A.

Element	Nature of element	Element in FSA_A
Q	Set of states	q0,q1,q2,q3
Σ	Alphabet	{c,a,t}
q0	Start state	q0
F	Final state	q3
δ	Possible transitions	{<q0,c,q1>, <q1,a,q2>, <q2,t,q3>}

Table 6.16. Elements of FSA_A (adapted from Wintner 2010:19)

Figure 6.4 diagrams FSA$_A$, with states shown as circles. In this customary graphical representation, the start state is coloured grey, and the final state has two circles. Transitions between states are indicated by arrows; the letter printed by a transition is above its arrow. Transitions are the key element of an FSA diagram. The list of transitions shows us that we must draw lines from q0 to q1 and from q2 to q3, but not, for example, from q2 to q0 or q3 to q1. These possible transitions determine which letter sequences in the input will and will not be successfully processed (accepted) by the automaton.

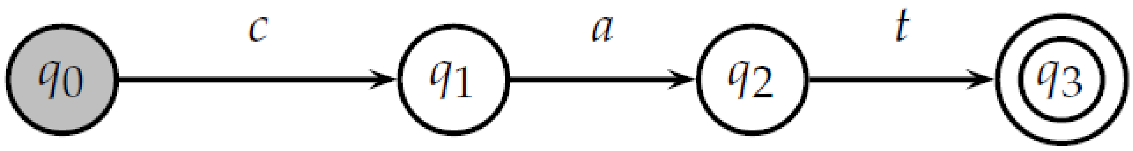


Figure 6.4. Diagram of FSA$_A$ (adapted from Wintner 2010:19)

The words accepted[52] by FSA$_A$ can be figured out by following the arrow. There is only one transition each from q0 to q1, q1 to q2, and q2 to q3. These transitions print the letters *c, a*, and *t* respectively, and therefore, this FSA accepts one word, 'cat'. Every set of words accepted by an FSA is a regular language, so the set {'cat'} is necessarily a regular language, describable via regular grammar or regular expression as well as FSA.

6.3.3.2.2.2 The epsilon move and loops in an FSA

What is dubbed the *epsilon move* in an FSA is a transition that prints no label, equivalent to a regular grammar production rule A → ∅. As noted in 6.3.3.1, the empty symbol is epsilon (ϵ). This is illustrated by FSA$_B$ in Figure 6.5, which accepts {'undone', 'done', 'undo', 'do'}. Accepting the last three words, but not the first, involves epsilon moves. To accept *do*, first *un* must be bypassed by the transition <q0,∊,q2>, which prints epsilon. The FSA then prints 'do' as it

[52] While I use the term 'accept', terms including 'produce', and 'generate' (Wintner 2010; Jurafsky 2007) are also used in the literature with the equivalent meaning of generating an output form, according to a set of rules.

traverses q2 to q3 to q4. Finally, 'ne' is bypassed by the transition <q4,∊,q6>, which prints another epsilon. The full word is thus '∊do∊', which is equal to 'do'. The other words, 'done' and 'undo', are accepted with only one epsilon move each.

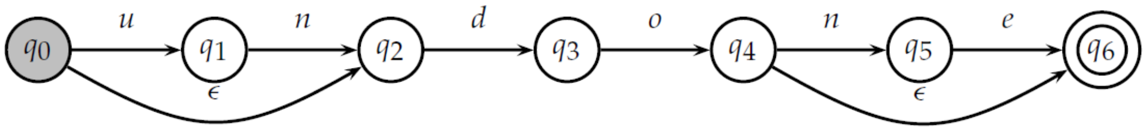


Figure 6.5. FSA$_B$ with epsilon moves (reproduced from Wintner 2010:18)

A *loop* is a transition from some state back to itself, following which allows the FSA to print the same letter over and over again. FSA$_C$ in Figure 6.6 has a transition from q2 to q2, printing letter *o* <q2,o,q2>. Thus FSA$_C$ recognises { 'meow', 'meoow', 'meooow' ...}. When it reaches state q2 in the process of accepting (or attempting to accept) an input string, the next input letter determines whether this FSA loops to q2 again (if it is *o*) or follows the transition from q2 to q3 to break out of the loop (if it is *w*).

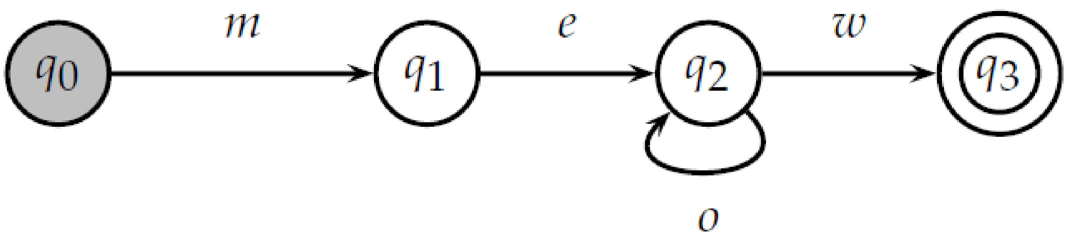


Figure 6.6. FSA$_C$ with a loop (reproduced from Wintner 2010:19)

6.3.3.2.2.3 FSAs and regular expressions

Because an FSA describes a regular language, it can implement regular language operations, for each of which there is an equivalent regular expression. Figure 6.7 presents three FSAs with equivalent regular expressions, each illustrating a regular language operation (see

6.3.3.2.1): concatenation, disjunction, and Kleene operation, respectively.

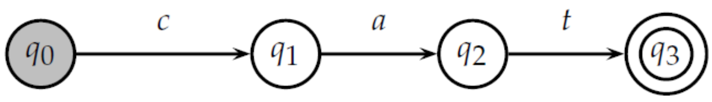
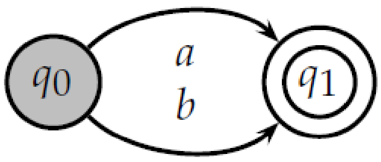
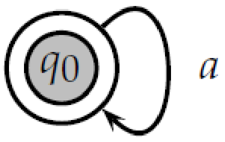
FSA (and operation exemplified)	Regular expression	Language
Concatenation 	cat	{'cat'}
Union (disjunction) 	a|b	{'a','b'}
Kleene operation 	a*	{ϵ, 'a', 'aa', 'aaa', … }

Figure 6.7. FSAs and regular expressions (adapted from Wintner 2010:18-19)

6.3.3.2.3 Regular relations and finite state transducers

The discussion in this section draws on Wintner (2010) and Jurafksy (2007).

6.3.3.2.3.1 The nature of regular relations

If two languages are regular, it is possible to *relate* them. This means that each word in one language is *paired* to a word in the second language. An FSA that accepts a regular language can either accept or reject inputs. However, when two regular languages are in this kind of regular relation, the automation has an additional power: an output in the second regular language can be used as an analysis of the output (that is, any accepted input) of the first regular language. This is the fundamental principle that underlies any automatic tagger based on formal rules and implemented as an automaton: the linking of words in one language (input tokens) to words in another language (analytic tags).

Words of L_C	Words of L_D	R_{CD}
I	PRON-1-sg	{‘I:Pron-1-sg’, ‘know:V-pres’, ‘some:DET-indef’, ‘new:ADJ’, ‘tricks:N-pl’}
know	V-pres	
some	DET-indef	
new	ADJ	
tricks	N-pl	

Table 6.17. Words and corresponding tags in a regular relation (adapted from Wintner 2010:24)

In Table 6.17, the words of L_C are the (linguistic) word types, while the words of L_D are corresponding POS tags. The inventory of the regular relation (symbolised R_{CD}) consists of words paired to tags, here shown with colon. Thus, a regular relation links two languages, but does not change either language; nor does it create a new language based on one or both. It is merely a set of pairings between their vocabularies.

6.3.3.2.3.2 Finite state transducers

FSAs can verify compliance of inputs with a regular language's grammar by accepting (and printing) valid words, but rejecting letter sequences that are not valid words. But an FSA cannot utilise a regular relation, as it is limited to one language. The type of finite state machine which can use a regular relation is called a *Finite State Transducer* (FST). An FST has all the elements of an FSA (see 6.3.3.2.2.1), but *also* a second alphabet, so that it can utilise a regular relation between two languages. FSTs' most striking difference from FSAs is in the labels on transitions (δ). Rather than one label (i.e. printed letter), there are two, one per language, joined with a colon, representing a pairing from the regular relation see Figure 6.8 to Figure 6.10.

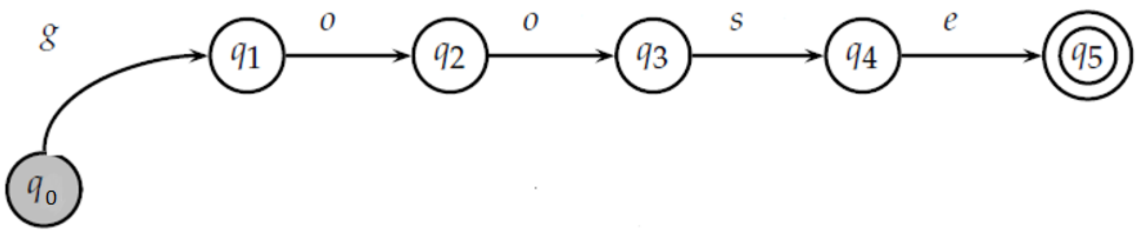


Figure 6.8. FSA that accepts a word ‘goose’ from a language of English singulars (adapted from Wintner 2010:25)

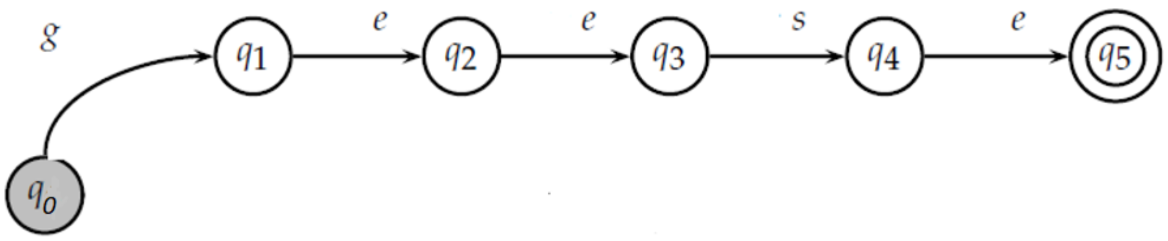


Figure 6.9. FSA that accepts a word ‘geese’ from a language of English plurals (reproduced from Wintner 2010:25)

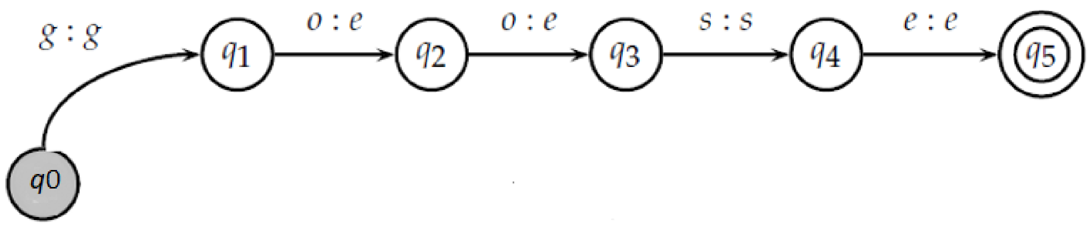


Figure 6.10. FST for the regular relation of *goose* and *geese* {‘goose:geese’} from the two languages, with all pairs of labels shown (reproduced from Wintner 2010:25)

Figure 6.8 and Figure 6.9 present FSAs from two different languages that produce *goose* (singular) and *geese* (plural) respectively. Figure 6.10 presents an FST that utilises the regular relation *between* the two languages. Each transition label has a letter from the first and the second language. For instance, the transition <q1,o:e,q2> leads from q1 to q2; *o* is some letter of the first alphabet, *e* is some letter of second alphabet, and these two letters are associated. This FST exemplifies a system which, having recognised an English singular noun, outputs the corresponding plural. In this case, it yields ‘geese’ when it recognises ‘goose’.

The letters linked within an FST transition label may be identical. For example, the first, fourth and fifth letters of *goose* and *geese* are identical (as indicated by transition labels g:g and s:s in Figure 6.10). An alternative notation gives the letter just once when it is the same in both languages; see Figure 6.11.

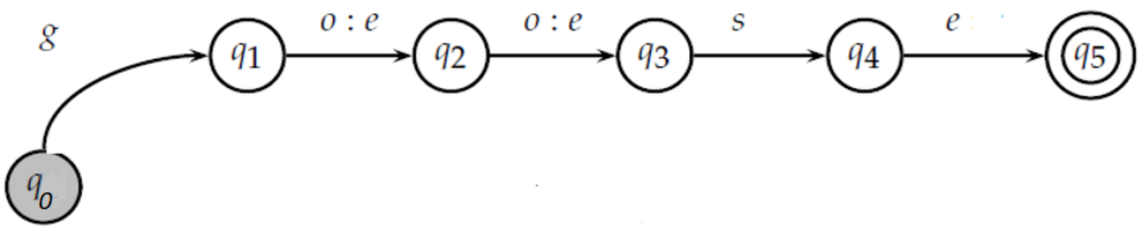


Figure 6.11. FST for the regular relation of *goose* and *geese* {'goose:geese'} using an alternative notation (reproduced from Wintner 2010:26)

The FSTs in Figure 6.10 and Figure 6.11 are equivalent. Words 'goose' and 'geese' represent the case in which the singular and plural forms have the same word length (five letters). However, English singular and plural forms may differ in length. For instance, 'ox' (singular) is two letters long, while 'oxen' (plural) is four letters long. An FST handles this by printing epsilon to omit one or more letters, as shown in Figure 6.12.

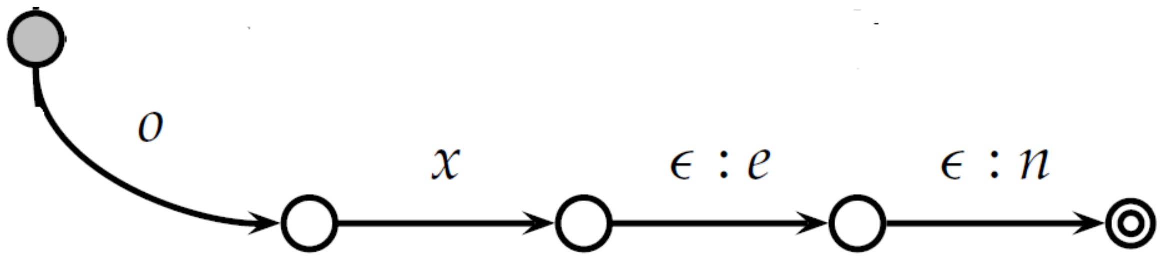
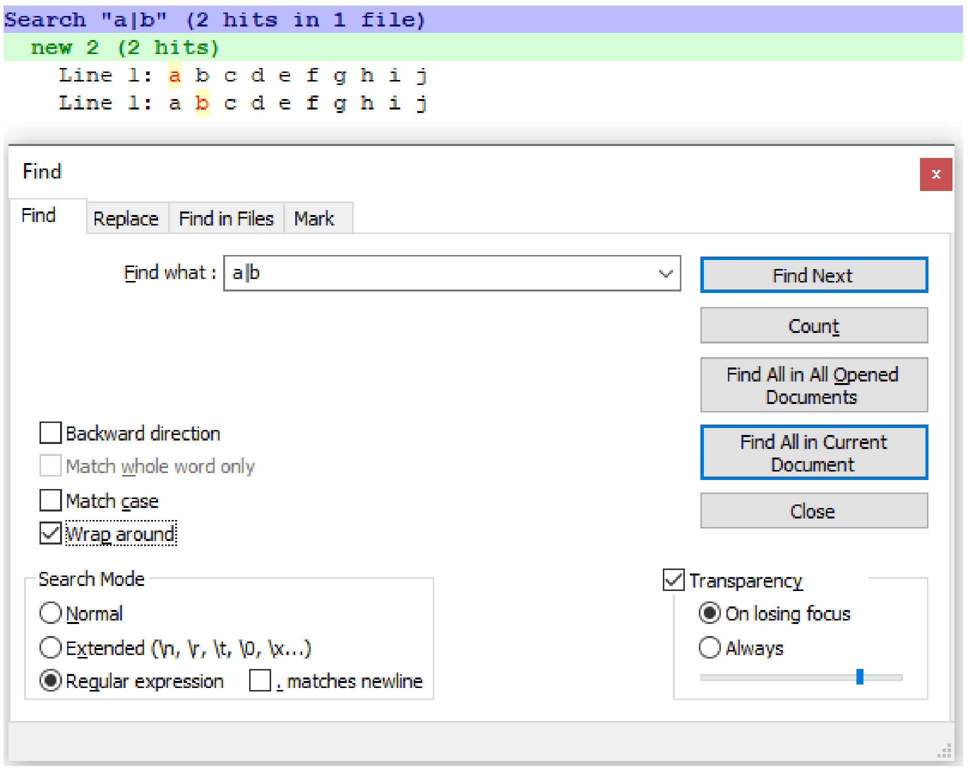


Figure 6.12. The use of epsilon to handle paired words of unequal length in an FST that pluralises 'ox' (reproduced from Wintner 2010:26)

6.3.3.2.3.3 Practical uses of FSTs

Both FSAs and FSTs are of great importance to NLP. For instance, many NLP applications allow their users to search through text – a function present even in most text editors – and this can often be done with regular expressions. When we input a regular expression into the search interface (e.g. a|b, as in Table 6.15), the computer compiles this regular expression into an FSA using a *compiler* module present within the application. The text is given as input to the compiled FSA, beginning at each possible start point. When the FSA accepts the input, a search result has been found, and the program will display it – either by showing it somewhere on screen, by highlighting it, or by moving the user's cursor to the result. Figure 6.13 illustrates all this for one popular text editor.



```
Search "a|b" (2 hits in 1 file)
  new 2 (2 hits)
    Line 1: a b c d e f g h i j
    Line 1: a b c d e f g h i j
```

Find

Find what : a|b

[✓ Wrap around]

Search Mode: [✓ Regular expression]

[✓ Transparency] [✓ On losing focus]

Figure 6.13. Results of a regular expression search in Notepad++[53] (matches highlighted)

Another application of FSTs is linguistic annotation, as previously discussed (6.3.3.2.3.2) For instance, a morphological annotation system can produce tags using FSTs compiled from regular expressions. Let us assume a tagging system whose output is laid out on two lines, where the first line presents the morpheme tokens recognised, and the second line presents the morphological tags assigned to them, as in the example in Table 6.18. Once a word is split into morphemes, we can use an FST to annotate the tokens by processing and accepting valid sequences of morphemes and producing the valid sequence of tags paired with the morphemes as its second output.

[53] https://notepad-plus-plus.org/ (last accessed 26/04/2021)

Turkish input	*uygarlastıramadıklarımızdanmıssınızcasına* "(behaving) as if you are among those whom we could not civilize"
Morphemes (1st language)	uygar +las +tır +ama +dık +lar +ımız +dan +mıs¸ +sınız +casına
Tags (2nd language)	civilized +BEC +CAUS +NABL +PART +PL +P1PL +ABL +PAST +2PL +AsIf

Table 6.18. Turkish morphemes and corresponding tags from a regular relation (adapted from Jurafsky 2007:52)

6.3.3.3 Context free grammars (CFGs)

Having dealt at length with regular grammars, let us now look more briefly at the other types of grammar in the Chomsky hierarchy. This account of CFGs is drawn from Levelt (2008) and Silberztein (2016). The restrictions on a CFG's production rules are more relaxed than those that apply to regular grammars. The right-hand side of a CFG's production rules can consist of any combination of terminals and variables as per the definition in section 6.3.3.1. One example CFG commonly seen in the literature is shown in Table 6.19. It generates (or accepts) only words which consist of a string of 'a' followed by a string of 'b' where the two strings are exactly equal in length (Silberztein 2016:165-166).

G_M				L_M
V	T	P	S	
{S}	{a, b}	S → aSb S → ab	S	{ab, aabb, aaabbb, aaaabbbb,…}

Table 6.19. G_M that produces L_M

The CFG for L_M contains a self-referential rule (S → aSb). It is thus able to process recursion. Recursion is the only way the grammar can 'remember' how many *a*'s have been produced in order to produce the same number of *b*'s. A regular grammar cannot have a rule like S → aSb, and thus has no memory and cannot process recursion.

A natural language example of this phenomenon is Prepositional Phrase (PP) recursion, as exemplified in Table 6.20. Since an NP can include a PP and a PP can include an NP, Noun Phrases may have one PP ('the <u>pen in the box'</u>), two PPs ('the <u>pen in the box</u> <u>in the drawer</u>'), three PPs ('the pen <u>in the box</u> <u>in the drawer</u> <u>under the desk</u>') and so on theoretically *ad infinitum*.

G_N				L_N
V	T	P	S	
{NP, PP, DET, N, P}	{the, pen, in, box, drawer, under, desk}	NP → DET N PP NP → DET N PP → P NP	NP	{ 'the pen', 'the pen in the box', 'the pen in the box in the drawer', 'the pen in the box in the drawer under the desk'…}

Table 6.20. G_N that produces L_N illustrating PP recursion

A CFG can also capture ambiguity, which is important for syntactic annotation. The sentence *this man sees a chair from his house* is syntactically ambiguous in that it could be interpreted as (1) the chair used to be in his house and the man sees it now somewhere else; or (2) the man is in his house and sees a chair that is not in the house and never has been. The CFG in Table 6.21 can represent both interpretations as the outcomes of two different subsets of its production rules given that sentence as input. The first interpretation begins with the production rule S → NP V NP, while the second begins with S → NP V NP PP. The linguistic phenomenon simulated is termed ambiguity of *PP attachment*.

G_O				L_O
V	T	P (in two subsets)	S	
{S, V, NP, DET, N, P}	{this, man, sees, a, chair, from, his, house}	S → NP V NP NP → DET N NP → DET N PP PP → P NP DET → this DET → a DET → his N → man N → chair N → house V → sees P → from	S	{ 'this man sees a chair from his house', …}
		S → NP V NP PP PP → P NPNP → DET N DET → this DET → a DET → his N → man N → chair N → house V → sees P → from		

Table 6.21. G_O that produces L_O, able to process PP attachment ambiguity

The corresponding automata for CFGs are Push Down Automata (PDA). A PDA must be able to 'remember' a previous state so that it can return to it after implementing a rewriting rule, to complete earlier production rules left unfinished. Let us illustrate this with a simple rule S → NP VP. This rule needs to be suspended, its processing incomplete, as first one right-hand side non-terminal, then the other, is itself fully processed; only then is the initial rule completely processed. Thus, to process NP, the subsequent variable VP must be 'pushed' into a 'stack'. Once the PDA has finished processing NP, the VP will be 'popped' out from the stack, and itself processed further. Once variable VP is completely handled, there is nothing more to pop, as no symbol follows VP in this example, so the rule for S is complete. However, had there been another symbol following VP, the push and pop process would have applied once more.

CFGs are insufficient for certain phenomena in natural language. The homorganic nasal assimilation involved in the allomorphy of prefix *meN-* in Indonesian (discussed at length in 2.1.2) serves to illustrate this. When *meN-* precedes a base that starts with /p/, the *N* becomes /m/ (allomorph *mem-*). This can be expressed by the rule N p → m p. A CFG cannot capture this phenomenon, as the rule includes a *context*: an adjacent symbol (here, p) which specifies whether or not the rule will actually rewrite any given instance of the left-hand side variable symbol (here, N). Such phenomena require a Context Sensitive Grammar (CSG).

6.3.3.4 Context sensitive grammars (CSGs)

This description of Context Sensitive Grammars (CSGs) draws on Silberztein (2016) and Levelt (2008). A CSG is more powerful than CFG because it may specify a context around the symbol on a rule's left-hand side, and/or around the symbols on the right-hand side. Such rules are impossible in a CFG due to the restriction that the left-hand side can only consist of a single variable.

G_P				L_P
V	T	P	S	
{S, N}	{m, ng, n}	S → N N p → m p N k → ng k N t → n t	S	{ 'm', 'ng', 'n'}

Table 6.22. G_P that produces L_P in which N is conditionally rewritten

The rules in Table 6.22 represent part of Indonesian homorganic nasal assimilation allomorphy, namely the rewriting which determines the place of articulation of the prefix-final nasal. The three rules for variable N specify three contexts (*p, k* and *t*) on both sides of the rule, following N and following the symbol to which the rule rewrites N (*m, ng,* or *n*). Therefore, the rewriting of N is *conditioned* by its context.

An automaton with the power to execute rules conditionally is called a Linear Bounded Automaton (LBA). While PDAs require a stack for context free rules, the LBA requires a 'tape' in order to implement the more powerful conditional (context sensitive) rules. A tape allows access to the preceding and following material in the automaton's input: at any point, the LBA can 'see' what is to the left and right. This implements contexts. In the example in Table 6.22, when an N is being processed, the nature of the material adjacent to N on the tape determines whether N will be rewritten into *m, ng,* or *k*. In this example, the rule is conditioned by a right context. However, it is also possible for an LBA to make use of a left context, or both. Full details on LBAs would not be directly relevant here, but for further background the reader is referred to Levelt (2008:85-96).

6.3.3.5 Unrestricted grammars

This account of unrestricted grammars draws on Silberztein (2016) and Levelt (2008). Unrestricted (or *recursively enumerable*) grammars are the most powerful type in the Chomsky hierarchy, since as the name indicates, there are no restrictions on the rules. However, this type of grammar is of little interest from the linguistic point of view, as definitionally its procedures may be completely arbitrary (Levelt 2008:9). Both sides of the production rules may consist of any combination of variables and terminals. Finding natural language examples which can only be characterised by this type of grammar is difficult, precisely because natural language grammars tend to be patterned and not to exhibit such arbitrariness. The automaton that corresponds to an unrestricted grammar is a Turing Machine (TM), named after noted early computer scientist Alan Turing.

TMs model the abilities of an actual computer or programming language. A computer may be programmed to read and write arbitrary memory locations, and to follow paths among states defined, likewise, arbitrarily. Equivalent behaviour is not found in natural language. The storage of a TM, like that of an LBA, is conceptualised as a tape. However, while a PDA can only examine the top of its stack, and an LBA can only look immediately ahead or immediately back on its tape, a TM is not limited to any defined context. Its tape can be moved left or right, and the content of the tape at *any* location 'seen', without any restriction. The literature on TMs is vast (see Levelt 2008:108-121), but will not be discussed further here; as Levelt (2008:9) observes, natural languages are *not* unrestricted, so the TM is not a good model for natural language. Other types of automaton are thus theoretically preferable.

6.4 Linguistic and morphological formalisms

6.4.1 Introduction to linguistic formalisms

Linguistic formalisms are systems which apply formal grammar to the description of natural language. All linguistic formalisms are formal grammars (as defined in 6.3). However, modified and compact notations specifically tailored to natural language are often used in linguistic formalisms. These notations are not commonly used in other formal grammars (i.e. those that describe non-natural language).

One prominent theory centred on formal grammars is *generative grammar,* pioneered by Chomsky (1957; 1965; 1968). Generative grammarians develop formalisms by proposing rules that govern the structures either of particular languages or natural languages in general (the latter being termed *universal grammar*).

Although the history of generative grammar runs from the 1950s to the present day, the account of this approach – and particularly of generative morphology — given here addresses mainly the period through to the early 1980s. The generative morphology of that period is the model that computational linguists working on morphological analysis utilised prior to Koskienniemi's (1983) invention of the Two-Level Morphology (TLM) formalism. Today's generative morphology incorporates novel advances such as Optimality Theory (Prince and Smolensky 2004; Kager 2004; McCarthy 2002). However, such developments have had, as will be shown, little impact on computational morphology due to the success of TLM. I also restrict the discussion to three particular matters that will prove relevant to computational morphology (in 6.4.2 to 6.4.4). Generative grammar has also contributed to the development of fields such as cognitive linguistics and psycholinguistics, but I will pass over this as irrelevant for present purposes.

Generative morphology became an area of study in the early 1970's – later than generative syntax or phonology, which were by then well-established. Important figures in generative morphology include Morris Halle and Mark Aronoff. Halle, with Chomsky, pioneered generative phonology (Chomsky & Halle 1968), but to begin with, generative morphology was understudied. Most generative grammarians argued that it was not needed – it could be covered by syntax and phonology (Spencer 1991:63). To substantiate the importance of generative morphology, separate from phonology and syntax, Halle (1973) developed Word Formation Rules (WFRs). Aronoff (1976) responded to Halle's work by proposing refined WFRs, with a similar formal notation but slightly different conceptual framework (see Spencer 1991:73-92). Both versions of WFRs work within generative grammar, and thus differ little in terms of formalism (see 6.4.3). Other landmark works in generative morphology in this period include Siegel (1979), Selkirk (1983), and Scalise (1986).

6.4.2 Transformations and underlying and surface forms

A central concept of generative grammar is the *transformation*. This term predates generative grammar, having been used by structuralist linguist Zelig Harris, of whom Chomsky was a student, to refer to the alteration of one observable linguistic form to another (Barsky 2011:132).

Chomsky (1965) uses the term *transformation* in another sense. In generative grammar, particularly syntax, a transformation alters what Chomsky dubs *Deep Structure* (DS) to *Surface Structure* (SS). The respective equivalent terms *Underlying Form* (UF) and *Surface Form* (SF) are normally preferred in phonology and morphology. The SF is the phonetic or orthographic realisation of a linguistic unit that is observable in writing or speech; but the UF is defined by Chomsky (2015:145) as the abstract underlying grammatical structures and functions that make up the framework of a sentence into which lexical items are inserted. So, for instance, in Indonesian *orang-orang* 'people', pluralised by reduplication from *orang* 'person', the SF is clearly *orang-orang*. But the UF consists of the root's UF *plus* the underlying abstract representation of the morphological operation of reduplicating a noun to mark plurality. We could represent this UF in different ways according to the formalism; one example might be *<RED.Full>*. The relevant transformations then consist of the formal rules needed to generate SF *orang-orang* from UF *<orang><RED.Full>*.

6.4.3 Word formation rules

Halle's and Aronoff's *Word Formation Rules* (WFR) framework, introduced in 6.4.1, is a morphological formalism based ultimately in formal grammar. WFRs are rules that describe how a base concatenates with one or more other morphemes to form a valid word (Aronoff 1976:36). To capture linguistic features relevant to word formation, the WFR formalism adds labels and symbols to basic rewrite rules, as exemplified by the WFRs in Table 6.23.

WFR 1	WFR 2
$\emptyset_{V}$ $\rightarrow$ +ion $_{N}$ [+transitive]	$d \rightarrow s/n\_\begin{Bmatrix}+ive\\+ion\\+abl\end{Bmatrix}$

Table 6.23. Sample WFRs (adapted from Aronoff 1976:36)

WFR 1's left-hand side contains epsilon (written as ∅) with subscript V, indicating an absence (no suffix) at the end of some base which must be a verb. The conditioning of the base as a verb is called a *contextual condition* in linguistics (in formal grammar terms, this is an implementation of a CSG rule; see 6.3.3.4). The [+…] below this symbol is generative grammar notation indicating a value of some feature. Thus, [+transitive] under the epsilon here indicates a contextual requirement for the verb base to be transitive.

WFR 1's right-hand side is composed of suffix *-ion* with subscript N for noun. The latter is *output information* indicating that the output of this suffixation is a noun. Altogether, WFR 1 formalises the generalisation that suffix *-ion* can be directly concatenated to a transitive verb base to derive a noun; e.g. UF *infect* + *ion* becomes the valid SF *infection*.

Like other generative formalisms, the WFR model often involves *ordered series* of transformations, that is, multiple transformations which must be applied in a particular order to ensure a well-formed result. This is because (1) the operation of one rule may create or remove the contextual conditions that trigger some other rule, and (2) the order of transformations determines the linear order of morphemes in the resulting word. Both points can be illustrated by considering the application of WFR 1 and WFR 2 to base *extend*.

It takes both WFRs to transform 'extend' to the well-formed noun *extension*. First, WFR 1 applies to generate intermediate form **extendion* (direct concatenation of suffix *-ion* to the base), which is not yet well-formed. WFR 2 applies second, and operates on the intermediate form. This allomorphy rule alters *d* to *s* in any context where *d* is preceded by *n* and followed by *-ion* (or other suffix triggering this rule); the conditioning context is given after the slash /, and the underscore indicates the position of the 'd' relative to that context. The intermediate form *exte<u>n</u>d<u>ion</u>* meets both conditions, and so WFR 2 alters *d* to *s*, transforming *extendion* to *extension*, a well-formed word.

If WFR 2 came first, it would not be triggered, as its context (a following *-ion*) is not fulfilled. Application of WFR 1 would then generate *extendion*, but since WFR 2 has already operated, this form could not be transformed further. Thus, one sequence of these two rule generates the true English form, whereas the reverse order cannot.

6.4.4 Overgeneration and blocking

Two other pertinent concepts in WFR are overgeneration and blocking. *Overgeneration* is a term used by Halle (1973) to refer to anomalous surface forms generated by a language's WFRs. In theory, all WFRs ought to apply whenever their contextual conditions are met, generating well-formed words; but in practice, some of the words thus generated are not well-formed (they do not actually exist in the vocabulary of the language). For example, the WFR for suffix *-tion* applies to verb bases, generating e.g. *derivation*. It ought, therefore, to apply to the base *arrive*. However, the result is the non-existent word **arrivation.* Meanwhile, the WFR for suffix *-al* accurately generates *arrival* from the same base. In Halle's terminology, the incorrect form **arrivation* is overgenerated.

Why do overgenerated words not, in fact, exist in the language? WFR explains this with the concept of *blocking*: a word generated by the rules is blocked from entering the vocabulary if another word already exists with the same function and meaning. In this case, *arrival* and **arrivation* would have the same meaning (nominalisation of *arrive*), so the existence of **arrivation* is blocked by the prior existence of *arrival*; only *arrival* is a well-formed SF. Another example is the blocking of the non-existent form *childs* (formed by adding *-s* to base *child*) by existing, irregular form *children.*

6.5 Computational morphology prior to TLM

6.5.1 Early computational morphology

The content of this section draws extensively on the historical accounts of Koskenniemi (1983), Roark & Sproat (2001), and Jurafsky & Martin (2007). It explores how the WFR formalism (discussed in 6.4) was first implemented in real computers, and the non-WFR-based developments in computational morphology that preceded this step.

For a formal grammar of any kind to be implemented practically, it needs to be 'translated' into *machine code,* a block of non-textual data which the computer can execute as a program. The proper computing term for this 'translation' process is *compilation*. The compiled machine code is what generates the actual output of a tagger based on a formal grammar. Compilation is performed by a program called a *compiler*.

Let us put this in the context of WFRs. WFRs all have the generic form $\alpha \rightarrow \beta / \gamma \ \_ \ \delta$ (*alpha* changes to *beta* if it is preceded by *gamma* and followed by *delta*). In a WFR, each of the four elements can be defined in terms either of their form or of category/feature labels, as outlined in section 6.4.3. Because it involves contextual conditions, a WFR is a CSG rule in terms of the Chomsky hierarchy (see 6.3). The type of automaton corresponding to CSG is the LBA. Therefore, a computer program implementing the abstract LBA can put the WFR into practice.

However, as we will see, the highly influential MA system of Koskenniemi (1983) in practice does not use an LBA. To understand why, let us consider earlier work on MAs.

Before Koskenniemi's MA, a number of NLP tools, namely stemmers and lemmatisers, were developed. Concerning these tools, we can ask at least two questions. First, what kind of morphology (sub-)task did each program perform? And second, is the program based on the aforementioned model of compiling rules (in WFR or some other formalism) to machine code, or not?

6.5.2 Stemmers

Jurafksy & Martin (2007:3) introduce the stemmer as a tool widely used in Information Retrieval (IR), defining *stemming* as stripping off 'ending(s)' from a word form so that only its stem remains; *ending* is not used in the technical linguistic sense in this context. Table 6.24 shows how two early stemmers, those of Lovins (1968) and Porter (1980), handle the word *variations*.

Input	Output	
variations	Lovins (1968)	Porter (1980)
	vari	*variat*

Table 6.24. Sample output from the Lovins and Porter stemmers

As a linguist, I expect the stem output for *variations* to be *variation*, its uninflected form. However, both stemmers under consideration remove much more material than just inflectional affixes. The task of a stemmer, then, is to render words into reduced forms so that IR systems can treat alike all related forms, regardless of whether the relationship is through inflection or derivation.

To determine whether these early stemmers use any morphological formalism, we need to review how they work. The three I discuss here (Lovins 1968; Dawson 1974; Porter 1980) utilise the 'table lookup' method (King 1961, cited in O'Halloran & Waite 1966:248). Such a stemmer identifies an ending that matches the final part of some token by searching through a table in memory which stores all the endings of which the stemmer is aware. The matched part of the word, if any, is removed.

Lovins' (1968) stemmer has 294 endings. Her stemming algorithm is based on the longest matching principle. If a word matches more than one ending, the stemmer removes the longest (e.g. for *absorptions,* both *-s* and *-tions* match, so the longer *–tions* ending is removed). Lovins subsequently applies recoding rules (Lovins 1968:23) to convert the resulting stems into so-called neutral forms. These rules are more rudimentary than WFRs, or as Koskenniemi (1983:13) puts it, they are *crude* rules. For Lovins, the neutral form is that from which any effect of root allomorphy is removed (e.g. for *absorptions*, the *absorb-/absorp-* alternation, inherited from Latin). Table 6.25 illustrates how a rule is applied to convert the *rpt* in *absorpt* (after *-ion* has been stripped) into *rb*. The final stem is then *absorb*, matching what would be produced from *absorbs* (also exemplified) or any other word based on the other root allomorph.

	Input			rpt > rb	Output
1	absorbs	-s	absorb	-	absorb
2	absorption	-tion	absorpt	absorb	absorb

Table 6.25. Recoding of 'rpt' to 'rb' (Lovins 1968:26)

Dawson's (1974) stemmer does not use recoding, but rather utilises 1,200 endings, more than Lovins uses. This stemmer improves the speed of lookup relative to Lovins' stemmer by organising the endings in memory as a set of branched character trees (Paice 1990). This strategy produces a significant performance improvement over Lovins' stemmer.

Porter's (1980) stemmer can be considered an advancement on the Lovins and Dawson stemmers in terms of avoiding words being over-stemmed (e.g. stemming *ring* > *r* because ending *ing* is known to the stemmer). To achieve this, Porter's algorithm does not strip the endings in just one pass, but instead operates across five stages with a different lookup table at each stage; Lovins' and Dawson's stemmers used just one table for all endings.

We see, then, that early stemmers utilise both rules which recode, i.e. transform, words, and a lexicon of endings roughly equivalent to suffixes. However, the rules are largely based on orthography, not formal morphology as in WFRs. Stemmer rules, referring only to concrete form, are clearly not as complex as WFRs, which can refer to theoretical categories – such as word or root POS or transitivity – of which stemmers are unaware.

None of Lovins (1968), Dawson (1974) or Porter (1980) provides any explicit discussion of the formalism their stemmer uses. However, Jurafksy & Martin (2007:74-25) argue that the cascaded rules used in the Porter stemmer can be modelled with an FST (and are thus provably equivalent to a regular grammar). Be that as it may, the early stemmers were not originally based on formal morphology, but on orthography, with only crude or rudimentary rules, if any.

While the stems identified by stemmers are not always linguistically accurate, these tools continue to be widely accepted in the field of NLP for their practical utility in IR. However, for a working MA system which respects linguistic concepts of stem, root, affix and so on, what a stemmer does is insufficient.

6.5.3 Lemmatisers

Lemmatisers are programs which supply a word's lemma – also a (sub-)task for morphological analysis. When lemmatised, a word is converted into its non-inflected form. This means that derivational morphemes must be preserved (e.g. *absorptions* lemmatises to *absorption,* not *absorb*), whereas in stemming, typically both inflectional and derivational affixes are stripped.

The lemma and stem of a word may be identical (e.g. both lemma and stem of *speaks* are *speak*) but may also differ. For instance, to stem *taking* to *tak* is acceptable, but a lemmatiser must annotate *taking* as *take*. Furthermore, words inflected by suppletion such as *better* must be analysed correctly, in this case, as part of the lemma *good*.

Like stemmers, lemmatisers are likely to be equipped with rules. But unlike stemmers, most lemmatisers rely heavily on a lexicon. A lemmatiser's lexicon must link word forms to corresponding lemmas, potentially also taking account of POS tags. Thus, given an input word form, the system looks it up in the lexicon and returns the corresponding lemma. The lexicon used in a lemmatiser is likely to include entries for words formed through both regular and irregular processes. This allows the lemmas of suppletive word forms to be returned despite being very different in shape (e.g. *went* to *go*; *better* to *good*).

Of course, many word forms will be absent from the lemmatiser's lexicon. To such forms, a lemmatiser applies rules to generate a best guess. If this fails, the last resort is returning the word form as its own lemma.

Few lemmatisers were developed in the 1970s and early 1980s. Hellberg (1972) creates a lemmatiser lexicon by reworking a corpus wordlist and adding lemmas to word forms. The same method is used to lemmatise French by Meunier et al. (1976). In some later lemmatisers, e.g. Krause and Wille's (1981), the system can make reference to a word's morphosyntactic context (its POS, and that of nearby words) to determine the correct lemma for an ambiguous form. So, for instance, the LDVLIB lemmatiser (Drewek & Erni 1982) uses a lexicon whose entries include inflectional features such as gender, number, and person.

Do these lemmatisers utilise WFR or any other linguistic formalism? None of the cited authors say explicitly that they do. Given the points above, I suspect that like stemmers, these lemmatisers use rules of a simpler kind than WFRs. And, as with stemmers, the lemmatiser rules exemplified in the literature are mostly based on orthography, not formal morphological criteria – *crude* rules in the sense discussed previously.

6.5.4 The origins of TLM

Prior to the 1980s, most computational linguistic systems were 'toys' (Roark & Sprout 2001:112). A 'toy' system is one that can handle only a limited number of linguistic inputs (as we may see in demo software today). Prior to Koskenniemi, no researcher had managed to fully implement generative morphology into a working MA system able to handle unrestricted text. What issues did work to implement morphological formalism face?

There are two separate tasks in implementing a formalism as an actual program: 1) writing the rules in some generative grammar formalism; 2) compiling the rules into machine code that the computer can execute to implement the grammar. Ideally, then, an implementation of WFR would be able to represent in its formalism, and utilise in its compiled machine code, all of WFR's complexities. But when the grammar is complex, the design of the compiler is more challenging. Moreover, the resulting machine code has greater resource requirements (in terms of memory and processing time). A complex grammar, such as a large set of WFRs, could easily be compiled to code that requires more resources to run than were available in most computers of the 1970s and early 1980s. Therefore, at that time it was preferable for an MA to be based on the simplest possible grammar, rather than a grammar complex enough to encode any possible WFR.

Complexity in the Chomsky hierarchy	Grammar type => automaton type
Bottom of hierarchy – simplest grammars	Regular => FSA
	Context free => PDA
	Context sensitive = > LBA
Top of hierarchy – most complex grammars	Unrestricted => TM

Table 6.26. The complexity of grammars in the Chomsky hierarchy (adapted from Silberztein 2016:121-122).

As mentioned in 6.5.1, WFRs allow reference to contextual conditions, and therefore, make up a CSG, the second most complex type in the Chomsky hierarchy. But the least complex formal grammar, and thus the type preferable for computer processing, is the regular grammar. So how did early researchers model the grammar so as to produce machine code lightweight enough to execute on 1970s/1980s computers?

The solution to this problem came gradually, and not directly from morphology (Kartunnen 1993). The generative phonologist Johnson (1972), drawing on earlier theoretical work by Schutzenberger (1961), argues that the contextual conditions in phonological rewrite rules can be modelled by FSMs. This is because these rules are *not* applied recursively to their own output. In generative phonology's use of the generic CSG rule α→β/γ _ δ, once α→β is completed, the same rule is never applied again to β. While CSG (and CFG) allows such recursion, formalist phonology does not use it. Consequently, Johnson argues, the input-output pairs of a phonological CSG-type grammar behave like a regular relation, which can be modelled by FSTs (Kartunnen 1993: 183). Thus, generative phonology does not need to be implemented as a complex LBA; a much simpler FSM is enough.

Kaplan & Kay (1981) were unaware of Johnson's proposal, but made a proposal parallel to Johnson's theoretical insight: that rules with contextual conditions can be implemented with FSMs rather than LBAs. Their project aimed to build a compiler for generative phonology rules. Kaplan and Kay explore the practical implication of what Johnson briefly discussed, that is, the nature of sequential transformation rules in relation to FSMs. Thus, they implement their phonological grammar formalism using an FSM.

Kaplan and Kay's approach compiles a sequence of phonological rules into a single large FSM (Figure 6.14). As Karttunen (1993:181) observes, this approach does not need any intermediate forms as typically required in generative grammar. This is because any system of ordered phonological rules applying in sequence *also* describes a regular relation, regardless of how many rules are involved. The single large FSM only recognises two levels: underlying level and surface level (this two-level approach was inherited, and significantly improved on, by Koskenniemi 1983:14, as he acknowledges). This transducer would theoretically be able to generate an underlying-to-surface form lexicon which contains all underlying forms and their corresponding surface forms.

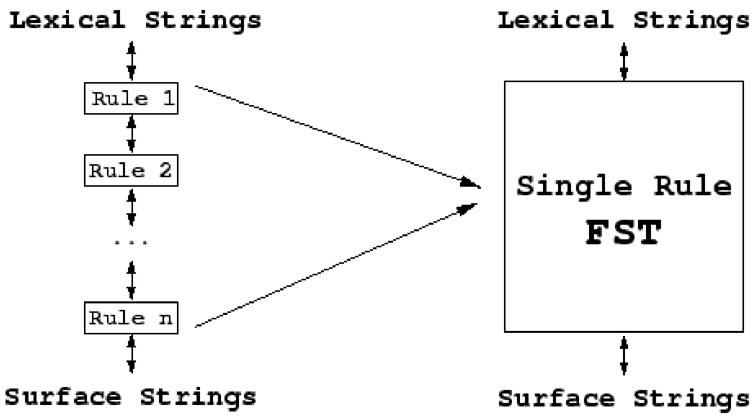


Figure 6.14. The composition of sequential rules into a single rule FST (reproduced from Kartunnen 1993)

In sum, the system Kaplan & Kay (1981) suggest can be described as follows. Phonological rules written in some generative grammar formalism are compiled and combined into a single large transducer. This single transducer is applied to the underlying forms, generating the corresponding surface forms. Paired underlying and surface forms are then collected into a full lexicon.

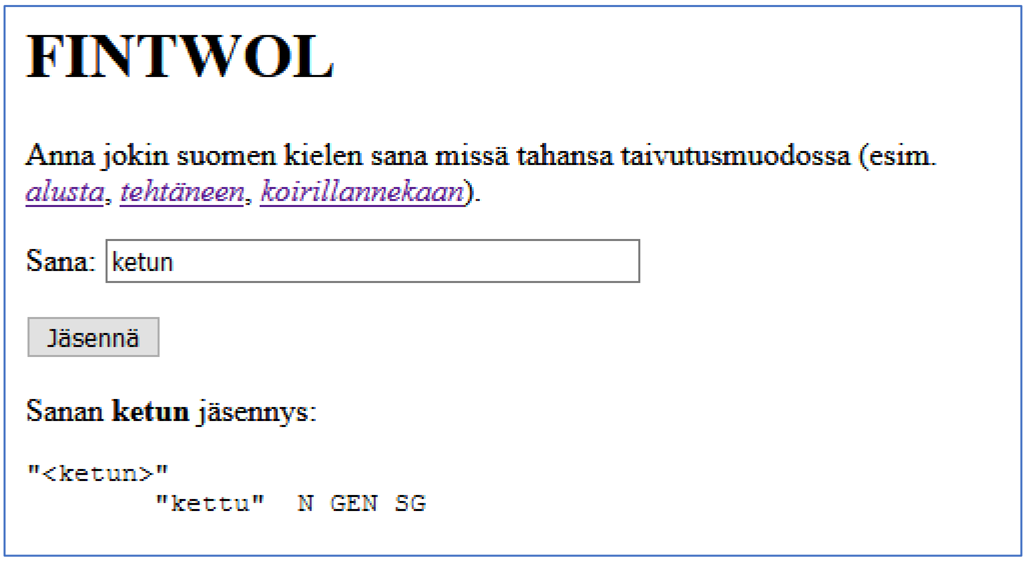
Kaplan and Kay's effort to implement this plan was successful in compiling a complete formal grammar of the phonological rules of Finnish into a single large transducer. However, their system failed to run when applied to underlying forms in the lexicon, because a lot of memory is required to execute such large FSTs, and in 1980, computer memory was limited. Only later would Koskenniemi (1983) find the solution to this problem.

What is important from Johnson's and Kaplan and Kay's (and later Koskenniemi's) findings is (1) that computational phonology and morphology based on generative theory has been able to presume that FSM-based systems will be able to implement the rules used in these fields; and thus (2) that there is no need to implement the rules using more complex automata. This remains true even as of 2021; the majority of MA systems today are based on FSMs (see further 6.7).

6.6 Two-Level Morphology

6.6.1 An overview of Koskenniemi's system

The overview of TLM given in this section draws extensively on Koskenniemi (1983). I focus on Koskenniemi's formalism rather than his software, as the TLM formalism remains in current use despite many developments in the implementation. Since the inception of Koskenniemi's system in 1983, a number of well-known FSM-based programs which adhere to the TLM formalism have been devised, including PC-Kimmo (Antworth, 1990), Fintwol (Koskenniemi 1995), xfst (Kartunnen & Beesley 2003), and foma (Hulden 2009). The TLM formalism guides the creation of the core lexicon for any MA system based on Koskenniemi's approach. That core lexicon determines what output is assigned for each input. The user interface and output format of one current TLM-based system, Fintwol, is illustrated in Figure 6.15.


FINTWOL

Anna jokin suomen kielen sana missä tahansa taivutusmuodossa (esim. alusta, tehtäneen, koirillannekaan).

Sana: ketun

Sanan **ketun** jäsennys:

```
"<ketun>"
        "kettu"  N GEN SG
```

Figure 6.15. Analysis of *ketun* by Fintwol (http://www2.lingsoft.fi/cgi-bin/fintwol)

To summarise Fintwol's operation concisely: Fintwol queries its *database*, i.e. lexicon, to retrieve the underlying form or forms corresponding to the input form (here, *ketun*). Each lexicon entry contains three pieces of information: 1) root, 2) ending, and 3) analysis (expressed as a string of tags). The system then produces output containing the root (in this case *kettu*) and the tags (N GEN SG) that correspond to the input. The lexicon itself has been created by applying TLM rules to the system's linguistic resources, as discussed above.

At this point it is necessary to must introduce some key terms in TLM. The *lexical string* (LS) and *surface string* (SS) are the TLM terms equivalent to underlying form (UF) and surface form (SF) respectively (see 6.4.2). A relation between an LS and an SS is termed a *correspondence*, rather than a transformation as in generative morphology. The letters (in the sense of formal grammar) of the TLM alphabet are called TLM *characters*.

6.6.2 TLM alphabet, lexical and surface strings

The characters of the TLM alphabet can be grouped into three subsets (see Table 6.27). The first subset (Ss) are the *surface characters* (such as *a, b, c*, etc.). These are always written in lowercase and represent characters used to write Finnish in actual text, including characters with diacritics. There is also an epsilon/null character, which has no actual surface representation, as usual in formal grammars (see 6.3). Despite their name, surface characters appear in both LS and SS.

Code	Characters	Example
S_S	Surface characters (LS + SS)	e.g. a, b, d, e, f … ∅
S_M	Morphophonemic characters (LS)	e.g. A (represents *a* subject to vowel harmony), T (represents *t* subject to consonant gradation), D (represents infinitive suffix)
S_F	Feature characters (LS)	e.g. = (wildcard), # (word boundary), $ (ending that requires weak vowel), …

Table 6.27. Character subsets used in TLM (adapted from Koskenniemi 1983:23-27)

Morphophonemic characters (S_M) are only used in LS and represent phonemes subject to morphophonemic alternation. The result of the alternation is what appears in the SS. So for instance, character K in LS may correspond to either *k* or null in the SS*,* depending on the morphophonemic context. *Feature characters* (S_F), also known as morphological feature characters, include wildcards, morpheme boundaries and characters that trigger changes in correspondences.

Table 6.28 presents actual examples relevant to the word analysed in Figure 6.15. Example 1 illustrates the LS-SS pair for *ketun* 'fox' (genitive singular), and the correspondence of T to ∅ (T:∅); example 2 shows the LS-SS pair of *kettu* 'fox' (nominative singular) and correspondence of T:t A character-to-character correspondence like T:∅ or T:t is termed a *character pair* (CP).

	Example 1	Example 2
Lexical string (LS)	k e t T u $ n	k e t T u
	| | | | | | | | | | | | | |	| | | | | | | | | |
Surface string (SS)	k e t ∅ u ∅ n	k e t t u

Table 6.28. TLM implementation of alternation from strong to weak grade consonant due to genitive suffix *-n* (adapted from Koskenniemi 1983:17)

Whether T corresponds to t or ∅ depends on the context. Strong consonant gradation (T:t) (in *kettu*) applies when there is no suffix; in the presence of suffix *-n,* weak consonant gradation (T: ∅) applies. This is indicated in the TLM LS for this suffix, $n, including the consonant gradation trigger $, which does not have any surface representation itself.

From these examples, we observe two basic principles of TLM. First, TLM does not view the SS as a result of transformation. Both LS and SS are simultaneously available and for this reason, an LS and its corresponding SS are viewed as a pair, termed a *concrete pair set* (CPS). The rules that create the SS are applied all-at-once rather than sequentially. As a result, intermediate forms are not required. Second, TLM is a symbol-to-symbol formalism. We see in the examples that each symbol in the LS must be paired to exactly one symbol in its SS, even if it is a null.

6.6.3 The TLM lexicon system

In TLM, all roots and endings (the latter being a cover term for suffixes, clitics, and particles; Finnish lacks prefixes) and their tags are included in a lexicon file, in LS form. The overall lexicon is composed of multiple sublexicons, each containing roots and/or various endings. Roots and endings in these sublexicons are combined to form full-word LSs; later, the LS-SS correspondences are created. These processes are automatically performed by TLM transducers (to be discussed in 6.6.4).

Each file containing a sublexicon has a unique name. Figure 6.16 shows part of a sublexicon named *Root*. As its name suggests, this lexicon file contains root entries (LS). A line in the lexicon contains three elements: *entry* (column 1), *continuation class* (column 2) and *information* (column 3).

```
LEXICON Root
     kaTo          /S     "Failure of Crops S";
     katTo         /S     "Roof S";
     liuKu         /S     "Glide S";
     puKu          /S     "Dress S";
     aiTo          /A     "Genuine A";
     tunnetTu      /A     "Well-known A";
     auti.o        /A     "Desert A";
     risti         /S     "Cross S";
```

Figure 6.16. Example root lexicon entries (reproduced from Koskenniemi 1983:155)

There are two types of *information*. For a root entry, this column contains the English gloss of the root followed by its POS tag. For instance, "Roof S" associated with *katTo* means that *katTo* 'roof' is a noun root. "Genuine A" associated with *aiTo* means that *aiTo* 'genuine' is an adjective root. For non-root entries, the information column contains analytic tags (see examples in Figure 6.18).

Continuation classes encode restrictions on root-ending combinations. In English, for example, the root *great* can occur with suffix *-er* (*greater*) but not *-ion* (**greation*). These codes behave like variables (see 6.3.2); each continuation class is a super-category for a group of entries or other continuation classes. The definitions of the continuation classes, and their encoding in lexicon entries, capture the morphotactics of the language (here, Finnish).

For instance, a root in continuation class /S (e.g. *katTo* and *kaTo* in Figure 6.16) can be followed by an ending in lexicon /S (where /S actually indicates any one of S0, S1, S2). But roots with /A (such as *aiTo* and *tunnetTu*) can only be followed by endings in lexicon /A. The definition of /S via subordinate continuation classes S0, S1, and S2 is illustrated in Figure 6.17; Figure 6.18 illustrates some entries for endings in S3.

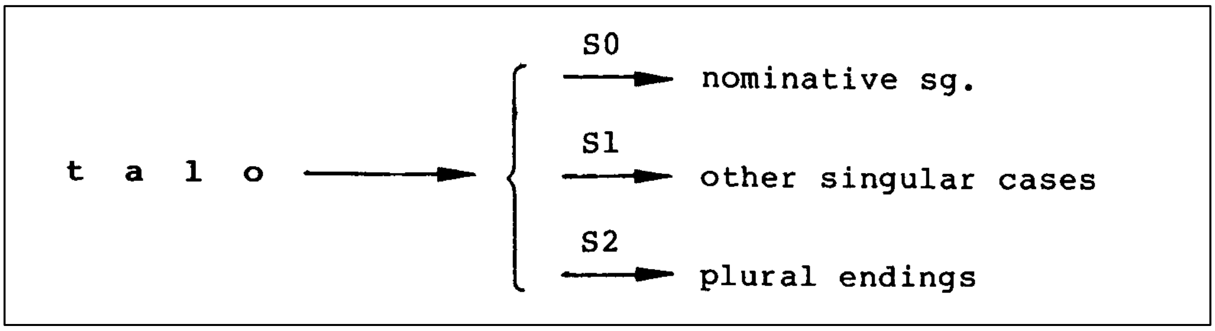


Figure 6.17. Subcategorisation of a continuation class (reproduced from Koskenniemi 1983:46)

```
LEXICON S3
     1+tA      P    "PTV SG";
     2+ten     P    "GEN PL"
```

Figure 6.18. Example S3 lexicon entries (reproduced from Koskenniemi 1983:28)

1+tA, the first entry in lexicon S3, is an ending entry. Its information is not a gloss but an analysis, two tags indicating partitive singular (PTV SG). The definition of the form, 1+tA, consists of four characters. The first is a *selector feature*; it conditions application of this entry on some phonetic feature of the foregoing stem (selector feature 1 is defined in Koskenniemi 1983:85). The second character + is a morpheme boundary. The third is a surface character, and the last a morphophonemic character. That is, this entry characterises a partitive singular suffix consisting of *t* plus some variant of *A*, according to Finnish vowel harmony, which can only follow stems with the given phonetic feature. The entry's continuation class indicates that this ending can be followed by another ending if that ending is in class P. Thus, the rules permit a three-morpheme sequence <root S> <ending S3> <ending P>, as long as all other conditions are fulfilled. Figure 6.19 shows some endings from lexicon P. All have continuation class K, allowing them to be followed by some unit from lexicon K. The continuation class sequence continues as long as the morphotactics of the language requires. Karttunen (1993) built *lexc,* a finite-state lexicon compiler, which is still currently in use.

```
LEXICON P
     /ni       K    "SG1";
     /si       K    "SG2";
     /nsA      K    "SG3/PL3";
     /:n       K    "SG3/PL3";
     /mme      K    "PL1";
     /nne      K    "PL2"
```

Figure 6.19. Examples of P lexicon entries (adapted from Koskenniemi 1983:154)

6.6.4 TLM rules and their FSTs

As established above, a TLM lexicon determines all the possible root-ending combinations on the LS level. The SS level, on the other hand, is defined automatically by TLM rules. TLM rules not only define all proper LS-SS correspondences, but also prohibit certain combinations (Koskenniemi 1983:30). This is reflected by the use of *operators* in TLM, an element not present in generative grammar formalism. A basic context sensitive rewrite rule in TLM has the form *a:b => LC_RC*, meaning "a corresponds to b in the context after LC and before RC", equivalent to $a \rightarrow b / LC\_\ RC$ in generative morphology. Some examples, plus alternative TLM notations used in settings other than actual lexicon files, are given in Table 6.29. But => is only one of four operators that define four types of TLM rule, as shown in Table 6.30.

	Two level rules	Alternative notations
1	a:b => LC_RC	$\frac{a}{b}$ => *LC_RC*
2	a:a => LC_RC	a => LC_RC
3	a:b => LC_RC c:d => LC_RC	[a:b | c:d] => LC_RC

Table 6.29. Alternative notations in TLM

	Rule type	a:b is allowed in context LC_RC	a:b is only allowed in context LC_RC	Must a always correspond to b in context LC_RC?
1	a:b => LC_RC	Yes	Yes	No
2	a:b <= LC_RC	Yes	No	Yes
3	a:b <=> LC_RC	Yes	Yes	Yes
4	a:b \<= LC_RC	No	NA	NA

Table 6.30. Two level rule types (reproduced from Oflazer 1999:194)

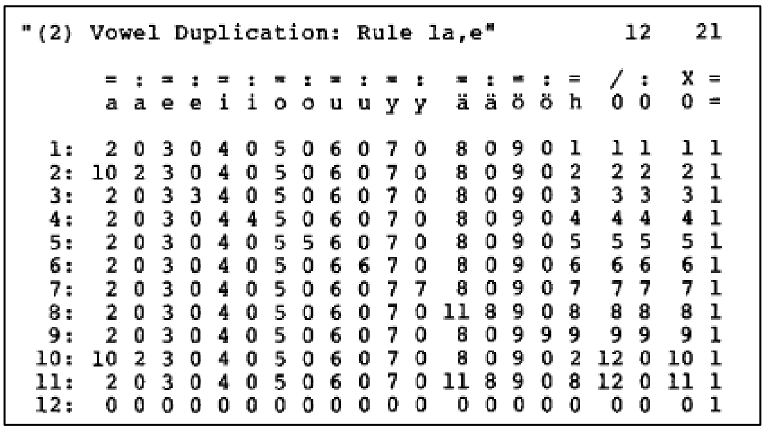
Let us now observe an actual rule, which implements Finnish vowel doubling, from Koskenniemi (1983:40), presented in Table 6.31. The main rule begins with a *correspondence pair* (CP). This CP states a correspondence between two characters, : in the LS and Vs in the SS.

Rule	TLM notation
Main rule: Vowel doubling	$\overset{:}{\text{<Vs>}}$ => $\overset{=}{\text{<Vs>}}$ (h) —
Sub-rule: Doubling of ‘a’	$\overset{:}{a}$ => $\overset{=}{a}$ (h) —

Table 6.31. Finnish vowel doubling as a TLM rule

The colon at LS is a TLM wildcard symbols (not, as in Table 6.30, the correspondence symbol), and means vowel doubling. *Vs* is a variable that represents a set of terminal characters, namely the eight different vowels to which the rule applies, the sub-rule for one of which, *a*, is also given in Table 6.31.

The main rule's contextual condition states that the CP occurs when the preceding CP is *=* paired with *Vs* followed by an optional *h*. The = is another wildcard; optionality is indicated by brackets. This is the complete required left context. The right context, after the underscore, is empty. The operator is =>. This means that CPs other than the CP of this rule *can* occur in the context given after =>. The machine-code version of this rule (i.e. the compiled transducer) is shown in Figure 6.20.



```
"(2) Vowel Duplication: Rule la,e"          12    21

      =  :  =  :  =  :  =  :  =  :  =  :   =  :  =  :  =   /  :   X  =
      a  a  e  e  i  i  o  o  u  u  y  y   ä  ä  ö  ö  h   0  0   0  =

  1:  2  0  3  0  4  0  5  0  6  0  7  0   8  0  9  0  1   1  1   1  1
  2: 10  2  3  0  4  0  5  0  6  0  7  0   8  0  9  0  2   2  2   2  1
  3:  2  0  3  3  4  0  5  0  6  0  7  0   8  0  9  0  3   3  3   3  1
  4:  2  0  3  0  4  4  5  0  6  0  7  0   8  0  9  0  4   4  4   4  1
  5:  2  0  3  0  4  0  5  5  6  0  7  0   8  0  9  0  5   5  5   5  1
  6:  2  0  3  0  4  0  5  0  6  6  7  0   8  0  9  0  6   6  6   6  1
  7:  2  0  3  0  4  0  5  0  6  0  7  7   8  0  9  0  7   7  7   7  1
  8:  2  0  3  0  4  0  5  0  6  0  7  0  11  8  9  0  8   8  8   8  1
  9:  2  0  3  0  4  0  5  0  6  0  7  0   8  0  9  9  9   9  9   9  1
 10: 10  2  3  0  4  0  5  0  6  0  7  0   8  0  9  0  2  12  0  10  1
 11:  2  0  3  0  4  0  5  0  6  0  7  0  11  8  9  0  8  12  0  11  1
 12:  0  0  0  0  0  0  0  0  0  0  0  0   0  0  0  0  0   0  0   0  1
```

Figure 6.20. Machine code of the FST for the Finnish vowel duplication rule (reproduced from Koskenniemi 1983:145)

In total, Koskenniemi (1983:41) lists 22 TLM rules for Finnish. Koskenniemi compiled these rules to machine code by hand, but later TLM systems used compiler programs not available in 1983 to build FSTs from the rules.

The compiled transducer is applied to all lexicons to obtain CPs (pairs of actual LS and SS for particular words; see 6.6.1). The resulting CPSs are stored in a new lexicon, which I will call the *operational* lexicon to distinguish it from the lexicons of LS roots and endings discussed heretofore. When the system receives an SS input, it scans this operational lexicon, and returns the LS and the tag that corresponds to the SS (see 6.6.1). This operational lexicon is the backbone of the system and is what allows it to analyse unrestricted text.

TLM is the underlying formalism of the majority of MA systems up to the present day. But work since Koskenniemi (1983) has added various advances. An early advance and refinement was the porting of Koskenniemi's original INTERLISP program to Common Lisp by Karttunen & Beesley (1992). Already mentioned is the use of an automatic compiler in systems including PC-KIMMO (Antworth 1990), Fintwol (Koskenniemi 1995), and xfst (Kartunnen & Beesley 2003), so that hand compilation is not necessary.

xfst (Karttunen & Beesley 2003:81)*,* the latest TLM system, incorporates not only a TLM compiler but also a regular expression compiler; it follows Koskenniemi's algorithm but is written in the C programming language. The lexicon compiler program mentioned earlier, lexc, is a counterpart to xfst which is capable of fast compilation of lexicons. xfst also has functions to handle more morphological phenomena than earlier implementations, for instance, a function to handle reduplication, which was not present in Koskenniemi's system. While xfst is best known as a base for MA systems, it can also be used to build POS taggers, syntactic chunkers, and shallow parsers.

Overall, TLM has made a significant contribution in two areas, computational linguistics and generative grammar. Karttunen & Beesley (2001) explain that in computational linguistics, TLM was quickly accepted as a useful and practical method to overcome the technological challenges that I reviewed in section 6.5.4. TLM was the first MA able to analyse unrestricted text; earlier MA systems were merely toys.

Kartunnen & Beesly argue that initially, TLM was not seriously considered by mainstream linguists, because many arguments had been advanced in the literature to show that transformations could not be adequately described without sequential rewriting rules. But this has changed. Karttunen & Beesley argue that Optimality Theory (see 6.4.1), a derivative of generative grammar sharply critical of the tradition of ordered rewrite rules, is in effect a two-level theory, but with *ranked* parallel constraints. Thus, the two-level approach to formal morphology has proven both practical for implementation and theoretically respectable.

6.7 A review of present-day tagging practice

6.7.1 Tokenisation

As outlined in 6.2, prior to analysis and disambiguation, a text must undergo *tokenisation* (van Halteren & Voutilanen 1999:110). This is a process in which the sequence of characters in the text is divided into analytical units or *tokens* (Grefenstette 1999:117).

For instance, within the character sequence *the cat*, the combination of *t, h,* and *e* is a word token *the*; the combination of *c, a,* and *t* is another word token *cat*. A program that implements tokenisation is called a *tokeniser*, or sometimes (in NLP) a *word segmenter.* Segmentation algorithms usually target *unsegmented languages*, such as Thai or Chinese*,* where adjacent tokens are rarely or never delineated by space characters (see Sproat et al. 1996; Wong and Chan 1996; Palmer 1997; Meknavin et al. 1997).

By contrast, the term *tokenisation* is more generic and can apply not only to word tokens but also to morpheme tokens (or, theoretically, any other unit of analysis). For instance, MorphInd (see chapter 5) tokenises text into morphemes, whereas IPOS-tagger (Wicaksono & Purwarianti 2011), a POS tagger for Indonesian, tokenises text into words.

The parameters that define what a token is – in a particular language and for a particular annotation scheme – must be encoded into the tokeniser, or made available to a generic tokeniser program as resources. To identify token boundaries, a tokeniser can utilise orthographic cues in the target language or its writing system

Based on orthographic cues, a *separator module* in the tokeniser implements the basic string-splitting task. For word-level tokenisation, spaces and punctuation symbols are likely to be treated as separation points indicating token breaks. A sophisticated tokeniser can also utilise resources (rules, or combinations of rules and lexicons) to perform better tokenisation than orthographic cues alone allow. For instance, to tokenise Multi-Word Expressions (MWEs) as single elements, a lexicon of the MWEs to be treated this way is needed (Grefenstette 1999:119), so that, for instance, *hot dog* can be tokenised as a single unit instead of two, even though the space would normally prompt a token break. Likewise, proper nouns (whose elements may contain spaces) or abbreviations (whose elements are connected by full stops) that would otherwise be treated incorrectly can be handled by listing them in a lexicon.

This approach can generate ambiguity in the sense of multiple possible analyses, for instance between *hot dog* and *hot* then *dog*. To resolve this ambiguity in favour of the MWE token, a tokeniser will typically give lexicon entries priority over automatic rules such as "every space indicates a token break".

The separator module needs to split punctuation marks away from word tokens, effectively making them word tokens themselves. If this is not done, the token data will be distorted. Table 6.31 presents frequency lists for a single sentence based on tokenisation that respectively does not and does split off punctuation. When no split is applied, the system misunderstands *hat.* at the end and *hat* in the middle as two distinct types. In the second column, when a split is applied, *hat* at the end is separated from the dot and therefore considered as a second token of the same type as *hat* in the middle.

Punctuation not split off	Punctuation split off
My (1) Hat (1) And (1) Your (1) Hat. (1)	My (1) Hat (2) And (1) Your (1) . (1)

Table 6.32. Two hypothetical frequency lists from *my hat and your hat.*

Rules for tokenisation typically take the form of regular expressions. Any input sequence that matches a rule's regular expression is assigned the tokenisation specified by that rule. An example of a system like this is the NLTK tokeniser (Bird et al. 2009: 111). Regular expression rules can tokenise whole classes of forms appropriately (e.g. all abbreviations with full stops), so that the actual forms can be left out of the lexicon.

It is also possible for a tokeniser to use rules in the sense of morphological analysis rules (including analytic codes), on condition that the tokeniser has access to this resource. In MA systems, it is typical for tokenisation and annotation to be performed as a single pass through the text, so the analysis rules *are* accessible throughout; see Koskenniemi (1983:89-123), Oflazer (1994:138-146), and Coltekin (2010:821-826), among others.

While lexicon and rule resources are typically used to treat special cases (titles, abbreviations, MWEs) in word-level tokenisers, they are relied on much more heavily by morpheme-level tokenisers in MA systems. Space and punctuation symbols do not reliably delineate morpheme tokens, since polymorphemic words are rarely written with morpheme-delimiting spaces or symbols (the hyphen being a semi-exception to this).

Maximum Matching (MM), sometimes also called *Greedy*, is a basic algorithm commonly used to tokenise words in the previously mentioned unsegmented languages such as Chinese or Thai (see Sproat et al. 1996; Wong and Chan 1996; Palmer 1997; Meknavin et al. 1997). An MM-based tokeniser scans the input string trying to find the longest possible tokens, by comparing different possible candidate substrings to known words in its lexicon – testing long words first, and then sequentially shorter words. Word segmentation in unsegmented languages addresses a similar task to morpheme tokenisation, in that both tasks require the tokeniser to detect invisible token boundaries; thus, the MM algorithm is one possible method for morpheme tokenisation.

MM is often combined with other approaches for optimal results. These include machine-learning techniques, statistical or otherwise. In an example of the latter, Palmer (1997:323) combines MM with Brill's algorithm (Brill 1995), which learns (and orders) the best segmentation rules based on a gold-standard corpus. Goh et al. (2005) combine MM with a probabilistic model, the *Support Vector Machine* (SVM), to resolve ambiguities; Wong and Chan (1996) combine MM with an approach that they refer to as *binding force.* Such algorithmic experimentation is prominent in the NLP literature on segmentation/tokenisation.

6.7.2 Annotation

Annotation is the assignment of potential analyses (tags) to tokens (van Halteren & Voutilanen 1999:110). The tags may come from various annotation resources such as a lexicon or rules; there are multiple approaches to using these resources for annotation, which the following two sections will explore.

6.7.2.1 Lexicons for annotation

The account in this section draws on the explanations of Schiller & Kartunnen (1999: 135-148), Monachini & Calzolari (1999:149-174), and Leech & Wilson (1999: 55-80). Regardless of the type of system, a lexicon is required for annotation. An annotation lexicon contains forms linked to analytic codes or tags, according to whatever predefined annotation scheme is in use.

A lexicon can be created in two ways (or, possibly, by combining both methods). First, it is possible to handcraft a lexicon by manually importing information from observations of texts, dictionaries, or the author's introspection. The lexicon author must encode the information in the entries according to the annotation scheme, and in the target software's required lexicon format. The other way to create a lexicon is to compile a frequency list of form/analysis pairings from a pre-tagged corpus. This results in a corpus-derived lexicon. The frequency of each analysis may be included in a lexicon; that frequency data is not utilised by rule-based annotation but *is* of use to statistical systems when estimating different analyses' probability. Forms not in the lexicon(s) must be handled by another annotation resource – most likely, annotation rules.

6.7.2.2 Rules for annotation

Rule-based annotation is characterised by the use of rules in addition to a lexicon. This approach is more prominently discussed in the literature on MA systems than in that on POS taggers, for reasons that will become clear. I will begin by illustrating the use of rules in a ‘guesser’ module for a POS tagger, but the same principle applies to an annotation module in an MA as well.

Lexical lookup can give ambiguous results (Voutilanen 1999:14), but it can also leave a token unanalysed, if its form is unknown (not in the lexicon). Words being unknown often arise from a paucity of data when the lexicon was generated (Brill 1999:207). Forms not in the lexicon can be handled in multiple ways. Unrecognised tokens might be left untagged. Alternatively, they might receive a specific label indicating ‘unknown’; for instance In MorphInd, the label *X—* is used (see 3.5.2). But a system may also use a guesser module, which will try its best to analyse the token.

Such a module utilises some kind of rule to generate a best-guess valid tag for the unknown form. According to Bird (2009:181), a variety of cues (affixes, neighbouring tokens, orthographic cues, etc.) can be used in a POS guesser. Two examples of guessing rules for English POS might be: (1) label the unknown token as *noun* if it is preceded by *a* or *the*; (2) label the token as *past tense verb* if it ends in *-ed*.

Rules are used to govern what analyses from the lexicon can be accepted for a given morpheme in the context of the word-form. Nevertheless, the mechanism of rule application may differ from one system to another. For instance, to appropriately analyse the two morpheme tokens of English *player* as *play/V* and *er/NOMZR*, multiple possible routes exist. If the lexicon contains an entry <play,V>, then the first morpheme can receive that analysis. However, many MA systems implement an additional mechanism which tests whether that analysis is valid given the token’s context, that is, the complete word it appears in. Only if the analyses the lexicon provides for *play* and *er* are accepted by the rules as a morphologically valid combination will that full-word analysis be accepted. Alternatively, the lexicon might contain a complete analysis that would pre-empt the rules, e.g. “play er” => <play/V><er/NOMZR> as a single lexicon entry.

Practically, the boundary between the roles of lexicon and rules in this process can be blurred. Koskenniemi’s (1983) original TLM system for Finnish MA (see 6.6) has rules which mostly account for morphophonemic and morphographemic (orthographic) variation; morphotactics are handled not in the rules but in the lexicon. The main lexicon in Koskenniemi’s system is the root lexicon. This is connected to *ending* lexicons by continuation class codes. As noted in 6.6.3, the continuation class codes implement morphotactic rules – via data stored in a lexicon file not a rule file.

Thus operationally, the morphotactic component in Koskenniemi's system is represented within the lexicon(s). However, on the conceptual level, the morphotactic component is part of the implementation of word formation rules. Regardless of the technicalities, which vary from one tool to another, rule-based annotation is the same basic procedure: using lexicon entries and rules to assign potential tags to the tokens, resulting in an analysis which may or may not be ambiguous.

Oflazer (1999:193-194) recommends that MA developers prepare the following materials: 1) a list of roots with corresponding analytic codes; 2) a list of other morphemes with corresponding analytic codes; 3) a morphotactic model; 4) a comprehensive list of morphographemic phenomena; and 5) a corpus on which to test the MA. These materials should then be adapted to a format compliant with the software being used to build the MA system.

Without rules, an MA assigns all tags acquired from lexicon lookup matches, regardless of morphotactic correctness. The hypothetical lexicon in Table 6.33 has entries for two roots and two affixes. Without morphotactic rules, lexical lookup for either of *smaller* or *player* will always produce ambiguous tagging for *-er*, namely *er/NOMZR* and *er/SUP*. Morphotactic rules are needed to enforce the restriction that only NOMZR is correct after V, and only SUP after A, eliminating the ambiguity.

small,A
play,V
er,NOMZR
er,SUP

Table 6.33. A hypothetical mini-lexicon for English

It is common practice to apply the rules of a guesser module *only* if no match is found by lexical or rule lookup. As noted above, this can be implemented via a prioritisation system for different resources and modules.

6.7.3 Disambiguation

As outlined in 6.2, disambiguation is the removal of (likely-to-be) incorrect tags from ambiguously tagged text (Voutilanen 1999:6). Disambiguation may or may not reference context beyond a single token (*contextual* or *non-contextual* disambiguation) and can be *rule-based* (or linguistic) or *statistical* (or data-driven). Taggers with rule-based disambiguation may be referred to as *finite state* or *syntax-based* taggers. These different terms reflect the different advanced techniques in use; all are conceptually similar. Likewise, statistical approaches utilise many different techniques or models.

6.7.3.1 Rule-based disambiguation

Rule-based disambiguation may be contextual or non-contextual.

Figure 6.21 exemplifies one of MorphInd's non-contextual disambiguation rules. This rule applies at word level and references no neighbouring words.

```
# If ambiguous between '+i_' and a full word, choose a full word
#^astronom<n>+i_VSA/astronomi<n>_NSD$ = ^astronomi<n>_NSD$
```

Figure 6.21. One of the non-contextual disambiguation rules in MorphInd (reproduced from MorphInd.pl: see section 5.5.5)

This rule expresses one concrete case of the general principle that when a word is ambiguously analysed at the annotation stage as monomorphemic and polymorphemic, the system should preserve the monomorphemic analysis, and remove the polymorphemic analysis. The word in question, *astronomi*, is annotated as verb plus verbaliser suffix (*astronom<n>+i_VSA*) and as monomorphemic noun (*astronomi<n>_NSD*). The polymorphemic analysis is removed, so only the monomorphemic annotation remains in the final output. This non-contextual disambiguation rule applies regardless of what precedes or follows *astronomi* in the input string.

Let us now consider application of contextual disambiguation rules, using the following hypothetical case. The word *walk* may be a plural verb, or a singular noun. A possible contextual disambiguation rule would be as follows: if the present word is tagged ambiguously as N-SG (singular noun) and as V-PL (plural verb), then if the previous word is *this* or *that* or *the* or *a*, then remove the tag V-PL. Thus, in the context *the walk,* the target word *walk* is correctly disambiguated as a noun.

This procedure can be implemented in many different formalisms. Moreover, just as tokenisation or annotation resources can be assigned different priority levels, so can disambiguation rules, if the software supports this functionality (Silberztein 2003: 150-154). We will return to this issue later on in 7.4.4.

6.7.3.2 Statistical disambiguation

Statistical disambiguation selects the most probable analysis for each ambiguous token (and deletes less probable analyses) from among those assigned in the annotation stage. Different systems apply a variety of statistical or machine learning models to this task. Any such model is made up of a (typically large) collection of statistical parameters. These statistical parameters are acquired (or *learned*) from a pre-tagged corpus, often called a *training* corpus or *gold-standard* corpus.

In its *learning* or *training* stage, the system processes the pre-tagged corpus to extract statistical information, which is then saved. The type of statistical information varies depending on the choice of statistical model or machine-learning technique. Common models include Hidden Markov Models (see El-Beze & Merialdo 1999), data-driven local rules, inductive learning, case-based learning, decision tree induction, and neural networks. Daelemans (1999) discusses many of these at length.

The result of training is a set of numeric parameters, forming a model as noted above. The disambiguation module or system can then use this statistical information to compute probabilities for alternative analyses of ambiguously annotated (sequences of) tokens. The system then resolves the ambiguity by selecting the analysis, or sequence of analyses, with the highest probability – removing the rest.

6.8 Measures for evaluation of tagging systems

6.8.1 Evaluating tagging systems

A number of concepts related to tagging system evaluation have been discussed earlier. The concept of a *training corpus* was introduced in 6.7.3.2; Bird (2009: 203) shows it is common to extract a small proportion (e.g. 10%) of the training corpus for use as the *testbed* (data for use in evaluation exercises). A non-tagged version of the testbed corpus is created, which the system than tags. To measure how well the system performs, the re-annotated testbed can directly be compared to the version of the testbed with gold-standard tags, and the difference in correctness quantified.

In section 5.5, I presented two evaluation measures: *coverage* and *accuracy*. Coverage is the proportion of tokens the system manages to analyse, regardless of the correctness of the analyses. Accuracy is the proportion of tokens in unambiguous output that have the correct analysis. However, some systems do not perform full disambiguation (Voutilanen 1999:18), but rather, remove as many incorrect analyses as possible and leave the rest. An evaluation of such a system needs to consider the remaining ambiguity. Hence, rather than accuracy, two evaluation measures that are sensitive to ambiguities, *precision* and *recall*, are used.

6.8.2 Precision, recall and F-measure

Precision and recall are evaluation measures commonly used in IR, but also to evaluate tagging systems in which ambiguities are reduced but not eliminated (Manning and Schutze 1999:534-536). *Precision* measures how well the system removes incorrect annotations. It is calculated by dividing the sum of *true positives* (correct annotations) by the sum of all annotations (including ambiguities). This can be illustrated by the evaluation of a hypothetical testbed corpus (Table 6.35) relative to a gold-standard (Table 6.34). The last token in the testbed has two analyses, one of which is incorrect (false positive). Thus, the precision of the system is 5/6, that is 0.83 or 83%.

	Tag
can	V
I	PRON
get	V
a	DET
lighter	N

Table 6.34. A hypothetical gold-standard annotation

	Tag	
can	V (TP)	
I	PRON (TP)	
get	V (TP)	
a	DET (TP)	
lighter	N (TP)	A (FP)

Table 6.35. A hypothetical testbed sentence, annotated by the system (TP = true positive/correct, FP= false positive/incorrect)

Recall measures to what extent all the correct tags are retained in the output. It is calculated by dividing the sum of true positives by the sum of all true positives *and* false negatives (i.e. total number of correct tags absent from the output). The hypothetical system discussed above has perfect recall on the sample data (5/(5+0)=1 or 100%) because there are no false negatives in Table 6.34; the additional incorrect tag on the last token does not affect the recall.

In some cases, a combined statistic called F-measure is reported (see Hripsack & Rothchild 2005). This is the harmonic mean of precision and recall. Its formula is (2 x Precision x Recall) / (Recall + Precision). Therefore, the F-measure for the above hypothetical is ((2 x 0.83 x 1)/(0.83+1)) = 1.66/1.83 = 0.91 or 91%.

6.8.3 Ambiguity rate and error rate

Ambiguity and error rates are the evaluation measures used to evaluate the POS tagging of the BNC test sample (Leech & Smith 2000 section C). In essence, they are very similar to the aforementioned measures (precision, recall, accuracy). *Ambiguity rate* means the proportion of tokens whose annotations are ambiguous. For the data in Table 6.35, the ambiguity rate is 0.2 or 20%: the total number of annotations, minus the number of tokens, divided by the number of tokens ((6-5) / 5 = 0.2). The *error rate* is the proportion of tokens for which no correct tag is retained: the number of tokens without a correct annotation, divided by the number of tokens. For the example in Table 6.35, the error rate is 0 or 0% (0/5). These measures will be utilised in Chapter 7 to evaluate my MA once implemented.

6.9 Choice of approach for a new Indonesian MA

The purpose of the foregoing review is to provide a rationale for the choice of approach for the new MA system whose implementation will be discussed in chapter 7. I argue that the rule-based approach is preferable in this project, that is, using rules to perform tokenisation, annotation, and disambiguation (both contextual and non-contextual). No machine-learning or statistical techniques will be used. The reasons for this decision are as follows. First, the rule-based approach is the basis for a number of instances of best practice. This includes Koskenniemi's (1983) seminal work, the Constraint Grammar tagger (Karlsson 1990), BAMA (Buckwalter 1999), and TR-Morph (Coltekin 2010).

Even statistical systems commonly incorporate rule-based subsystems, as exemplified by MorphInd (Larasati et al. 2011). MorphInd uses morphological rules from the earlier MA of Pisceldo et al. (2008), and then adds other rules including (non-contextual) disambiguation rules of thumb (see 5.2). MorphInd's statistical disambiguation module therefore does not need to target *all* ambiguity, but only ambiguity not resolved by the preceding rule-based disambiguation module.

Second, rules are linguistically explicit. Although rules must be written in a format compliant with the software, they are still relatively 'readable' for humans – with several advantages. This stands in contrast to statistical MA systems, in which linguistic knowledge is represented by numeric parameters, which are not 'readable' in that way. Readable rules allow a developer to modify or improve the existing rules with ease even if they are not the original creator. This facilitates continuous development of the system. Moreover, human-readable rules may be informative for analysts working with system output, for instance, enabling them to know what analyses are driven by definitive analysis (e.g. morphotactics) as opposed to being best guesses. In addition, readable rules assist evaluation. In chapter 5, I showed that MorphInd's incorrect analysis of polymorphemic words as monomorphemic came from its non-contextual disambiguation rules. None of this is possible within an approach which centres statistical parameters.

Third, rules are linguistically meaningful. They are reflections of the grammar of the language (morphotactic patterns, morphophonemic alternations, syntactic constraints, etc.) and are typically (not always) written by grammar specialists. Conversely, statistical learning does not produce linguistically meaningful rules, with the possible exception of the Brill tagger.

Fourth, a rule-based system does not rely on a pre-tagged corpus. Such a corpus might still be required for evaluation purposes, but not for system implementation. By contrast statistical disambiguation relies on a pre-tagged training corpus (usually large), from which the model's parameters are acquired. There do exist systems which use *unsupervised learning*, i.e. training on an untagged corpus. Brill (1995) and El-Beze & Merialdo (1999) both use unsupervised learning to train a tagger (rule-based and statistical respectively). However as El-Beze & Merialdo (1999:272) report, unsupervised learning leads to worse results.

Fifth, rule-based systems are not associated with poorer performance than statistical taggers (Voutilanen 1999: 18-20). Thus, rule-based systems get the aforementioned advantages of rules without losing correctness.

6.10 Choice of software for the new MA system

In section 6.7, I discussed various aspects of the software used for existing rule-based systems. My choice of program on which to construct a new MA system is based on that review. It is possible to build a rule-based MA from scratch using a general programming language such as Perl, Python, Java, or C. For instance, Koskenniemi (1983) wrote his MA system in LISP; the well-known Arabic MA, BAMA (Buckwalter 1999) is written in Perl. Both Koskenniemi and Buckwalter wrote, from scratch, both the core morphological analysis program (the FSM module) and other supplementary functions for their MA systems.

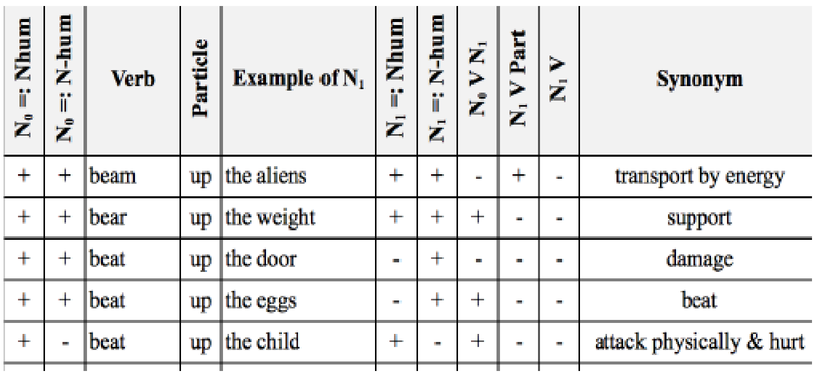
There exists general FSM software specifically equipped to carry out rule-based morphological analysis. Using such a program eliminates the need to write an FSM system from scratch. The developer only needs to incorporate morphological resources for the target language into the system. Such programs include xfst (Kartunnen and Beesley 2003), hfst (Linden 2011), and foma (Hulden 2009). These FSM programs follow Koskenniemi's system architecture. Of these programs, xfst and foma are the platforms that underlie Piscaldo et al.'s (1999) Indonesian MA and MorphInd, respectively.

Although the primary use of FSMs in MAs is to tokenise and annotate, some FSM software (xfst, foma) may also be used for disambiguation, often via experimental additional modules (Oflazer and Gokhan 1997; Pirinnen and Linden 2009). But in practice, in the field to date, these programs are still largely used for analysis (tokenisation/annotation), rather than disambiguation. For instance, in MorphInd, the disambiguation rules are implemented in Perl, even though foma is used for tokenisation and annotation. Of course, it is common for an MA system to link multiple programs together, but a rule-based MA must have an FSM engine as the core of the system, with other programs as auxiliaries.

While all the aforementioned FSM programs for morphological analysis follow Koskenniemi's TLM, others do not. NooJ (Silberztein 2003) and its predecessor Intex (Silberztein 1993;1997) possess slightly different characteristics. They use an FSM not only for morphological tokenisation and annotation, but also for rule-based disambiguation. These tools were developed to implement Maurice Gross's Lexicon-Grammar theory (Gross 1994; 1997)[54]. This theory centres the description of idiosyncratic properties of lexical elements. A partial example of a lexicon-grammar table is given in Figure 6.22; it describes the syntactic and semantic properties of phrasal verb *beam up* (N0 and N1 symbolise this verb's arguments). Before the creation of these tools, lexicon-grammar tables were compiled by researchers following Gross's lead, but could not be applied to automatic text analysis.

A computational implementation of Gross's Lexicon-Grammar not only requires full table data, it must also be able to take into account the language's morphological rules. This is done by incorporating an FSM system (the same approach used in TLM). The first program to accomplish this was Intex (Silberztein 1993;1997).

[54] Another program designed to implement Gross's Lexicon-Grammar theory is Unitex (Paumier 2014). However, some controversy attaches to intellectual property issues around this software.(an outline being given at http://www.nooj-association.org/intex-and-unitex.html; accessed 28/10/2021). Given this issue, the remainder of the review here omits mention of Unitex.



N_0 =: Nhum	N_0 =: N-hum	Verb	Particle	Example of N_1	N_1 =: Nhum	N_1 =: N-hum	N_0 V N_1	N_1 V Part	N_1 V	Synonym
+	+	beam	up	the aliens	+	+	-	+	-	transport by energy
+	+	bear	up	the weight	+	+	+	-	-	support
+	+	beat	up	the door	-	+	-	-	-	damage
+	+	beat	up	the eggs	-	+	+	-	-	beat
+	-	beat	up	the child	+	-	+	-	-	attack physically & hurt

Figure 6.22. Lexicon-grammar table for some phrasal verbs in English (reproduced from Silberztein 2016:92)

Today's equivalent to Intex is NooJ (Silberztein 2003). An MA built in NooJ or Intex could analyse *beaming up* as <beaming up,V+Progressive>, a word-level analysis, or just as easily as <beam,V> <ing,Progressive> <up,Particle>, a morpheme-level analysis. This fits exactly both with a crucial aim of this project, which is to perform annotation at morpheme level, and with the design of the Morphological Annotation Scheme (MAS) for use in the new system (see, particularly, 4.1.2).

NooJ, like Intex before it, is widely used, not only for tokenisation and annotation, but also for disambiguation. It can perform disambiguation using a priority system and disambiguation rules. Conversely, other FSM-based systems (e.g. xfst) do not standardly permit use of both forms of FSM during tagging. Some other systems have non-standard extensions for FSM-based disambiguation; from an abundance of caution, I have opted to avoid such non-integrated add-ons.

That all the three typical tagging procedures can be carried out by a single program, plus the disambiguation feature being included in the core program, makes NooJ and Intex ideal candidates for the software implementation of my MA. By comparison, in MorphInd, the three procedures are implemented by separate modules (see 5.2). Moreover, the MAS I devised requires both orthographic and citation forms to be presented and associated with morphological tags. NooJ and Intex support this feature. Conversely, other FSM-based systems do not standardly present both forms in the output.

While not directly related to tokenisation, annotation, or disambiguation, certain other features in NooJ and Intex further advantage them over xfst and foma. Both are completely free to use and download, and available for multiple operating systems. Conversely, according to Larasati et al. (2011:120-121), xfst's compile-replace function, used to analyse reduplication, is patent-encumbered, and only available in a commercial version of xfst. That said, the equivalent function in foma is not patent-encumbered, so foma's only disadvantage relative to NooJ or Intex is that the functionality is not integrated.

In addition to integrating the three sub-tasks in one system, NooJ, previously Intex, is also equipped with debugging and corpus query functions (the latter being similar to concordancers such as LancsBox, AntConc or WordSmith Tools). They also allow users to build multi-level annotation systems, simultaneously morphological, morphosyntactic and syntactic, with the potential for levels to interact – using annotation at one level to analyse units at another level without additional lexicon/rule lookup. In sum, NooJ and Intex are free to use, possess integrated tokenisation-annotation-disambiguation, and offer built-in debugging and corpus query functions.

Of the two, I prefer NooJ on grounds specific to this particular project. NooJ has a specific function (called 'equality constraint') for analysing full reduplication; this is required by my MAS (see 4.2.4). Relative to any other alternatives, NooJ is richer in functionality and more flexible.

6.11 Concluding remarks

In this chapter, I have explored the formal theoretical background that underlies MA systems. I have also provided a review of the development of MA programs, from the earliest work to the design of present-day MA systems. The findings of that review then fed through to the rationale for my choice of approach to implement the system, and my choice of software platform. The new MA system will use the rule-based approach, and it will be implemented in NooJ (Silberztein 2003). At this stage, everything needed to commence construction of the new MA system for Indonesian has been determined: a new MAS plus justified choices of approach and software. These design decisions ensure that the system shall permit an advance over the current state-of-the-art MA system (see Chapter 5).

CHAPTER 7

IMPLEMENTATION AND EVALUATION

7.1 Introducing SANTI-morf

The name for this thesis's new automatic morphological analysis system for Indonesian is *SANTI-morf*. SANTI is the acronym of *Sistem ANalisis Teks Indonesia* or in English 'A System for the Analysis of Indonesian Texts'. The word *Santi* is very familiar for Indonesians as it is a common female name, originally from Sanskrit and meaning 'peace'. The *morf* part is clipped from *morfologi* 'morphology'.

SANTI-morf implements the Morphological Annotation Scheme (MAS) I devised in Chapter 4. As noted in sections 6.9 and 6.10, the system's foundation is a rule-based architecture, implemented in a program called NooJ (Silberztein 2003). I introduce NooJ in section **Error! Reference source not found.**, using English examples for readers' convenience. SANTI-morf's architecture is described in section 7.3. The actual implementation of SANTI-morf for Indonesian with actual Indonesian resources is reported in section 7.4. An evaluation of its performance on the same testbed used in my evaluation of MorphInd (see 5.5) is presented in section 7.5.

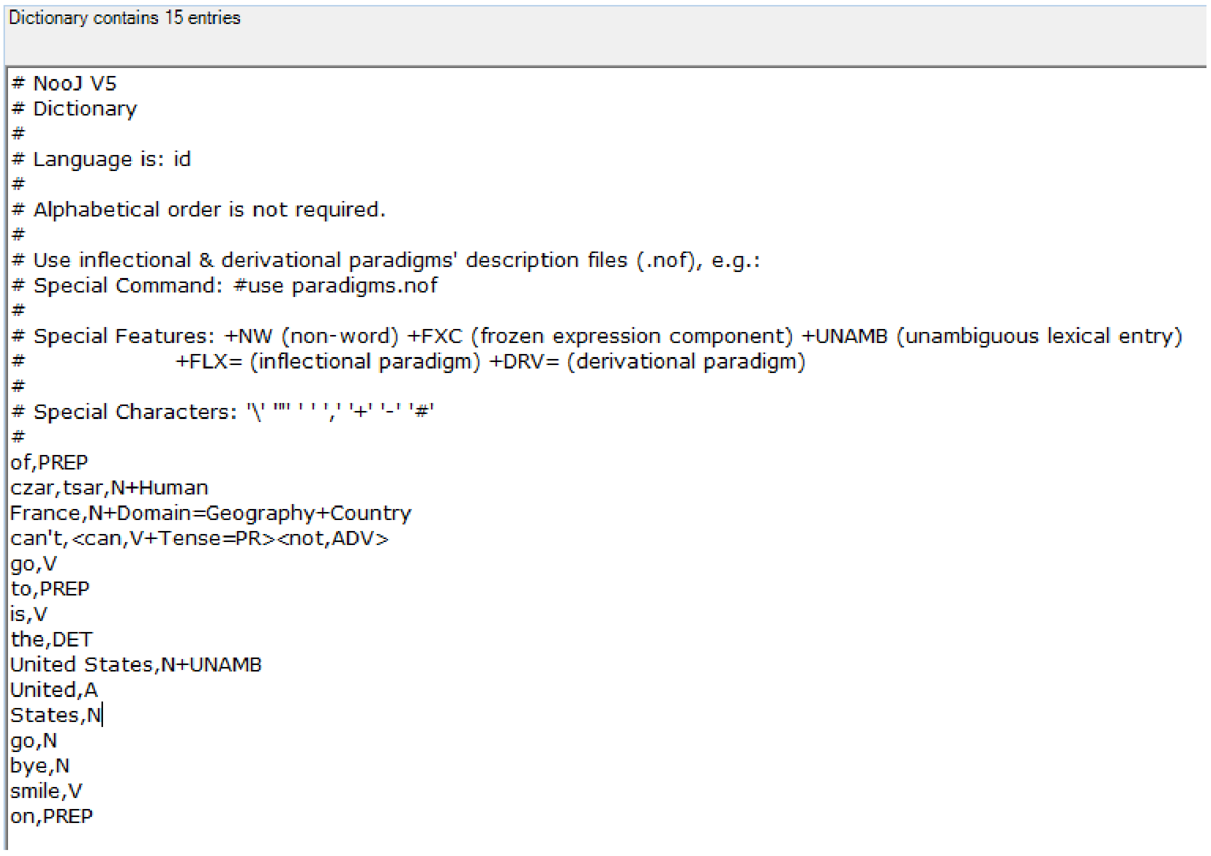
7.2 NooJ

NooJ is a program for searching and/or annotating text using finite state machines via a graphical user interface. NooJ allows its users to construct natural language resources in the form of lexicons and rules (termed *dictionaries* and *grammars* in NooJ). NooJ applies these resources (simultaneously or consecutively/in pipeline) to tag texts at various levels (morphological, morphosyntactic, syntactic, etc.). The overview of relevant aspects of NooJ in this section draws on the NooJ manual (Silberztein 2003) and accompanying book (Silberztein 2016).

7.2.1 NooJ lexicons

7.2.1.1 Basic format and operation

A NooJ lexicon file consists of lines that encode *lexical entries* (lines beginning with hash (#) are comments). A lexical entry consists of a *form* and *tag*. The forms may be word forms, morphemes, or multi-word sequences. A tag is composed of one or more *property codes* representing any combination of POS, formal or functional morphological features, and semantic properties.



Dictionary contains 15 entries

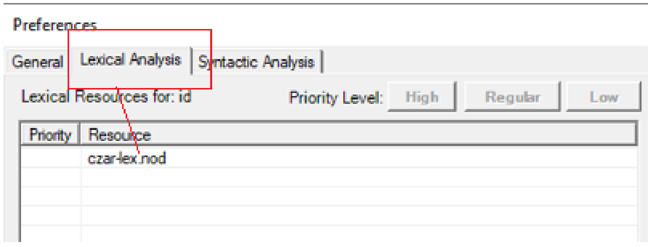
```
# NooJ V5
# Dictionary
#
# Language is: id
#
# Alphabetical order is not required.
#
# Use inflectional & derivational paradigms' description files (.nof), e.g.:
# Special Command: #use paradigms.nof
#
# Special Features: +NW (non-word) +FXC (frozen expression component) +UNAMB (unambiguous lexical entry)
#                 +FLX= (inflectional paradigm) +DRV= (derivational paradigm)
#
# Special Characters: '\' '"' ' ' ',' '+' '-' '#'
#
of,PREP
czar,tsar,N+Human
France,N+Domain=Geography+Country
can't,<can,V+Tense=PR><not,ADV>
go,V
to,PREP
is,V
the,DET
United States,N+UNAMB
United,A
States,N
go,N
bye,N
smile,V
on,PREP
```

Figure 7.1. An example NooJ lexicon

The simplest lexical entries, such as `of,PREP` in Figure 7.1, consist only of a form (`of`), a delimiter (comma), and a tag composed of a single property code (`PREP`). Because this lexical entry contains only one form, NooJ treats this as both orthographic and citation form (see definitions in section 3.7.2). Forms may include non-letter symbols such `£`, `$`, or punctuation marks. But in some cases, the same non-letters are meaningful to NooJ. For example, the comma is used as a delimiter. If we want to include a comma (or another non-letter symbol meaningful to NooJ) as part of a form or tag, we need to escape the symbol with slash, as in `\,` instead of just `,`. Lexical entries like `czar,tsar,N+Human` contain two comma-delimited forms. The first is the orthographic form, the second the citation form. The NooJ terms for the two are *word form* and *lemma*, representing the perspective of POS tagging rather than morphological tagging. This entry's tag exemplifies multiple properties, *N* (noun) and *Human*, connected with a plus symbol. The difference between the two forms in this example is one of spelling regularisation, but the same layout is used for allomorphy.

Property codes can alternatively be written in *attribute value* style, so that in `France,N+Domain=Geography+Country`, `Domain` is an attribute and `Geography` is the value it has for the word *France*.

NooJ uses the extension .dic ('DICtionary') for lexicon text files, and compiles each such file to a corresponding .nod ('NooJ Dictionary') file with same base name. For use in annotation, .nod lexicon files must be loaded into NooJ. The resource panel interface in NooJ lists all resources actively in use. It also controls the priorities of these resources (to be discussed later). There are two resource panels: *Lexical Analysis* and *Syntactic Analysis* (see Figure 7.2).

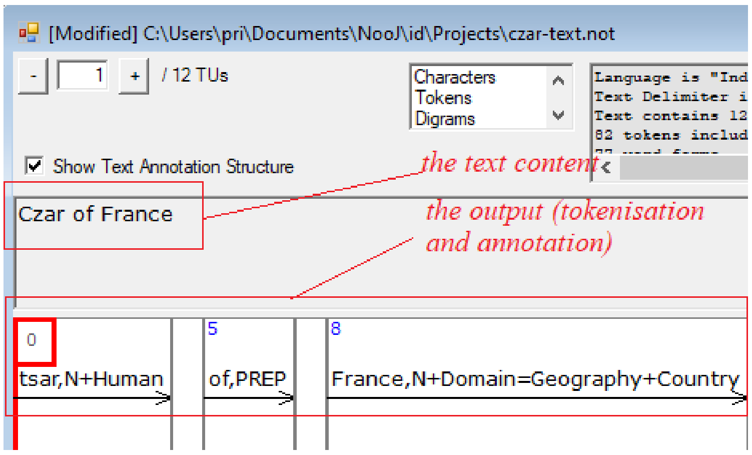


Lexical Resources for: id

Priority	Resource
	czar-lex.nod

Figure 7.2. Resource panel in NooJ, showing one active lexicon

NooJ files containing textual data for analysis are given the extension .not ('NooJ Text'). Figure 7.3 below illustrates the application of a lexicon to a short text.

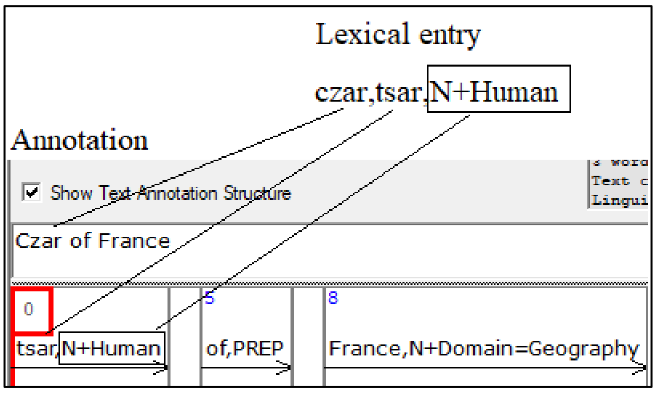

[Modified] C:\Users\pri\Documents\NooJ\id\Projects\czar-text.not

Czar of France

tsar,N+Human | of,PREP | France,N+Domain=Geography+Country

Figure 7.3. Applying a lexicon to a text in NooJ

NooJ processes the text using the FST compiled from the lexicon. When an input sequence is accepted, the linked annotation is the output. This is the standard operation of an FST as outlined in 6.6. An illustration of how the NooJ results for the token *czar* correspond to the lexical entry components is given in Figure 7.4.



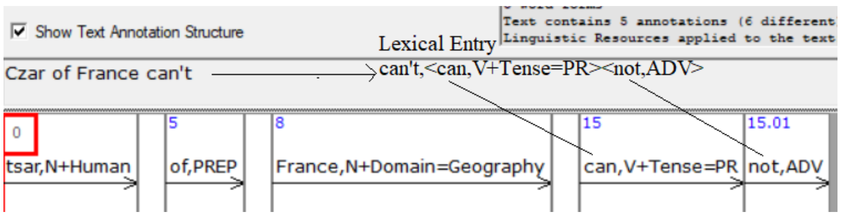
Czar of France

tsar,N+Human | of,PREP | France,N+Domain=Geography

Figure 7.4. A lexical entry in relation to input and annotation

Orthographic forms in lowercase in the lexicon are matched case insensitively; if any uppercase character is present, the form is matched case sensitively. Thus, *France* is matched to the entry `France,N+Domain=Geography` because the character case matches; *france* or FRANCE would not be accepted. Conversely, *czar* in the lexicon matches *Czar* in the text.

The lexical entry `can't,<can,can,V+Tense=PR><not,ADV>` (encoded in the dictionary file shown in Figure 7.1) exemplifies an analysis which tokenises one space-delimited form, *can't*, into two tokens; Figure 7.5 shows the resulting output in NooJ. When this lexical entry is applied, *can't* in the text is analysed as two tokens instead of one. The output, and its links to the lexical entry, are shown in Figure 7.5.



```
Text contains 5 annotations (6 different
Linguistic Resources applied to the text
```

Czar of France can't → can't,<can,V+Tense=PR><not,ADV>

tsar,N+Human | of,PREP | France,N+Domain=Geography | can,V+Tense=PR | not,ADV

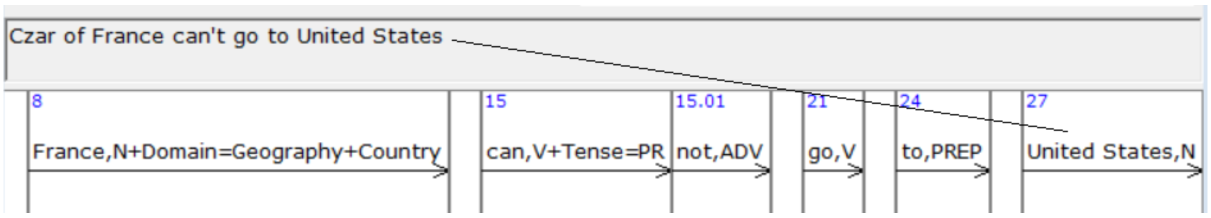
Figure 7.5. The analysis of *can't* into two tokens

7.2.1.2 The application of 'unambiguous' lexical entries

NooJ has certain built-in property codes termed *special features*, which trigger implementation functions. One such special feature is `+UNAMB` ('unambiguous'), which marks an entry as *prioritised*: to be checked prior to other entries (and prior to division of the text by whitespace). Thus, the lexicon in Table 7.1 would always treat `United States` according to the first entry with `+UNAMB`, not according to the general entries for `United` and `States`. The code `+UNAMB` is omitted in the annotation output (see Figure 7.6).

1	`United States,N+UNAMB`
2	`United,A`
3	`States,N`

Table 7.1. A lexicon with a `+UNAMB` entry



Czar of France can't go to United States

France,N+Domain=Geography+Country | can,V+Tense=PR | not,ADV | go,V | to,PREP | United States,N

Figure 7.6. Annotation with the `+UNAMB` lexical entry `United States,N+UNAMB`

Without +UNAMB, NooJ produces all possible analyses, as shown in Figure 7.7. Here, NooJ produces an ambiguous annotation. Ambiguous analysis is also the result when two matching lexical entries are found (and neither is +UNAMB), such as for a word which may be a verb or noun (entries go,N and go,V in Figure 7.7, for example).

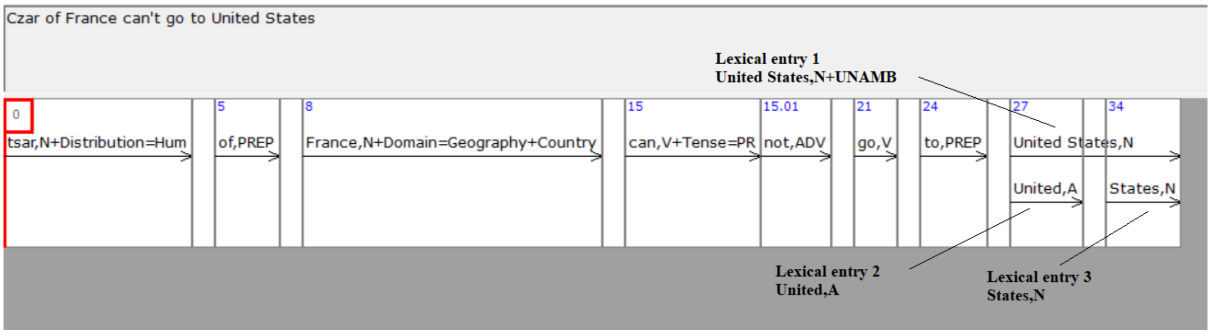


Figure 7.7. Annotation without any priority

7.2.2 NooJ rules

In NooJ, rules are used to generate the FST that processes and tags the text. NooJ supports three types of rules: *inflectional-derivational grammars, morphological grammars,* and *syntactic grammars*. Of these I use the morphological and syntactic grammars.

7.2.2.1 Morphological grammars

Morphological grammars are saved in files with the .nom extension (*NooJ Morphological* grammar). The rules in these files are used for the *tokenisation* and *annotation* of words, not for disambiguation. The input words must be composed of a string of letters only. For example, NooJ will apply morphological rules to the form *going,* but not *go-ing*, because the latter contains a hyphen, which is a non-letter symbol.

All NooJ rules, not only morphological grammars, are written in a regular grammar formalism, although the notation is not identical to that discussed in section 6.3. Every rule must have a *name,* followed by an equals (=) sign; this is equivalent to the FST's start state, and thus to a regular grammar's start symbol.

The first rule in each resource file must be named `Main`. Other rules can be named anything, but must be organised under the hierarchy of the Main rule. This is because each rule's name is a non-terminal symbol (or variable) in the regular grammar (see 6.3.3). Rules end with a semi-colon. Non-terminals are marked with a colon (:) preceding the rule name (e.g. `:ING` for the rule of *-ing* suffixation). All other symbols are terminals. NooJ's notation for epsilon (empty) is `<E>`.

The terminals of two grammars are linked in a regular relation as input-output components (see 6.3.3.2.3). In NooJ notation, the format is *input / output* . In morphological rules, the output is written in the format for lexical entries described in 7.2.1.1, and enclosed in angle brackets. So, for example, the rules `go/<go,V>` and `ing/<ing,SFX>`, would pair each of two terminal nodes in the input language, *go* and *ing* from an input *going*, to the equivalent terminal node (tag) in the output language.

NooJ compiles rules at runtime, unlike lexicon files, which are compiled in advance. Unlike basic regular grammars (see 6.3.3.2), NooJ rules do not limit how many terminals and non-terminals may be present in a single rule.

A rule may be defined abstractly to apply to many lexical items. For this purpose, a NooJ *variable* (not in the sense of the regular grammar term) must be used. A NooJ variable (henceforth just variable) is a string that can match many inputs depending on its *constraint*, which can target either lexical properties or forms. In addition to a variable and constraint, an abstract rule must include an output component; see below.

```
SAMPLERULE =       $(X <L>* $)        <E>/<$X=:V>        <E>/<$1L,$1C$1S$1F>
# rule name        variable           constraint         output
```

`$(X <L>* $)` defines a variable named X. This variable will store any string of letters matching the NooJ regular expression `<L>*`, in which `<L>` means any letter, and the Kleene star (*) means unlimited repetition (Silberztein, 2003:113).

Next, a constraint component pairs an epsilon with output `<$X=:V>`. The `:V` in `<E>/<$X=:V>` indicates a constraint applied to variable X. The equals-colon symbol (=:) limits this rule to entries in the lexicon whose tag begins with V (verb roots). This limits the string stored in variable X to verb roots, rather than just any string of letters. A constraint can be made more stringent by adding further conditions on the lexical entries the rule will apply to, positive or negative (Table 7.2 gives some examples). This constraint mechanism makes NooJ's rules as powerful as a context sensitive grammar (see 6.3.3.4).

Code	Constraint on lexical entries	Examples of matching lexical entries

`<$X=:N>`	Tag begins with N	`bye,N`
		`czar,tsar,N+Human`
		`France,N+Domain=Geography+Country`

`<$X=:N+Human>`	Tag begins with N and contains +Human	`czar,tsar,N+Human`

`<$X=:N-Human>`	Tag begins with N and *does not* contain +Human (thus –Human)	`bye,N`
		`France,N+Domain=Geography+Country`

`<$X=:ALU>`	*Atomic Linguistic Unit* (ALU) is a NooJ term for any lexical entry or rule	`of,PREP`
		`bye,N`
		`czar,tsar,N+Human`
		`France,N+Domain=Geography+Country`
		`go,V`
		`to,PREP`

`<$X=:ALU+Human>`	Tag contains +Human	`czar,tsar,N+Human`
		`think,V+Human`

Table 7.2. Examples of constraint codes and entries they match

The special character hash (#) in a constraint allows a *partial match*[55]. Such a constraint must, however, define the letter(s) to be omitted relative to the string in the variable. For instance, the constraint `<$X#e=:V>` allows *smil* in *smiling* to match the lexical entry `<smile,V>`. The `#e` means that *smil*, when stored in variable X, will match any entry for *smile* as well as *smil* (which does not exist). This allows *smil* in *smiling* to match the lexical entry `<smile,V>`.

[55] At the beginning of a line, a hash is used to specify comments; see 7.2.1.1. In a morphological grammar, the type under discussion here, the hash indicates the application of a partial match to a lexical (dictionary) entry; for futher examples see the explanations of lexical constraints in the NooJ reference documentation (Silberztein 2016:185, 240, 241; Silberztein 2003:117, 132). Finally, in NooJ *syntactic* grammars, which I do not make use of here, the hash is a string concatenation operator (Silberztein 2003:38).

Let us now return to `$(X <L>* $)/<$X=:V> <E>/<$1L,$1C$1S$1F>`. After the constraint is a pairing of epsilon input with output `<$1L,$1C$1S$1F>`. This output is composed using additional variables to represent forms and properties from whatever lexical entry has been matched. This is termed *transfer of features and properties* in NooJ.

A rule's variables (`$1`, `$2`, etc.) are numbered in the order in which they were defined. Subsequent letters specify components from the matched lexicon entry, as follows: L = lemma, i.e. citation form; C = category (the first property, typically a POS category); S = Syntax (other properties, not necessarily syntactic); F = inflection (not used in SANTI-morf).

An output annotation can include additional properties not defined in the matched lexical entry. Thus, in `<$1L,VERB+ROOT>`, the property `VERB+ROOT` is added to the analysis. The `$1L` slot here is filled by the citation form of the entry matched by the string stored in the first variable.

It is also possible to use `$X` to replace `$1L`, as in `<$X,VERB+ROOT>`. This means that NooJ will directly insert whatever string is stored in variable X, without checking the lexicon. This is a feature I use for guessing annotation for unknown words (see 7.4.2).

In NooJ, multiple rules can be placed into a single union, indicating that the rules are alternatives to one another; this is done with the vertical bar symbol (Silberztein 2016:132), the typical representation of alternation ("or") in regular expressions (see 6.3.3.2.1). For example, the *peR—an* allomorphy rule is defined as a union of two sub-rules, one to handle *per—an* allomorph (`:NOU_peR-an_per-an`) and another to handle *pe—an* allomorphs (`:NOU_peR-an_pe-an`).

```
SAMPLE_peR-an  = :peR-an | :pe-an;
```

7.2.2.2 Syntactic grammars

A *syntactic grammar* is saved in a file with extension .nog. Syntactic rules *can* be used to generate syntactic annotation, and disambiguation, but their functions are actually generic across different linguistic levels.

A syntactic rule's input can be composed of 1) spaces, letters, and non-letter symbols (e.g. *bye-bye* or *bye bye*); 2) annotations assigned by the lexicon and morphological rules (e.g. `<N> <N>`); or 3) a combination of 1 and 2 (e.g. `<N> - <N>`). If input type 2 or 3 is used, the text must have previously been tagged by the lexicon and morphological grammar.

In section 6.3.3.2.3, I introduced the Kleene operation (repetition). Silberztein (2016:164-170) notes that applying the Kleene operation to a multi-token pattern allows recursive structures (those with one non-terminal of a given type embedded within another of the same type, not easily represented in regular grammars) to be captured. He uses the fact that left and right recursions can be removed from a context-free grammar to enable NooJ to compile some context-free grammars into finite state transducers; thus NooJ, though based on a regular grammar, supports constructs of context-free grammars (see 6.3.5.3). For instance, the rule `PP = (<PREP> <DET> <N>)*` can capture a recursive prepositional phrase such as *under the tree in the box beside the bottle.* However, this rule would fail to capture *under the tree in the **big** box beside the bottle* – or rather, it would capture only the first three words. While use of repetition in a regular grammar can simulate much of the complexity of recursive rules in context-free grammars, it cannot simulate all of it.

Matching constraints (typically used to check agreement across tokens), *equality constraints* (to identify identical tokens), and *existence constraints* (to check if a token exists) can be added to syntactic rules. This gives them power equal to a context-sensitive grammar (see 6.3.3.4). The MA implemented in 7.4 uses equality constraints extensively to handle reduplication, but does not use matching or existence constraints. An equality constraint ensures that a rule only matches sequences where the forms of two specified tokens in the sequence are identical, as shown below.

```
EqualMultipleToken =                         #[56] rule name
<E>/<RED                                     # tag for multiple input (open)
      $(A <ITJ> $) - $(B <ITJ> $)            # all input tokens
      <E>/<$A_=$B_>                          # equality constraint
 <E>/> ;                                     # tag for multiple input (close)
```

[56] These comments explain the code example. They are not in the actual NooJ code.

The two tokens on either side of a hyphen must be stored in different variables, expressed as `$(A <ITJ> $) - $(B <ITJ> $)`. Then, the equality constraint `<E>/<$A_=$B_>` pairs epsilon with an equality assertion. This constraint causes the orthographic/citation forms (_) of the tokens in variables A and B to be checked for equality (=). *Bye-bye* satisfies this equality constraint, for instance.

Annotation output from syntactic rules can be at the same level at or a higher level than the tokenisation initially given by the lexicon and morphological grammar. To illustrate the latter, in the *EqualTheSameToken* rule below, the annotation of reduplication is composed of two parts, `<E>/<RED` and `<E>/>`, that encompass the rest of the rule. This tags *bye-bye* as one unit at a level above that at which *bye* and *bye* are two separate tokens. This higher-level annotation has two key features: 1) the output only has properties (no orthographic/citation form); and 2) the annotation is added as an additional layer, without overriding the initial annotation. In NooJ, this is termed *add annotation.* An example of an output at the same level as existing annotation is given by the rule shown below.

```
EqualTheSameToken =         # rule name
$(A <N> $) -                # input (no additional tag to give)
       <E>/<$B_,RED         # lexical entry output code for the same token (open)
               ($B <N> $)   # another input (to which additional output is given)
       <E>/>                # lexical entry output code for the same token (close)
<E>/<$A_=$B_> ;             # equality constraint
```

In contrast to the previous example, when annotation is added at the same token level, the output must also be written in standard lexical entry format; therefore `$B_,RED` is used.

Syntactic rules can also be used for contextual or non-contextual disambiguation, via what is termed the *remove annotation* function. Disambiguation rules are based on the *select one and remove others* principle (see 6.7.3). For instance, let us consider the ambiguous annotation of *go* as either a verb or noun root. When NooJ applies this annotation, both possibilities can be viewed in the Text Annotation Structure (TAS), a feature of the NooJ interface which visualises annotation output, as shown in Figure 7.8 and Figure 7.9.

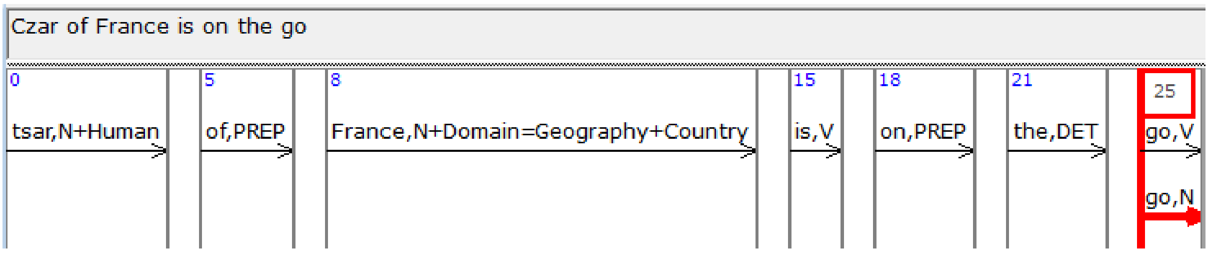

Czar of France is on the go
0
tsar,N+Human
5
of,PREP
8
France,N+Domain=Geography+Country
15
is,V
18
on,PREP
21
the,DET
25
go,V
go,N

Figure 7.8. Ambiguous annotation of *go* (case 1: correct annotation is noun)

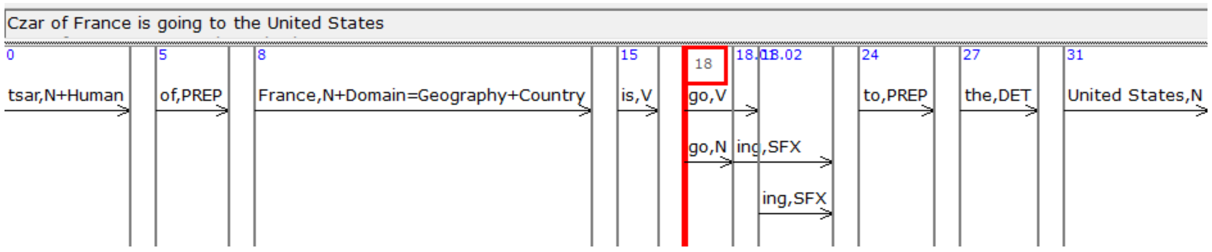


Figure 7.9. Ambiguous annotation of *go* (case 2: correct annotation is verb)

A non-contextual disambiguation rule takes an ambiguously tagged token as input and outputs the correct analysis. An example of a non-contextual disambiguation rule is (7.1).

(7.1) `Non-context = <go>/<V>;`

This syntactic rule removes annotations other than <V> from tokens of *go*. This would cause *go* in either *is going to* or *on the go* to be analysed as a verb – incorrectly, in the latter case. To correctly disambiguate the two possibilities, contextual disambiguation rules are needed.

A contextual disambiguation rule requires the context to be incorporated into the input. Examples (7.2) and (7.3) illustrate this.

(7.2) `Contextual1 =     <go>/<V> <SFX>;`

(7.3) `Contextual2 =     <DET> <go>/<N>;`

The rule in (7.2) applies only when *go* is followed by a suffix, as in *going* or *goes* but not *give it a go*. The rule in (7.3) applies when *go* is preceded by any determiner (*the*, *a*, etc.). They select the verb and noun analyses, respectively. This illustrates the importance of studying ambiguities carefully to determine the correct disambiguation rule type (contextual or non-contextual). This completes my overview of grammar rules as NooJ resources.

7.2.3 Priority configuration for a set of resources

When using a set of multiple resources in NooJ, their priorities must be organised to specify how they work in sequence. As usual in priority systems, once a match is found by any resource, processing stops, so resources of lower priority than that resource are never used. NooJ has two priority sequences: Lexical Analysis (LA) and Syntactic Analysis (SA). By default, resources loaded to LA are applied first. Within LA resources, lexicons and morphological grammars can be given different priority codes (see Table 7.3), determining what order they will be applied in. Syntactic grammar resources are placed into the SA, which means that regardless of priority level, they apply after the LA resources. The priority codes are alphanumeric, H or L plus a number. NooJ allows up to 9 different priority codes to be assigned to resources, the highest being 9 and the lowest 1 for H, and the opposite for L. In the NooJ interface, the LA and SA resources are configured via the priority control panel tool.

Priority	Code (from highest to lowest)
H(igh)	H9 H8 H7 H6 H5 H4 H3 H2 H1
Normal	*No code*
L(ow)	L1 L2 L3 L4 L5 L6 L7 L8 L9

Table 7.3. NooJ's priority codes for LA resources

A set of lexicon and grammar resources loaded into the LA and SA, and given priorities, constitute a NooJ-based annotation system. The configured priorities of the lexicon and grammar resources can be saved as a .noj configuration file. In this project, SANTI-morf consists of its resources plus the SANTI-morf.noj configuration file. Thus, users do not have to manually insert and prioritise the resources one by one. Once the SANTI-morf.noj file is loaded into NooJ, all resources are automatically loaded with the assigned priorities.

7.3 System architecture

SANTI-morf is composed of four *modules,* implemented as a pipeline in NooJ. A module is defined for present purposes as some collection of resources (lexicons and rules) which performs a particular task(s). The four modules of SANTI-morf implement the tasks of the typical tagging procedure described in 6.7.

When a text is loaded to NooJ, NooJ initially breaks it up into words at whitespace and punctuation characters, without annotating it. If any resources have been configured, they are then applied to the text in order (see 7.2.3).

The first SANTI-morf module to process the text after NooJ's default tokenisation is *Module 1: the Annotator*. This set of resources (lexicons and morphological grammars) carries out morphological tokenisation and annotation in a single pass (see 7.4.1). All recognised morphemes are given an analysis at this stage.

The Annotator leaves words without morphological tokenisation/annotation if they are not recognised by the resources; these unanalysed words are termed *unknowns*. MorphInd does nothing with these unknown sexcept simply mark them as <X>, but SANTI-morf takes a different approach. Specifically, *Module 2: the Guesser* (see 7.4.2) applies further morphological grammar resources to make best guesses for all unknowns. This ensures virtually 100% coverage (see 5.5.8).

Some analyses produced by modules 1 and 2 will be incorrect. In particular, they cannot correctly annotate full reduplication (see 7.2.2.2). For these, we need *Module 3: the Improver*, which consists syntactic rules able to recognise sequences of annotated morpheme tokens, and remove or add an annotation for the sequence, or for a particular token. SANTI-morf uses the latter kind of rule to contextually identify morphological errors produced by the Annotator, and to supply the correct annotations.

Module 4: the Disambiguator contains syntactic rules which implement contextual and non-contextual disambiguation techniques. These utilise the *remove annotation* function of NooJ grammars (see 7.2.2.2). The rules in this module are designed to recognise ambiguously annotated tokens, select the correct annotation, and remove incorrect annotations. Unresolved ambiguities are retained in the annotation output, which is presented in the *Task Annotation Structure* (TAS) interface or saved as a .not file (NooJ Text). How the modules fit together is shown in Figure 7.10.

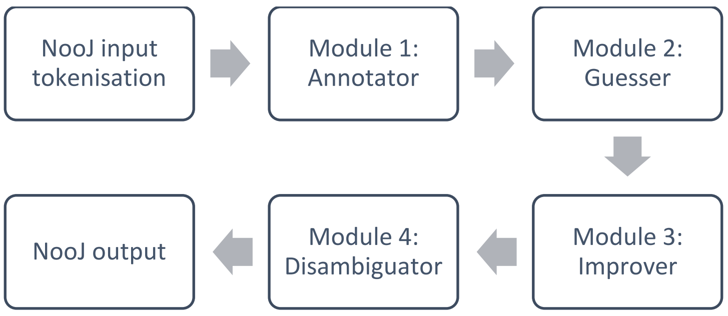


Figure 7.10. Overview of SANTI-morf modules

The aim of the pipeline architecture is to minimise ambiguity, even before disambiguation rules (the Disambiguator) apply. Consider the following illustration. Module 1's lexicon contains `diam,ROOT+ADJ`,[57] an adjective root entry. The Guesser has a guessing rule which analyses any word that begins with *di* as consisting of passive prefix *di-* and a verb root. If these two resources had the same priority, any instance of *diam* 'silent' would be ambiguously analysed as both monomorphemic (by the Annotator) and polymorphemic (by the Guesser). But because the

[57] This is a simplified format. The full format is discussed in 7.4.1.2

Annotator has higher priority than the Guesser, the guessing rule is never applied to *diam*: the lexicon entry pre-empts the guessing rule, and is the only analysis applied.

7.4 Implementation

7.4.1 Module 1: the Annotator

7.4.1.1 Overview

The Annotator performs morphological tokenisation and annotation. It is the core component of SANTI-morf, consisting of three lexicons and four morphological grammars applied in sequence. The resources are named using arbitrary codes that correspond to the resource type.

The lexicon filenames begin with *Dyka*, followed by A (annotator) and the number (1-3) that shows the lexicon's place in the rank order of priority for application. The grammar filenames begin with *Yumi*, followed by A (annotator) and the number (1-4) for that grammar's rank in the priority order. These resources are listed in Table 7.4, and discussed in detail in sections 7.4.1.2 to 7.4.1.8, in order of their overall priority (H9 to H3; see 7.3).

NooJ priority code	Resource label	Resource type	Content	Units analysed
H9	DykaA1	Lexicon	Roots; fully analysed unproductive forms	Monomorphemic words and unproductive polymorphemic words
H8	DykaA2		Proper nouns	
H7	DykaA3		Foreign words	
H6	YumiA1	Morphological grammar	Rules for affixation	Polymorphemic words (non-compound)
H5	YumiA2		Rules for cliticisation	
H4	YumiA3		Rules for affixation for words with two roots (i.e. compounds)	Compound words
H3	YumiA4		Rules for cliticisation for words with two roots	

Table 7.4. List of resources making up Module 1: Annotator

Applying lexicons (DykaA1-A3) before rules (YumiA1-A4) is more reliable than *vice versa*, because the lexicon contains (among others) those forms that are exceptions to rules. For example, prefix *ter-* is always a nominaliser in the word *ter-dakwa*, but this is exceptional; the analysis only exists for that entry in lexicon DykaA1. By contrast, the rules in YumiA1 would analyse *ter-* as a passive marker, which for this word is incorrect. Applying the lexicons *first* means that exceptional cases will be handled *before* any rules are used. DykaA2 precedes DykaA3 to prevent incorrect analysis of proper nouns such as *Apple* (brand name) as monomorphemic foreign words.

The priorities among the grammars are designed to avoid words being overanalysed by splitting up parts of the word which ought not to be split up. Thus, YumiA1 precedes YumiA2 to avoid splitting non-cliticised words which coincidentally begin/end in the letters of a clitic; e.g. *kuras-kan* 'have it drained' must not be analysed as *ku=ras-kan* 'I race-CAUS/APPL', a nonsensical analysis. Similarly, the rules for compound words *without* clitics (YumiA3) must be prioritised over the rules for compound words *with* clitics (YumiA4).

Likewise, affixed words with one root must not be overanalysed as affixed compound words with two roots in cases where both are possible (e.g. *pen-didik-an* 'education' must not be analysed as *pen-di-dik-an* 'at younger sibling (nominalised)', a nonsensical analysis). Thus, YumiA1 (affixation rules for single-root words) is prioritised over YumiA3 (affixation rules for compound words). In parallel to all this, the rules for affixation of single-root words with clitics (YumiA2) are prioritised over the rules for affixation of compound words with clitics (YumiA4).

7.4.1.2 DykaA1.nod (H9): the core lexicon

While I wrote all the rules from scratch during the development of SANTI-morf, I compiled the lexicons from reformatted versions of various pre-existing resources for Indonesian. The lexicon DykaA1 (Dyka = arbitrary code for lexicons, A = Annotator, 1 = 1st in order) is the highest priority resource, not only in the Annotator, but across all SANTI-morf resources. This lexicon implements tokenisation and annotation of monomorphemic words (11,546 entries) and of a handful of polymorphemic words (130) which are exceptions to the rules in the lower-priority grammar resources.

The list of roots was initially derived from a lexicon database[58] extracted from Kateglo[59] an online Indonesian dictionary. Each line in that lexicon contains a single root which can be a monomorphemic word (fully lowercase), followed by the word's POS category in Indonesian (fully uppercase), e.g.:

```
makan VERBA
minum NOMINA
```

Manual inspection of the Kateglo lexicon showed it to include 1) contemporary roots, 2) archaic roots, 3) polymorphemic words derived by *unproductive* processes, 4) polymorphemic words derived by productive processes, and 5) non-Indonesian words. Of these, the contemporary roots and unproductively-formed words were incorporated into DykaA1. I excluded a number of archaic roots because their inclusion caused other analyses to be incorrect. For example, when SANTI-morf scans *bad* in *this is a bad day,* it should be analysed as FRG (foreign root). However, in the Kateglo lexicon, *bad* is categorised as a noun because it is an archaic root meaning 'wind'. If this archaic form along with its POS category were included in DykaA1, SANTI-morf wouldincorrectly analyse *bad* as a noun instead of a foreign word. Polymorphemic words formed by productive processes are also excluded, because they can be analysed by the morphological rules (see 7.4.1.5). I transferred the selected parts of the Kateglo lexicon to a spreadsheet. To add the tags of the SANTI-morf MAS, an additional column was inserted and automatically filled with ROOT as the first property, followed by the POS.

Kateglo lexicon:	`tabrak       Verba`
DykaA1:	`tabrak,ROOT+VER+PS+AT+T1+DykaA1`

I mapped the Kateglo POS codes to the corresponding codes in the SANTI-morf MAS (e.g. *adverbia* to *ADV*) using global search-and-replace. Incorrect POS categories (around 150) were corrected. To the resulting initial SANTI-morf lexicon, I added entries for 396 roots and 58 words

[58] https://datahub.io/aps2201/kateglo_scrape#resource-kateglo_scrape_zip (last accessed 24/05/2021)
[59] https://kateglo.com/ (last accessed 24/05/2021)

based on my own observations of Indonesian texts. DykaA1 contains in all 11,546 entries for roots and 130 for complete words.

Following ROOT and the POS in the entries are a number of implementation properties. Unless specified otherwise, these were inserted manually. The first implementation property is *syllable*. Each root is coded either +MS (monosyllabic) or +PS (polysyllabic). These features are used to constrain allomorphy rules. For instance, monosyllabic roots co-occur with prefix allomorphs not used with longer words (e.g. active prefix *menge-,* agentive/instrumental nominaliser *penge-*).

Next*,* I added a property called *first letter*, which contains the initial letter of the root. The format is *A* (meaning initial), followed by the first letter of the root in uppercase; this property is added automatically. Thus, the root *ramai* 'crowded' has +AR because *ramai* begins with *r*. Like *syllable*, this property is also used to constrain allomorphy rules. For example, the superlative prefix *te-*, an allomorph of *teR-*, occurs with roots beginning with *r*.

The next property is *transitivity*, also for rule constraints. For instance, circumfix *ber—an* typically has reciprocal function when it occurs with transitive verb roots (+T1). Thus, the rule for the reciprocal circumfix is constrained to apply only with roots that have +T1. Other values are intransitive (+T0), ambitransitive (+T2), and non-verb (+TX). While I created this property for implementation purposes (and, therefore, it is not part of the MAS in chapter 4), nothing prevents end-users from using it for analytic purpose, such as to retrieve verbs on the basis of their root's transitivity.

The final property present on entries in DykaA1 is the name of the lexicon, +DykaA1. This is used for debugging purposes, and is inserted at the end of each entry. In all, the annotation in the lexical entry for a root in DykaA1 consists of a form followed by seven properties. Two of the seven are analytic annotation (ROOT and the POS) while the others exist for sake of the implementation. Lexical entries for polymorphemic words produced by unproductive processes have no implementation properties, because these entries are exceptions to the rules which utilise those properties. As exceptions, all entries for polymorphemic words are given `+UNAMB` at the end of the annotation, which makes them the highest priority in analysis. The format of entries for non-root morphemes is fully discussed in 7.4.1.5. The analyses for the morphemes making up one word are laid out following the order of the morphemes, as the following example for *jejaring* 'many webs' (formed by partial reduplication; see 4.2.4) shows:

```
jejaring,<je,RED+PART+NOU+DykaA1><jaring,ROOT+NOU+DykaA1>+UNAMB
```

The format of the analysis depends on the morphological structure of the word. In addition to root and word forms, the lexicon also includes non-letter symbols. These symbols are annotated in the lexicon with the POS (DGT for numbers and PNC for non-letter symbols) and the resource name. This type of entry has no implementation properties, as the entry for the pound sterling symbol illustrates:

```
£,PNC+DykaA1
```

Text	Analysis
tabrak	`tabrak,ROOT+VER+PS+AT+T1+DykaA1`

jejaring	`je,RED:PART+DER:NOU+DykaA1`
	`jaring,ROOT+NOU+DykaA1`

sosial	`sosial,ROOT+NOU+PS+AS+TX+DykaA1`

Table 7.5. A sample of words morphologically annotated using DykaA1

7.4.1.3 DykaA2 (H8): the proper noun lexicon

The lexicon DykaA2 (Dyka = lexicon, A = Annotation, 2 = 2nd in order) is designed to analyse proper nouns. The creation of this lexicon was motivated by MorphInd's failure to identify many proper nouns, which was one major factor reducing its coverage (see 5.5.7). I sourced the lexical entries in DykaA2 from various resources, as detailed in Table 7.6.

Data source (number of resulting entries)	Origin
BNC wordlist (7,764)	Leech and Smith[60] (2000)
Names of provinces, cities, districts, and villages in Indonesia (45,149)	*edwardsamuel*[61]
Names of countries and capital cities (266)	*kata-ai*[62]
Collection of person names in Indonesia (6932)	*seriously*[63]

Table 7.6. Existing resources used to build DykaA2

The first letter of each entry's form was changed to uppercase, i.e. *apple, blue, moon* to *Apple, Blue, Moon*. The 544 proper nouns acquired from the BNC wordlist were already in this format, e.g. *James, London, Lancaster*. This anticipates use of the words in this lexicon as proper nouns created from English common nouns such as *Apple* (name of a company) or *Blue* and *Moon* in *Blue Moon* (name of a restaurant in Lancaster). This is only the correct analysis if the word has an initial capital. All words in DykaA2 are analysed as noun roots, e.g. `London,ROOT+NOU+DykaA2`, because they are all considered unanalysable proper nouns in Indonesian. This lexicon is not exhaustive of all possible proper nouns. Proper nouns not matched in this lexicon will instead be handled by guessing rules, discussed in section 7.4.2.

Text	Analysis
restoran	`restoran,ROOT+NOU+PS+AR+TX+DykaA1`
Blue	`Blue,ROOT+NOU+DykaA2`
Moon	`Moon,ROOT+NOU+DykaA2`
Lancaster	`Lancaster,ROOT+NOU+DykaA2`

Table 7.7. Some proper nouns originating as foreign words (in the name *Blue Moon Lancaster*), as annotated by DykaA2

7.4.1.4 DykaA3.nod (H7): the foreign word lexicon

Lexicon DykaA3 contains entries for analysis of foreign words. The foreign language data that I used to compile DykaA3, and their sources, are given in Table 7.8. Regardless of their POS, all these words are assigned the `+FRG` (foreign) property. There is, however, no indication of the

[60] http://ucrel.lancs.ac.uk/bncfreq (last accessed 26/06/2020)
[61] https://github.com/edwardsamuel/Wilayah-Administratif-Indonesia(last accessed 26/06/2020)
[62] https://github.com/kata-ai/kawat/blob/master/semantic/country-capitals.txt (last accessed 26/06/2020)
[63] https://github.com/seuriously/genderprediction/blob/master/namatraining.txt (last accessed 26/06/2020)

original language. An example full entry in this lexicon is `copy,ROOT+FRG+DykaA3`. No uppercase conversion is applied.

Foreign language (number of entries)	Derived from
English (7,764)	BNC wordlist
Javanese (2,394)	Javanese Wiktionary[64]
Dutch (4,617)	Dutch Wiktionary[65]

Table 7.8. Foreign word lexicon sources

Text	Analysis
blue	`blue,ROOT+FRG+DykaA3`
line	`line,ROOT+FRG+DykaA3`

Table 7.9. A two-word foreign phrase (*blue line*) annotated by DykaA3

7.4.1.5 YumiA1 (H7): the rules of affixation

7.4.1.5.1 Preliminaries

YumiA1 (Yumi = arbitrary code for morphological grammar resources, A = Annotation, 1 = 1st in order) is a resource consisting of rules for tokenisation and annotation of single-root polymorphemic words formed by means of productive affixation.

Unlike the lexicons, where a large proportion of the content was created by reuse from existing resources, all the SANTI-morf grammar files were written from scratch. Although rules are available in the code of PMA (Pisceldo et al. 2008) and MorphInd (Larasati et al. 2011), they do not adhere to the MAS that SANTI-morf implements, and are therefore not re-usable in the present project.

Just as with the definition of the MAS (Chapter 4), I used two Indonesian reference grammars (Alwi et al. 1998; Sneddon et al. 2010) as my main sources for writing these rules. Supplementary resources were Indonesian morphology textbooks (e.g. Kridalaksana 1989;

[64] https://id.wiktionary.org/wiki/Lampiran:Kamus_bahasa_Jawa_%E2%80%93_bahasa_Indonesia (last accessed 25/05/2021)
[65] https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/Dutch_wordlist (last accessed 25/05/2021)

Kridalaksana 2007; Chaer 2008); my own observation of testbed texts; and my introspection as a native speaker of Indonesian.

My convention for names of affixation rules is that they combine the output category of the affixation with the form of the affix in question. For instance, `VER_di-` is the rule for *di-* prefixation, whose outcome is a verb. Names of rules for polysemous affixes (see 4.1.15) end in (an abbreviation of) the applicable function, yielding for instance `VER_-kan_caus` (causative *-kan*) alongside `VER_-kan_appl` (applicative *-kan*). Other rule components will be covered in 7.4.1.5.2 to 7.4.1.5.4; how they are aggregated into complete rules is discussed in section 7.4.1.5.5; how the rules are organised within YumiA1 is discussed in 7.4.1.5.6.

7.4.1.5.2 Input-output codes for affixes

The input-output codes for each affix are written directly in the rule, without referring to the lexicons (see 7.2.1 and 7.2.2). The input is the orthographic form. The output has the following components; 1) orthographic form, 2) input-output delimiter (slash), 3) formal morphological criteria, 4) form-property delimiter (comma), 5) citation form (if different from orthographic form), 6) opening/closing property (only for circumfixes), 7) outcome POS property, 8) other functional and/or implementation properties (if any), 9) rule number in the format *`RL=n`*, and 10) the resource name `+YumiA1`. Examples are given in Table 7.10.

Not all 10 components must be present. For example, the input-output codes for *meng-* and *-kan* in Table 7.10 lack the opening/closing property, because they are not circumfixes. There is no citation form for *-kan*, because it is the same as the orthographic form. There are no rules for infixation, because infixation is unproductive and so all words with infixes are listed in lexicon DykaA1.

Input-output codes for circumfixes are in two parts. Although they resemble two separate affixes, circumfixes should be considered as single discontinuous morphemes (see 2.1.3.4.2.1 and 4.2.3.1). To indicate this, the additional property `+A` or `+Z` is added to respectively the opening and closing part of a discontinuous element.

Prefix	`meng/<meng,PFX+meN+DER:VER+ACV+RL=112304+DykaA1>`
Suffix	`kan/<kan,SFX+DER:VER+CAUS+RL=112200+DykaA1>`
Circumfix (opening)	`ke/<ke,CFX+A+DER:NOU+RL=060800+DykaA1>`
Circumfix (closing)	`an/<an,CFX+Z+DER:NOU+RL=060800+DykaA1>`

Table 7.10. Sample input-output codes from rules for affixes

For a polysemous affix, the input-output codes for all analyses must be present in the affix's rule. The input-output codes for the various analyses are joined in a union by the `|` symbol (pipe, see 7.2.2.1), which means *or.* For example, *-kan* can function as causative (+CAUS) or applicative (+APPL) and these two possibilities are expressed by the following union within the rule for *-kan*:.

```
(
kan/<kan,SFX+DER:VER+CAUS+RL=112100+YumiA1 >
| kan/<kan,SFX+DER:VER+APPL+RL=112100+YumiA1 >
)
```

Both analyses from the union will be applied; the Disambiguator module (see 7.4.4) will later remove the incorrect analysis so that only the most likely to be correct analysis remains.

7.4.1.5.3 Root component: variable and output

I use the term *root component* to refer to the codes within an affixation rule that define the slot that the root would have to occupy in a word formed by the rule in question. Technically, it targets a sequence of letters in a word being processed, which will be treated as the possible root and be looked up in the list of roots in the lexicon. Once a match is found in the lexicon, the corresponding lexical entry is used as the annotation output for the root morpheme token. Thus, in a morphological rule, the root component is not the actual root, but rather a cluster of information required to identify and annotate the actual root.

The input-output code for each root component includes *variable, matching constraint,* and *output* components. The variable which stores the string to be analysed as the root is always named X. An example of this (given previously in Table 7.2) is the code `$(X <L>* $)`. This indicates that variable X is defined to be a collection of letters `<L>*`, and thus will store whatever root happens to be found within a polymorphemic word being processed by the rule.

Likewise, the output, marked by `<E>/<$1L,$1C$1S$1F>`, is the same for (nearly) every rule. Each $ indicates an element of the output to be pulled in from a lexicon entry. $L pulls in the form in the lexicon to be used as an annotation output, and $C, $S and $F pull in the annotation properties (*category*, *syntactic and semantic features*, and *inflectional features* respectively. The prepended digit 1 indicates that the lexicon entry from which material will be drawn is the one matched by the string stored in the *first* variable (i.e. variable X). In general, there is only one variable in eacj rule, except for compounds; since this variable stores the material at the position where the root is expected, it should in theory match the lexicon entry for that root.

As these two codes (variable and output) are recurrently re-used, I encapsulate them as non-terminals `varX` and `out1`: `varX = $(X <L>* $)` and `out1 = <E>/<$1L,$1C$1S$1F>`. The actual rules then only contain the names *varX* and *out1*. A minor variation on *out1*, *out1L*, is defined for roots that can undergo initial consonant deletion (see discussion and examples at 7.4.1.5.5); its output code is the same as *out1*'s except for the additional property `+LOST`.

7.4.1.5.4 Root component: matching constraint

The term *constraint* was defined in section 7.2.2.1. A constraint affects the lookup of the variable, usually X, on which that constraint is placed. Unlike the variable and output, the constraint components (see Table 7.2) vary from one affixation rule to another; thus, they cannot be simplified in the way used above for the variable and output.

In the YumiA1 rules, all constraints include the property `+ROOT`, so that the string in the rule's variable can only match lexical entries tagged as roots. Otherwise, there are three types. Lexical property constraints (L) specify lexical properties which matched roots must possess for the rule overall to apply (e.g. the root's number of letters or its first letter, or features such as transitivity)[66]. For example, `<$X=:ROOT+ADJ+AR>`, a constraint that is part of the *te-* affixation rule (given in Table 7.11), ensures that only roots whose first letter is `r`, such as *rendah* 'low' (cf. *terendah* 'lowest'), are acceptable as matches for the string stored in this rule's variable.

Partial match constraints (P) restrict the lookup based on orthographic form. Adding such additional constraints based on (part of) the orthographic form is necessary to handle the morphophonemic alternation of roots after certain prefixes (see 7.2.2.1). For example, `#s` in the P constraint `<#s$X=:ROOT>` asserts that the variable will match a lexical entry whose form is the string in the lexicon entry *with letter s prepended* (as long as that entry is tagged with ROOT). Thus, when processing the word *meny-apu* 'sweep (a floor)', the affixation rule with this constraint stores *apu* in the variable, and the partial match constraint enables the variable to match the lexical entry for root *sapu* 'broom', whose initial *s* is lost after *meny-*. The third type of constraint combines the former two (L+P). Some examples of constraints are given in Table 7.11; in this project, I use only constraints of types L and L+P.

[66] Constraints of this type can speed up NooJ processing, because they remove the need to process each lexical entry for every word-token.

Constraint		Rule using constraint	Description of constraint and example matching lexical entries
Type	Code		

L	`<$X=:ROOT+T1-AR>`	Reciprocal circumfix *ber—an*	Matched entry must be a transitive verb root which does not begin with *r* e.g. *ber-**pukul**-an* 'hit one another'.
			
			`pukul,ROOT+FULL+VER+L5+AP+T1+DykaA1`
			`cium,ROOT+FULL+VER+L4+AC+T1+DykaA1`
			`peluk, ROOT+FULL+VER+L5+AP+T1+DykaA1`

L	`<$X=:ROOT+T0-AR>`	Random action circumfix *ber—an*	Matched entry must be an intransitive verb root which does not begin with *r* e.g. *ber-**jatuh**-an* '(multiple things) to fall randomly'.
			
			`jatuh,ROOT+FULL+VER+L5+AJ+T0+DykaA1`
			`gugur,ROOT+FULL+VER+L5+AG+T0+DykaA1`
			`lari,ROOT+FULL+VER+L4+AL+T0+DykaA1`

L	`<$X=:ROOT+L3>`	Active verb prefix *menge-*	Matched entry must be a monosyllabic root e.g. *menge-**bom***
			
			`las,ROOT+FULL+VER+L3+AL+TX+DykaA1`
			`bom,ROOT+FULL+NOU+L3+AC+TX+DykaA1`
			`peluk, ROOT+FULL+NOU+L5+AP+TX+DykaA1`

L	`<$X=:ROOT+ADJ+AR>`	Superlative adjective prefix *te-*	Matched entry must be an adjective root that begins with *r* e.g. *te-**ramah*** 'friendliest'
			
			`ramah,ROOT+FULL+ADJ+L5+AR+TX+DykaA1`
			`rendah,ROOT+FULL+ADJ+L6+AR+TX+DykaA1`

L+P	`<#s$X=:ROOT>`	Active verb prefix *meny-*	Matched entry must be a root that begins with *s*, but that *s* is absent in the orthographic form e.g. *meny-**(s)apu*** 'sweep (a floor)'
			
			`sapu,ROOT+FULL+NOU+L4+AS+T0+DykaA1`
			`sapa,ROOT+FULL+VER+L4+AS+T1+DykaA1`
			`saring,ROOT+FULL+VER+L5+AS+T1+DykaA1`

Table 7.11. Some constraints (L = lexical property constraint, P = partial match constraint)

7.4.1.5.5 The rules in full

Each full rule is composed of an affix component and the root component (including the root's matching constraint if any) (see 7.4.1.2). Components must be ordered correctly (e.g. a suffix component must follow a root component, not the other way around) because NooJ processes components sequentially from left to right. Rules in YumiA1 capture seven patterns of affix/root combinations (see Table 7.12).

	Basic affixation rule pattern	Example
1	prefix-root	*di-ambil* 'be taken
2	root-suffix	*cuci-kan* 'wash (causative)'
3	prefix-root-suffix	*di-cuci-kan* 'be washed (causative)'
4	circumfix(open)-root-circumfix(close)	*ke-merah-an* 'reddish'
5	prefix-circumfix(open)-root-circumfix(close)	*ber-pe-rasa-an* 'have feelings'
6	circumfix(open)-prefix-root-circumfix(close)	*ke-pe-muda-an* 'youthhood'
7	prefix-prefix-root	*di-per-besar* 'be made bigger/magnified'

Table 7.12. Affixation rule patterns (patterns are in true order rather than the order in the actual code)

The complete rule for circumfix *pen—an*, `NOU_peN-an_pen-an`, illustrates the overall composition, here following pattern 4. After the rule name, the input-output code for the circumfix consists of two elements: `pen/<pen,CFX+A+peN+DER:NOU+RL=060602>` and `an/<an,CFX+Z+DER:NOU+RL=060602>`. These respectively precede and follow the part of the rule specifying the possible roots, i.e. `:varX ( <E>/<$X=:ALU>) :out1` , itself in three parts (variable, constraint, and output, in that order), yielding:

```
NOU_peN-an_pen-an =
pen/<pen,peN,CFX+A+DER:NOU+RL=060602>
:varX ( <E>/<$X=:ALU>) :out1
an/<an,CFX+Z+DER:NOU+RL=060602> ;
```

7.4.1.5.6 Organisation of rules

In all, there are 121 rules in YumiA1. The main rule is a union of 10 non-terminals, named after the possible word-level outcome POSs, e.g. :ADVERBIA and :AJEKTIVA. The list of relevant POS classes here (see Table 4.7 in 4.2.2) excludes some generally accepted categories, for instance interjections, which never take affixes and are therefore not affected by any of the rules.

Each of these non-terminals is itself a union of the non-terminals for the rules that produce that union's POS. Then come the rules themselves, before the union for the *next* outcome POS begins. Thus, all 121 rules are conceptually grouped based on outcome POS. Over half the rules characterise affixations whose outcome POS is verb (62 rules); next most common are rules deriving nouns (25).

```
Main = :ADVERBIA| … |:NOMINA| ….| :VERBA  ; # union of rules based on outcome POS
ADVERBIA =   ADV_se-an | …                  # union of adverb outcome rules
VERBA = :VER_beR- | .. | :VER_meN-          # union of verb outcome rules
VER_meN- = ….|:VER_meN-_meng-|…             # union of allomorphy rules for meN-
VER_meN-meng-                               # actual rule for meng-
```

Each output code for an affix includes a rule number (e.g. RL=112304 is the number of the *menge-* prefix rule; see 7.4.1.5.2), which reflects the organisation of YumiA1 based on outcome POS and is used for debugging purposes.

Text	Analysis
kesatuan	ke,CFX+A+DER:NOU+RL=060800+YumiA1 satu,ROOT+NOU+PS+AS+TX+ +DykaA1 an,CFX+Z+DER:NOU+RL=060800+YumiA1
makna	makna,ROOT+NOU+PS+AM+TX+DykaA1

Table 7.13. Annotation of *kesatuan* as a combination of root *satu* and circumfix *ke—an* using YumiA1

Some rule-based annotation systems operate by attempting to apply rules to the token under analysis one at a time, in the order that those rules occur in the grammar file. Once such a system finds a rule that applies to the current token, it stops, and accepts that rule's analysis as correct. The remaining rules are not even tested. This means that care must be taken to put the rules in an order that results in rule conflicts being resolved correctly. Fortunately, this is not a problem with NooJ. In NooJ, all rules are attempted, and if more than one rule applies to the token at hand, the token will receive all resulting analyses, and be ambiguously annotated. While NooJ does support prioritisation of resources relative to one another, this sequencing only takes effect if the user explicitly and intentionally introduces it to the system configuration (see 7.2.3). Otherwise, rule sequencing in NooJ grammar files does not affect the outcome.

7.4.1.6 YumiA2 (H6): rules for affixation involving clitics

The YumiA1 rules cannot analyse affixed words which also have clitics, e.g. *mem-*

bawa=ku 'take me'. Such words are handled by YumiA2 (A= Annotation, 2 = 2nd in order), a morphological grammar containing rules for cliticisation.

Discussing clitics, reference grammars of Indonesian typically focus on whether each clitic precedes or follows its host, without discussing in detail what constraints might restrict their combination. By contrast, affix combination is dealt with rigorously in this literature. My MAS distinguishes numerous categories of clitic (see 4.2.2.2 and 4.2.6). Rules exist for sets of two proclitics (one pronoun and one numeral, handled by rule PCLT); three enclitics (three pronouns, handled by rule ECLT1); and four word-final enclitic-like particles (handled by rule ECLT2). Since clitics are never compulsory for any word-form, they must be specified as optional (by being in union with epsilon `<E>`); this is implemented in the `Main` rule, which specifies all possible combinations of clitics:

```
Main         = (:PCLT|<E>)      :WORD          (:ECLT1|<E>) (:ECLT2|<E>);
# main rule  optional proclitic  obligatory host  optional enclitic(s)
```

The non-terminal `WORD` is a union of a copy of all the rules from YumiA1, so that this element, representing the 'host' of the clitic, may match any recognised affixed word. Affixes and clitics have the similarity that they are bound in some manner to other units. But their differences, discussed in 2.1.3.4.2 and 2.1.3.4.5, are reflected in the corresponding rule output codes. Notably, the output code for a clitic output code does not have an outcome POS property (see 7.4.1.5.2). Instead, it has a POS, just like a root. This is because a clitic *is* a monomorphemic word in itself, attached only phonetically/orthographically. This is clear in the citation forms defined for clitics, which are identical to the equivalent independent word that has been cliticised, as Table 7.14 shows. The output codes for clitics include a POS property PRO (pronoun) instead of an outcome POS property DER:PRO.

Text	Analysis

jawaban=ku 'my answer'	`jawab,ROOT+VER+PS+AJ+T1+DykaA1`
	`an,SFX+DER:NOU+RL=060500+YumiA2`
	`ku,aku,ROOT+ECLT+PRO+YumiA2`

benar 'correct'	`benar,ROOT+ADJ+PS+AB+TX+DykaA1`

Table 7.14. Annotation of enclitic pronoun *=ku* 'pronoun' in *jawab-an=ku* 'my answer' using YumiA2

The input-output code for a clitic consists of 1) orthographic form, 2) citation form, 3) delimiter, 4) the tag for the morpheme type, which is ROOT plus either proclitic +`PCLT` or enclitic +`ECLT`), 5) POS of the clitic (e.g. PRO), 7) functional tags (if any) and 8) the resource name, for debugging. A full example is `<E>/<ku,aku,ROOT+PCLT+PRO+YumiA2>` for proclitic *ku=*. As usual, if orthographic and citation forms are identical, the latter is omitted.

The enclitic pronoun *=nya* is ambiguous with definite suffix *-nya*; this ambiguity can only be resolved by pragmatic context (see 4.2.2.2). While it is beyond this project to incorporate pragmatic context, there do exist some exceptional cases in which disambiguation is possible (see 4.2.6). For this reason, distinct tags for clitic *=nya* and suffix *-nya,* and also +NYA tag are preserved at this stage.

7.4.1.7 YumiA3 (H6): rules for affixation involving two roots

YumiA3 contains rules to analyse compounds (two roots), with or without affixation. It began as a full copy of YumiA1. I then changed each rule to add input-output codes to match with the second root of the compound. The variable component for the second root is given the label Y – not X – thus: `$(Y <L>* $)`. The output component references this variable as 2 instead of 1 because Y is the second variable in the rule. This yields <`E>/<$2L,$2C$2S$2F`>. The constraint for the second root is only `ROOT`, i.e. `<E>/<$Y=:ROOT>`, since all constraints based in, for instance, allomorphy, are tested against the form or properties of the first root, not the second. Because the same input-output codes (variable, constraint, annotation output) are used to represent second roots as are used for a first or sole root, I defined a reusable non-terminal named `2root`:

```
2root = $(Y <L>* $) <E>/<$Y=:ROOT> <E>/<$2L,$2C$2S$2F>;
```

The non-terminal `2root` is then added to the YumiA3 version of the *per—an* affixation rule (`NOU_peR-an_per-an`), given below. This allows the rule to analyse the affixed compound word *per-tanggung-jawab-an* 'responsibility', with *tanggung* 'carry' analysed by `:varx <E>/<$X=:ROOT> :out1`, and *jawab* 'answer' analysed by `:2root`.

```
NOU_peR-an  = :NOU_peR-an_per-an | :NOU_peR-an_pe-an;


NOU_peR-an_per-an =
per/<per,peR,CFX+A+per_an+peR_an+DER:NOU+RL=060701A+YumiA1>
:varX <E>/<$X=:ROOT+DykaA1> :out1 :2root   an/<an,CFX+Z
+DER:NOU+RL=060701Z+YumiA1>;
```

In these rules, affixes are optional, as it is possible for a compound word just to be composed of two adjacent roots, as in *kaca-mata* 'eye glasses', from *kaca* 'glass' and *mata* 'eye', whose analysis is shown in Table 7.15.

Text	Analysis
jual	`jual,ROOT+VER+PS+AJ+T1+DykaA1`

kacamata	**`kaca,ROOT+NOU+PS+AJ+T1+DykaA1`**
	`mata,ROOT+NOU+PS+AM+TX+DykaA1`

pria	`pria,ROOT+NOU+PS+AM+TX+DykaA1+YumiA3`

Table 7.15. The annotation of a compound *kaca-mata*

7.4.1.8 YumiA4 (H6): rules for affixation involving two roots and clitics

Grammar YumiA4 contains rules to analyse affixed two-root compounds which also have clitics. Its main rule is identical to that of YumiA2, which analyses words with clitics. However, the non-terminal symbol `:WORD`, used inside the main rule, is defined differently than in YumiA2. Here, `:WORD` is defined as a union of rules copied from YumiA3 (rules for affixation of compounds), rather than as a union of YumiA1 (rules for affixation of non-compounds). As usual this grammar adds a distinct resource name property to every rule

Text	Analysis
pertanggungjawabanmu	`per,CFX+A+peR+DER:NOU+RL=060701A+YumiA4`
	`tanggung,ROOT+VER+PS+AT+T1+DykaA1`
	`jawab,ROOT+NOU+PS+AJ+T1+DykaA1`
	`an,CFX+Z +DER:NOU+RL=060701Z+YumiA4`
	`mu,ROOT+ENCLT+kamu+PRO+YumiA4`

Table 7.16. The annotation of *per-tanggung-jawab-an=mu* as compound *tanggung-jawab* plus circumfix *per—an* plus enclitic *=mu* by rules and lexicon entries within the Annotator

7.4.2 Module 2: the Guesser

The Guesser module implements morphological tokenisation and annotation of *unknown words*, i.e. all words that were not analysed by the Annotator. Rather than label all unknown words with a single "unknown" tag, like MorphInd, SANTI-morf always generates a "best guess".

It is crucial not to let the level of ambiguity rise out of control at this early stage. Thus, the Guesser is designed to produce the minimum ambiguity possible. The Guesser consists of sets of morphological rules, labelled *Yumi* like the other grammar resources, followed by G (for Guesser) and a number (from 1 to 6) that indicates the resource's relative position in the order of priority.

The Guesser rules are based on affixation, cliticisation, and orthographic cues. Affixes and clitics are recognised by position-based rules. For instance, the non-word *diaambil* is not recognised by the Annotator, but the Guesser has a rule which analyses words beginning with *di* as consisting of *di-* as a prefix, and whatever comes after it, here *aambil,* as a root.

Many affixes (unlike clitics; see 4.2.2.2) are not exclusive to one root POS; for instance, *ter-* can appear on an adjective, noun, or verb. In such cases, the Guesser assigns the analysis with the greatest frequency. This frequency information was extracted from the testbed as processed by the Annotator only. *Ter-* is most often a verbal prefix, and so the Guesser's rule for *ter-* only assigns that analysis. The same procedure applies to the root; since *ter-* is most frequently followed by a verb root, the rule treats whatever follows *ter-* as a verb root.

The Guesser's affixation rules differ from those in the Annotator. First, the rules in the Guesser *do not have any constraints*, because no reference needs to be made to the lexicon. Second, the annotation output is *explicitly* part of the rule, rather than being a reference to a lexicon entry; two rules with the explicit output code `<$X,ROOT+FULL+VER+YumiG1>` are discussed below.

The Guesser also uses orthographic cues. For instance, an unknown word with an initial uppercase followed by lowercase letters is analysed as a noun (because it is likely to be a proper name).

The priority of the Guesser resources begins with the more reliable rules, to avoid ambiguity. Affix-recognising rules are prioritised over rules using orthographic cues, the former being more reliable. Thus, the orthographic cues resource, YumiG6, is given very low priority (L4: see Table 7.9). Further, rules that recognise both a prefix and a suffix (or a circumfix) have priority over rules that recognise just one affix, since an unknown word is less likely to spuriously match *two* affixes. Thus, a word that matches the patterns for *ber—kan* (circumfix), *ber-* (prefix), and *-kan* (suffix) will be handled by the rule for *ber—kan*, pre-empting the other two.

The Guesser is designed to recognise the longest possible affix, rather than any substring which is also a possible affix. For instance, the rule for *meng-* has priority over the rule for *men-*. To implement this prioritisation, NooJ's +`UNAMB` feature (see 7.2.1.2) is used, as these rules illustrate:

```
VER_meng—kan=
meng/<meng,PFX+DER:VER+YumiG1>
$(X <L>* $)
<E>/<$X,ROOT+FULL+VER+YumiG1>
kan/<kan,SFX+DER:VER+YumiG1>
+UNAMB
;


VER_men—kan=
men/<men,PFX+DER:VER+YumiG1>
$(X <L>* $)
<E>/<$X,ROOT+FULL+VER+YumiG1>
meng/<meng,PFX+DER:VER+YumiG1>
;
```

Although the +`UNAMB` feature can be used multiple times within a grammar, it only expresses a single distinction in priority level. In some cases, however, more than one distinction is needed (e.g. recognising the allomorphs of *meN-* requires rules across three priority levels). For this reason, the morphological-cue rules are split across multiple resources to create additional priority levels (two per resource). The full list of Guesser rules is given in Table 7.17.

Resource	YumiG1	YumiG2	YumiG3	YumiG4	YumiG5	YumiG6
NooJ priority	H2	H1	L1	L2	L3	L4
Affix(es) or form(s) recognised by resource's rules	meng—kan* meny—kan* men—kan mem—kan	me—kan*		meng- meny- men- mem-	me-	Fully-lowercase form with clitics (e.g. *kimchi=ku*) Fully-lowercase form (e.g. *kimchi*) Fully-uppercase form (e.g. *CHAPTER*) Initial uppercase form (e.g. *Sensei*) Mixed case (e.g. *EduTainment*)
	ter—kan* te—kan			ber-* be- per-* ter-* te-		
	per—kan* pe—kan			peng-* peny-* pem- pen-	pe-	
		ber—an* be—an				
		peng—an* peny—an* pem—an pen—an per—an	pe—an			
		ke—an				
	di—kan	se—nya				
	All rules above recognise the following optional clitics *ku=* *=ku* *=nya* *=mu* *=lah* *=pun* *=kah*					

Table 7.17. Rules and their priorities in the Guesser (* = prioritised using +UNAMB)

The final resource, YumiG6, contains rules based on the aforementioned orthographic cues, and in addition, rules which analyse all remaining unknown words as either roots (not prioritised) or roots with clitics (prioritised with +UNAMB). For instance, the remainder of a word after proclitic *ku=* is always analysed as a verb root; this is correct in more than 90% of cases in the Annotator-processed testbed:

```
ku_VER_ROOT =
ku/<ku,ROOT+PCLT+PRO+YumiG6>
$(X <L>* $)
<E>/<$X,ROOT+FULL+VER+YumiG6>
+UNAMB;
```

The remainder of any word recognised as ending in an enclitic is analysed as a noun root; this is correct in more than 70% of cases in the Annotator-processed testbed. The remaining unknown words, lacking clitics, are guessed to be monomorphemic nouns, because more than 70% of monomorphemic words in the testbed are nouns. These final guessing rules are linked to orthographic cues as discussed above.

YumiG1-YumiG5	YumiG6
Affixation with optional clitics (35)	Monomorphemic words with clitics (7) Monomorphemic nouns with no clitics (4)

Table 7.18. Number of rules in the resources in the Guesser (46 rules)

Text	Analysis
Eirlyn	`Eirlyn,ROOT+NOU+YumiG6`

Diinta	`di,PFX+DER:VER+PSV+YumiG4`
	`inta,ROOT+NOU+YumiG4`

mengaweetkan	`meng,meN,PFX+DER:VER+ACV+YumiG1`
	`aweet,ROOT+NOU+YumiG1`
	`kan,SFX+DER:VER+YumiG1`

keju	`keju,ROOT+NOU+PS+AK+TX+DykaA1`

Table 7.19. Some words annotated using the Guesser

7.4.3 Module 3: the Improver

The Improver consists of two resources: DitoR1 and DitoR2 (Dito = arbitrary code for syntactic grammar resources, R = 'improveR', 1 and 2 = 1st and 2nd in order).

NooJ priority code	Resource	Disambiguation rule type	Units targeted
1	DitoR1	Contextual	Full root reduplication
2	DitoR2	Contextual	Full word reduplication *Special case* reduplication

Table 7.20. Improver resources

DitoR1 has the exclusive purpose of supplying the correct analysis for *full root reduplication morphemes*, which the Annotator inaccurately analyses as lexical roots rather than realisations of the abstract reduplication morpheme. So, for instance, for the word *pukul-pukul* 'hit iteratively', both elements are given the analysis `ROOT+VER`, but this is only correct for the first (source); for the second (copy), `RED:FULL+VER+ITRV` is the desired analysis.

To adhere to the MAS (see 4.2.4), the code `ROOT` on the copy token must be converted into `RED:FULL` (full reduplication) followed by the POS of the reduplicated root, plus in some cases functional tags. Since, in NooJ, a syntactic rule cannot replace one analysis with another in a single pass, this correction involves two steps: adding the correct analysis, and subsequently removing the incorrect analysis/analyses. In SANTI-morf, the Improver performs the former step; the latter is left to the Disambiguator (see 7.4.4).

Text	Analysis
pukul	`pukul,ROOT+VER`
pukul	`(pukul,ROOT+VER` **`| pukul,RED:FULL+DER:VER+ITRV+DitoR1`**`)`

Table 7.21. Annotation of *pukul-pukul* by the Improver (added analysis in bold)

Addition of the `RED:FULL` analysis needs to be sensitive to context; it must only apply when an instance of a root form is used as a reduplication morpheme and nowhere else (i.e. the second *pukul* in *pukul-pukul* but not the first), since other instances of the same forms are probably correctly tagged already.

A syntactic rule that analyses reduplication must incorporate *variables* (e.g. A and B) and an *equality constraint* (`$A_=$B_`), as described in 7.2.2.2, to check whether one token of a pair is a copy of the other (the source). The analysis must also capture affixes preceding, following, or encompassing the root (so, not only *pukul-pukul* 'hit iteratively' but also *di-pukul-pukul* 'be hit iteratively' and *di-pukul-pukul-i* 'be hit iteratively (applicative)').

I address this by incorporating epsilons in union with affixations into the rule (given in full below). For example, in `(<PFX>|<E>) $(A <ROOT+VER> $) (<SFX>|<E>)` epsilons are used in union with a prefix and suffix, enclosing variable A, which stores the source verb root. A similar alternation is used for variable B, which stores the copy root – the location to which the reduplication analysis is to be added. This makes the affixes optional.

```
VER_RED =
(<PFX>|<CFX>|<E>)
$(A <ROOT+VER> $)
(<SFX>|<CFX>|<E>)
(<E>|-)
(<PFX>|<CFX>|<E>)
<E>/<$A_,RED
$(B <ROOT+VER> $)
<E>/:FULL+DER:VER+ITRV+DitoR1>
<E>/<$A_=$B_>
(<SFX>|<CFX>|<E>)
;
```

Some reduplication morphemes may be targeted by multiple rules. For instance, in *pukul-memukul*, the reduplication morpheme would be recognised by rule `RED_VER` supplying <`RED:FULL+DER:VER+ITRV`> but also by a more specific rule (`VER_RED_ITRV_RECP`) which supplies <`RED:FULL+DER:VER+ITRV+RECP`>, the difference being the added `RECP` for reciprocal. To prevent both rules adding an analysis, the `+UNAMB` priority code is added to the more specific rule:

```
VER_RED_ITRV_RECP=
$(A <ROOT+VER> $)
(<E>|-)
<PFX+meN>
<E>/<$B_,RED
$(B <ROOT+VER> $)
<E>/:FULL+DER:VER+ITRV+RECP+DitoR1>
<E>/<$A_=$B>
+UNAMB
;
```

Text	analysis
pukul	`pukul,ROOT+VER+PS+AP+T1+DykaA1`
memukul	`mem,PFX+meN+DER:VER+ACV+RL=112302+YumiA1` **(**`pukul,ROOT+VER+PS+AP+T1+DykaA1` **|pukul,RED:FULL+DER:VER+ITRV+RECP+DitoR1)**

Table 7.22. Annotation of *pukul-memukul* by the Improver (added analysis in bold).

Full-word reduplication rules are stored in *DitoR2*. Unlike the rules in *DitoR1*, which only target roots, these target all morphemes in a word. Thus, the number of constraints is twice the number of morphemes copied. An example from *compound reduplication* (see 2.1.3.4.4 and

4.2.4.1) serves to illustrate this. In *es krim-es krim* ‘ice creams’, the noun-noun compound is formed from two morphemes, *es* ‘ice’ and *cream* ‘krim’, and is then fully reduplicated. Therefore, the rule that analyses this word uses four variables. Variables A, B, C and D store *es, krim, es,* and *krim,* respectively. There are also *two* equality constraints (for A and C, and for B and D), shown in bold in the rule:

```
NOU_ADJ_RED =
$(A <ROOT+NOU> $)
$(B <ROOT+ADJ> $)
(<E>|-)
<E>/<$A_,RED
$(C <ROOT+NOU> $)
<E>/:FULL+DER:NOU+DitoR1>
<E>/<$A_=$C_>
<E>/<$B_,RED
$(D <ROOT+NOU> $)
<E>/:FULL+DER:ADJ+DitoR1>
<E>/<$B_=$D_>
;
```

DitoR2 also includes rules to annotate what I term *special-case* reduplication, where a single circumfix surrounds a fully reduplicated root. This case is not handled by DitoR1. For instance, adverbialiser circumfix *se—nya* applied to reduplicated adjective *cepat* ‘quick’ in *cepat-cepat* ‘quickly’ results in *secepat-cepatnya* ‘as quickly as possible’. DitoR1 analyses the root reduplication correctly, but the circumfix remains incorrectly analysed as a combination of prefix and suffix. DitoR2 contains rules to apply a correct analysis to these cases (more detail on the method is given in Prihantoro 2021); as previously, the incorrect analysis is left to be removed at a subsequent stage.

Overall DitoR1 has 7 reduplication rules. The rule that analyses reduplication with reciprocal or iterative function is prioritised. DitoR2 has 19 full-word reduplication rules and 9 special-case reduplication rules.

7.4.4 Module 4: the Disambiguator

The *Disambiguator* consists of three syntactic rule resources beginning with DitoD1 (Dito = arbitrary code, D = disambiguator, 1 =1st in order) (see Table 7.23). The priority level of these resources starts at 3, so that they take effect after the Improver resource DitoR1.

NooJ priority code	Resource name	Disambiguation rule type	Units disambiguated
3	DitoD1	Non-contextual	Reduplication
4	DitoD2	Contextual	All units
5	DitoD3	Non-contextual	

Table 7.23. Disambiguator resources

7.4.4.1 DitoD1: disambiguating reduplication

The only rule in DitoD1 is `Main = <RED>/<RED>`, a non-contextual disambiguation rule that targets all reduplications regardless of context. Whenever a token annotated by `<RED>` is found, this rule selects `<RED>` as the correct annotation and removes any other annotation(s). It thus removes non-reduplication annotation from reduplication morphemes, preserving only the annotations added by the Improver (see 7.4.3), as laid out in Table 7.24.

Word	Before the Disambiguator module	After the Disambiguator module
pukul	`pukul,ROOT+VER+PS+AP+T1+DykaA1`	`pukul,ROOT+VER+PS+AP+T1+DykaA1`

pukul	`pukul,ROOT+VER+PS+AP+T1+DykaA1`
	`pukul,RED:FULL+DER:VER+ITRV+DitoR1`
		~~pukul,ROOT+VER+PS+AP+T1+DykaA1~~
		`pukul,RED:FULL+DER:VER+ITRV+DitoR1`

Table 7.24. Effect of DitoD1 on the annotation of reduplication *pukul-pukul* (change in bold)

7.4.4.2 DitoD2 and DitoD3

Resources DitoD2 and DitoD3 contain disambiguation rules to resolve the remaining ambiguity. DitoD2 contains rules which test the context to select a particular analysis. DitoD3 contains unconditional rules which apply to the same targets in all other contexts than those tested in DitoD2. The disambiguation of *bisa* (noun root *bisa* 'poison', modal adverb root *bisa* 'be able to') illustrates this. *Bisa* is a noun root when preceded by prefix *ber-* or followed by *ular* 'snake' or a noun for a type of snake (*bisa ular* 'snake venom'; *bisa kobra* 'cobra venom'); but it is likely to be an adverb root in other contexts, e.g. when followed by a passive or active prefix or a verb root, when preceded or followed by a personal pronoun, and in various other settings. It is only necessary to define a rule to select the analysis which is correct in the smaller number of distinct contexts. For *bisa*, the three noun contexts are written in union in rule `nou_bisa`:

```
nou_bisa =
<bisa>/<ROOT+NOU> <ular>
| <bisa>/<ROOT+NOU> :snake
| <ber> <bisa>/<ROOT+NOU>
;
```

With an analysis selected as correct, NooJ automatically removes the adverb analysis in those cases. At this point, characterising the contexts where the other analysis is correct is not necessary; one non-contextual disambiguation rule will suffice if it is lower priority (i.e. in DitoD3, as opposed to DitoD2). This is more efficient than defining all possible contexts for the more widespread analysis.

Thus, for *bisa*, all other cases are dealt with by `adv_bisa = <bisa>/<ROOT+ADV>`, a simple rule in DitoD3 which always selects the adverb root analysis if present, and removes other annotations. DitoD2 contains 56 contextual disambiguation rules targeting individual ambiguous morphemes like *bisa.* DitoD3 contains 38 non-contextual disambiguation rules (less than 56 since some morphemes have more than one rule in DitoD2).

Ambiguity may persist even after the application of DitoD2 and DitoD3, and thus in the output. For example, Table 7.25 shows analyses arising from three different tokenisations produced by the Annotator for *mengemas* 'pack', ambiguity not resolved by the Disambiguator.

Three possible tokenisations: *meng-(k)emas*, *meng-emas*, *menge-mas*	
meng,meN,PFX+DER:VER+ACV+RL=112304+YumiA1	**kemas,ROOT+VER+PS+AK+TX+DykaA1**
meng,meN,PFX+DER:VER+ACV+RL=112304+YumiA1	emas,ROOT+NOU+PS+AE+TX+DykaA1
menge,meN,PFX+DER:VER+ACV+RL=112306+YumiA1	mas,ROOT+NOU+MS+AM+TX+DykaA1

Table 7.25. Unresolved ambiguity for *mengemas*

This is a morphophonological ambiguity: any of three different roots *could* be the one combined with the active prefix. Phonological context cannot resolve this kind of ambiguity. It would be possible to 'guess' by selecting the most frequent of the three in all cases, as per practice in the Guesser module, reducing ambiguity at some cost to correctness. However, unlike in the Guesser, the three analyses are *known* (not just *likely*) to include the correct analysis. For reasons of caution, to minimise rejection of correct analyses, no further action is taken on these ambiguities.

7.5 Evaluation

7.5.1 The testbed

In chapter 5, I undertook a number of evaluations using a corpus annotated by MorphInd. I here argue that this testbed may also be appropriately used to evaluate SANTI-morf. The size of the testbed is approximately 10K words, but (as argued in 2.2) this is sufficient for evaluation.

One benefit of reusing this testbed is that it allows comparison between MorphInd and SANTI-morf. While the systems' MASs differ, a fair comparison can still be made regarding coverage. MorphInd annotates a considerable proportion of words as 'unknown' (see 5.5.8). Evaluating SANTI-morf's coverage will reveal whether this aspect of its performance is better or worse than MorphInd's.

7.5.2 Procedures for evaluation

After applying SANTI-morf to the testbed, I used NooJ's *export* function to transfer the annotation output from the native NooJ format to a plain-text file. In NooJ, this export function seems to be optimised for morphosyntactic and syntactic annotation, rather than morpheme-level annotation. In consequence, exporting SANTI-morf output causes data loss, as some morphemes/annotations are missing. This problem has been reported to NooJ's author, who reports that fixing it requires heavy modification of NooJ's core engine (Silberztein, personal communication).

To get around this problem, I created the following procedure. First, I added *exportable* codes to all lexicons and rules in all modules. These codes are reformatted annotations capable of passing through the NooJ export tool, albeit in an unconventional format. This unconventional format is then mapped back to the normal codes, in the format I need for evaluation (see Table 7.26), using a small PHP script. I then transfer the reformatted data to a spreadsheet for evaluation coding.

In this spreadsheet, the first column contains the word tokens, one per row in vertical order, as in Table 7.26. The analyses of the morpheme(s) that constitute each word, concatenated in order of occurrence, are given in the second and subsequent columns. Each morphological analysis follows the MAS described in Chapter 4. If a morpheme is ambiguously annotated, the alternative analyses are enclosed in brackets, and separated by pipes, as exemplified for *banding* 'ratio' in Table 7.26.

Word	Morpheme 1 analysis	Morpheme 2 analysis
mencapai	`men,meN,PFX+DER:VER+ACV`	`capai,ROOT+VER`
1	`1,DGT`	
banding	`banding,ROOT+NOU|banding,ROOT+VER`	
4	`4,DGT`	

Table 7.26. Sample output in spreadsheet format for *mencapai 1 banding 4* 'reach 1:4 ratio'

Six columns for evaluation codes are added before the word column (see Table 7.27). The evaluation categories, based on those introduced in section 5.5.4 but, as will be explained, adjusted slightly for the present task, are coverage (CV), correct tokenisation (CT), incorrect tokenisations (IT), number of morphemes (NM), correct analyses (CA), and incorrect analyses (IA). The evaluation consisted of manually entering a code or number for each category for each word token. After fully analysing the annotation of the testbed in this way, I used the results to calculate the following evaluation measures: *coverage, precision, recall, F-measure,* and *ambiguity rate*.

CV	CT	IT	NM	CA	IA	Word	Morpheme 1	Morpheme 2
1	1	0	2	2	0	mencapai	`men,meN,PFX+DER:VER+ACV`	`capai,ROOT+VER`
1	1	0	1	1	0	1	`1,DGT`	
1	1	0	1	1	1	banding	`banding,ROOT+NOU` `|banding,ROOT+VER`	
1	1	0	1	1	0	4	`4,DGT`	

Table 7.27. Sample evaluation codes with the corresponding output

Coverage (C) is an evaluation measure I introduced in section 5.5.8. If a word has been analysed (regardless of correctness and ambiguity), I assign code 1 to CV. If the word is left unanalysed, I assign 0. Coverage is calculated by dividing the sum of CV by the number of word tokens in the corpus, including punctuation marks and numbers (WT):

$$C = \frac{\sum \mathrm{CV}}{\mathrm{WT}}$$

SANTI-morf's MAS does not require full disambiguation of the final output. Thus, rather than *accuracy*, a measure commonly used to evaluate unambiguously annotated data (and that I used in Chapter 5 to evaluate MorphInd), I calculate *precision* and *recall* to evaluate SANTI-morf's performance. I also calculate the *F-Measure*, which is the harmonic mean of precision and recall. The F-measure is required as there is a natural trade-off between precision and recall. The ambiguity rate is also calculated.

I measure precision and recall for both tokenisation of morphemes and morphological analysis. The basic datapoints for tokenisation precision and recall are CT and IT. If a word has no analysis, both are set to 0. If a word is correctly tokenised in *any* of its analyses, 1 is recorded under CT (otherwise 0). The number of incorrectly-tokenised analyses is recorded under IT. For example, if a word is given three analyses, one correctly tokenised and two incorrectly tokenised, CT is 1 and IT is 2. Tokenisation precision (TP) is then the sum of all CT values divided by the sum of all CT *and* IT values. Tokenisation recall (TR) is the sum of all CT values divided by the number of word tokens in the corpus (WT). The tokenisation Ambiguity rate (TA) is the sum of all tokenisations (CT plus IT) minus WT, divided by WT (i.e. the number of surplus tokenisations per word token).

$$TP = \frac{\sum \mathrm{CT}}{(\sum \mathrm{CT} + \sum \mathrm{IT})}$$

$$TR = \frac{\sum \mathrm{CT}}{\mathrm{WT}}$$

$$TA = \frac{(\sum \mathrm{CT} + \sum \mathrm{IT}) - WT}{\mathrm{WT}}$$

The scores for the correctness of annotation are calculated as follows. NM records the number of morphemes in each word. CA records the number of correct morpheme-level analyses and IA the number of incorrect morpheme-level analyses. Due to ambiguity, CA and IA may sum to more than NM.

Morphological analysis precision (MAP) is then the sum of all CA values, divided by the sum of all CA *and* IA values. Morphological analysis recall (MAR) is the sum of all CA values, divided by the sum of all NM values. Morphological analysis F-measure (MAF) is the harmonic mean of MAP and MAR. Morphological analysis ambiguity rate (MAA) is the total of all morphological analyses (sum of CA and IA), minus the sum of NM, divided by the sum of NM.

$$MAP = \frac{\sum \mathrm{CA}}{(\sum \mathrm{CA} + \sum \mathrm{IA})}$$

$$MAR = \frac{\sum \mathrm{CA}}{\sum \mathrm{NM}}$$

$$MAA = \frac{(\sum \mathrm{CA} + \sum \mathrm{IA}) - \sum \mathrm{NM}}{\sum \mathrm{NM}}$$

7.5.3 Coverage

SANTI-morf achieves 100% coverage (C=1). Every word token is given an analysis. By contrast, MorphInd achieved 93% coverage. Words that MorphInd failed to analyse such as the place name *Yogyakarta* 'Yogyakarta' and *pidana* 'crime' are analysed by SANTI-morf. Although the Guesser guarantees an analysis, of the overall tokens, only slightly more than 6% are annotated by the Guesser; see Table 7.28. This means the coverage provided by the Annotator alone is more than 94% , which is quite high and still above MorphInd's 93%.

Analysed by...	Word tokens	Percentage
Annotator	10,115	93.8%
Guesser	626	6.2%
Total	10,228	100%

Table 7.28. Proportion of words analysed by the Annotator and the Guesser

The breakdown of words dealt with by the Guesser (see Table 7.29) shows that the majority are proper nouns such as *Sayyid, SMKN* (abbreviation for *Sekolah Menengah Kejuruan Negri* 'state vocational school'), or *Doraemon*. Polymorphemic unknown words such as *ber-cengkerama* 'have fun' or *di-elektrolisis* 'be electrolysed' are in the minority (113 out of 626).

Guesser resource	Word tokens	Example(s)
YumiG1	2	*se-baagai-nya* 'etc.'
YumiG2	6	*pem-bahagi-an* 'division', *ke-radja-an* 'kingdom'
YumiG3	2	*pel-adjar-an* 'lesson'
YumiG4	86	*di-elektrolisis* 'be electrolysed', *ber-cengkerama* 'have fun'
YumiG5	7	*me-lajoe* 'run', *pegih-nja* 'the leaving'
YumiG6	523	*Sayyid*, *SMKN* (proper nouns)
Total	626	

Table 7.29. The distribution of words guessed by different resources in the Guesser

7.5.4 Tokenisation

SANTI-morf seems to perform tokenisation well, as indicated by the tokenisation precision, recall, and F-measure scores; all are 99% or above (Table 7.30). The most polymorphemic word in this corpus, *ke-tidak-ber-daya-guna-an* 'the unpowerless-and-unusefulness' is among those successfully tokenised.

$$TP = \frac{10228}{(10160 + 126)} = 0.99$$

$$TR = \frac{10160}{10228} = 0.9976$$

$$TF = \frac{2 \text{ x } 0.9900 \text{ x } 0.9976}{(0.9900 + \ 0.9976)} = 0.9938$$

$$TA = \frac{(10160 + 126) - 10228}{10228} = 0.01$$

Evaluation	Percentage
Tokenisation Precision (TP)	99.0%
Tokenisation Recall (TR)	99.7%
Tokenisation F-Measure (TF)	99.3%
Tokenisation Ambiguity rate (TA)	1.0%

Table 7.30. Evaluation of SANTI-morf tokenisation

Some incorrect tokenisations are still present, and can be divided into two types. In the first, a single incorrect tokenisation is given. This occurs for a number of words that are analysed by the wrong resources. For example, *Binus* (university name) should have been analysed as a monomorphemic word by DykaA2 (proper noun lexicon) or DitoG6 (monomorphemic noun guesser). Instead, it is analysed as a combination of two foreign roots (from English) *bin* and *us* by YumiA3 (compound word rules). This happens because *Binus* is absent from the DykaA2 lexicon, and YumiA3's priority is higher than DitoG6's. Similarly, the abbreviation *Menkopolhukam* 'coordinating minister of law, security, and defence' is incorrectly tokenised as a combination of *men-* and a non-existent root *kopolhukam* by YumiG4 (affixation rules). It could have been correctly analysed as a monomorphemic noun using YumiG6, but was not because YumiG4's priority is higher. The principle of the "lexicon as a repository of exceptions" would suggest this word needs to be in the lexicon. As explained in 7.4.1.2, these exceptions are given higher priority than the rules that they go against.

In the second type, alternative tokenisations occur alongside the correct tokenisation produced by the same rule. This occurs because the constraint(s) in the rule is not rigorous enough to prevent the alternative tokenisations being found. For instance, in *mengemas* 'pack', the correct tokenisation (*meng-(k)emas*) and the alternative tokenisations (*meng-emas; menge-mas*) are all given by affixation rules in YumiA1.

However contextually incorrect, these alternatives are morphophonologically valid because *meng-* occurs with vowel-initial roots like *emas* 'gold', and *menge-* occurs with monosyllabic roots like *mas* 'brother'*.* The word *mengemas* is thus genuinely ambiguous. This kind of problem is a major contribution to the ambiguity rate of 1%.

7.5.5 Morphological analysis

This evaluation considers the correctness of each morphological analysis for each morpheme in every word. SANTI-morf seems to perform well, scoring above 99% in morphological analysis precision and recall (Table 7.31). Correspondingly the morphological analysis ambiguity rate is less than 0.5%.

$$MAP = \frac{13529}{(13406 + 285)} = 0.9954$$

$$MAR = \frac{13406}{13529} = 0.9909$$

$$MAA = \frac{(13406 + 185) - 13529}{13529} = 0.0045$$

	Evaluation	Percentage
1	Morphological Analysis Precision (MAP)	99.54%
2	Morphological Analysis Recall (MAR)	99.09%
3	Morphological Analysis Ambiguity rate (MAA)	0.45%

Table 7.31. Evaluation of SANTI-morf morphological analysis

There are two types of incorrect morphological analyses. In some, no correct analysis is given because of incorrect tokenisation. For example, *Penangsang,* a personal name which should have been analysed as a monomorphemic noun, is inaccurately tokenised into *pen-angsang* (*angsang* being a non-existent root). In consequence, the analyses for both morphemes are counted as incorrect, without any correct answer (since the true analysis, monomorphemic noun root, is not possible given then incorrect tokenisation).

In this case, the reason for the error is that this name is not in the proper noun lexicon (DykaA2). The Guesser could have analysed this word correctly but was never applied, a match being made by YumiA1, the affixation rule resource, which has higher priority than the Guesser. This issue consistently affects proper names with parts that resemble affixes (such as *Pen* in *Penangsang* resembling prefix *pen-*).

In some other cases, the tokenisation is correct, but the analyses are incorrect. For instance, verb root *kerah* in *meng-(k)erah-kan* is wrongly analysed as a noun (even though active prefix *meng-* and verbal suffix *-kan* are correctly analysed). This is because the DykaA1 lexicon incorrectly lists *kerah* as a noun root only. Adding the verb analysis to the lexicon entry for *kerah* is the only possible solution.

7.6 Summary and conclusion

In this chapter, I have described SANTI-morf's architecture, resources, and system configuration. I have also completed an evaluation of SANTI-morf's performance using the same testbed I used to evaluate MorphInd in Chapter 5. The evaluation shows that SANTI-morf performs well. It outperforms MorphInd in terms of coverage; SANTI-morf has perfect coverage, whereas MorphInd only covers 93% of the testbed (see Table 7.32).

Since SANTI-morf allows ambiguous analyses to be retained, its tokenisation is not directly comparable to MorphInd's. Nevertheless, SANTI-morf scores above 99% in precision and recall. The ambiguity of the tokenisation is quite low, only 1% at maximum.

The evaluations undertaken in Chapter 5 of MorphInd's word-level tag, root tag, and overall token accuracy are not applicable to SANTI-Morf. This is due to the characteristics of the MAS that each system applies. MorphInd leaves affixes untagged and supplies word-level tags. Conversely, SANTI-morf applies morphological analyses to all morphemes, including affixes, but does not provide a single word-level tag.

For a rule-based system, the performance of SANTI-morf's morphological analysis is quite high. It scores above 99% in precision and recall. The ambiguity level is less than 0.5% . SANTI-morf cannot be evaluated on aggregate accuracy, as MorphInd was (see 5.5.7), because it allows ambiguous analyses to be retained in the annotation output. However, SANTI-morf's more fine-grained tagset, and its performance as measured by precision and recall, demonstrate it to be an advancement over the previous state-of-the-art system.

Evaluation	MorphInd	SANTI-morf
System coverage	93%	100%
Overall token accuracy	89%	NA
Tokenisation Precision (TP)	NA	99.0%
Tokenisation Recall (TR)		99.7%
Tokenisation Ambiguity rate (TA)		1%
Morphological Analysis Precision (MP)		99.5%
Morphological Analysis Recall (MR)		99%
Morphological Analysis Ambiguity rate (MA)		0.4%

Table 7.32. Summary of SANTI-morf evaluation scores compared to MorphInd

SANTI-morf's high performance arises from its wide-coverage rules and lexicon and its system of resource prioritisation. The multi-module pipeline allows SANTI-morf to reduce ambiguity even before the Disambiguator applies. On the one hand, this makes the design of rules slightly more complicated, but on the other hand, it contributes to the ultimate low ambiguity rate and high precision.

Compared to two successful and well-known rule-based taggers for other languages, the performance of SANTI-morf is not bad (see Table 7.33). SANTI-morf performs slightly better than Oflazer and Kuruoz's (1994) tagger, as reported in Voutilainen (1999:18). Its recall is lower than EngCG (Voutilanen 1992), but its precision is higher.

	Evaluation	SANTI-morf (Indonesian)	EngCG (English)	Oflazer and Kuruoz (1994) (Turkish)
1	Precision	99%	95%	97%-98%
2	Recall	99%	99.8%	98%-99%
3	Ambiguity rate	1%	4.5%	1-2%

Table 7.33. Cross-language comparison of SANTI-morf with other rule-based taggers

Overall, that SANTI-morf performs at more than 99% precision and recall reconfirms Voutilanen's (1999:18-19) argument that rule-based systems do not always perform worse than data-driven/statistical or hybrid systems. More importantly, these measures demonstrate that the objective of this chapter, which was to implement SANTI-morf, has successfully been accomplished.

CHAPTER 8

CONCLUSION

8.1 Summary of thesis and aims achieved

In this final chapter, I will first present a summary of this thesis (in section 8.1), showing a) how I successively completed each required step for the subsidiary aims of this thesis; and b) what outcome was produced at each step. Next, in 8.2, I will highlight some limitations of this project and how they were dealt with. I then move on to discuss potential directions for further research (in 8.3). In 8.4, I conclude by highlighting how this thesis contributes to the field of Indonesian corpus linguistics.

The primary objective of this thesis, the creation of SANTI-morf, has been accomplished in three stages as stated in the aims of this thesis (see 1.3): *preliminaries*, *scheme creation*, and *implementation*. The preliminary aims were: to explain the scope of my study; to introduce Indonesian; and to prepare a testbed. In Chapter 2, I introduced the structures of standard Indonesian, particularly morphology. I then reported the purpose, design, and creation of a testbed.

In Chapter 3, I reviewed Morphological Annotation Schemes (MASs) for a number of different languages, including that used by MorphInd (Larasati et al. 2011), the state-of-the-art morphological annotation system for Indonesian. The outcome of this survey, that is, the review presented in Chapter 3, was a prerequisite to my development of a novel MAS for Indonesian (*scheme creation*). From this survey, I deduced the accepted best practices for MASs, which I used as a guide to devise the new MAS in Chapter 4. I translated the best practices into 15 guiding principles for a MAS which differs substantially from, and is more fine-grained than, MorphInd's MAS. The successful creation of my MAS fulfilled the second aim of the thesis.

System implementation involved: a) an evaluation of the present state-of-the-art system (Chapter 5); b) selection and justification of an approach and software framework, accomplished via a review of work on morphological annotation (Chapter 6); c) creation of SANTI-morf's computational language resources (sections 7.2 to 7.4); d) evaluation of SANTI-morf (section 7.5); and e) a final comparison with the prior state-of-the-art system (section 7.6).

Evaluation of MorphInd as the state-of-the-art morphological analysis system for Indonesian was undertaken in Chapter 5 and showed that MorphInd covered only 93% of word tokens in the testbed with an accuracy of only 89%. This evaluation set a benchmark for subsequent assessment of SANTI-morf.

In Chapter 6, the theoretical foundations of formal grammar and generative morphology, as well as the historical development of Koskenniemi's (1983) Two-Level Morphology, were reviewed as background to my choice of a rule-based approach for SANTI-morf and to my selection of NooJ (Silberztein 2003) as implementation software. The rule-based approach was chosen because it does not rely on a large training corpus, as usually required by data-driven or statistical tagging techniques. Rather it relies on hand-crafted rules and lexicons. Prior work in the field has shown that adopting the rule-based approach for the aforementioned reasons does not result in worse performance in comparison to a data-driven or hybrid approach. The rule-based approach requires a finite-state program for its implementation; among the candidate systems, I determined that NooJ is a very good fit for the morphology of Indonesian as it has built in functions to tokenise, annotate, and disambiguate (at morpheme level) all morphological processes in Indonesian.

In Chapter 7, I reported the process of creating the various modules of SANTI-morf and the linguistic resources of which they consist, as well as presenting an evaluation of its performance. The outcomes of this chapter were the aforementioned SANTI-morf language resources (86,590 lexicon entries and 659 rules across 3 lexicon files and 15 rule files), its system configuration (as a multi-module pipeline including an Annotator, Guesser, Improver, and Disambiguator), and a novel technique to export complex annotation data from NooJ to plain-text format, necessitated by the SANTI-morf MAS's use of normally non-exportable constructs. The evaluation of SANTI-morf, performed against the same testbed used earlier to evaluate MorphInd, showed that SANTI-morf has 100% coverage due to the methods of the Guesser module, and higher than 99% precision and recall. These findings demonstrate that SANTI-morf is an advancement relative to MorphInd. With that, the main objective of this study is fulfilled.

Also in this chapter, I introduced the evaluation methods used to measure SANTI-morf's performance. *Coverage*, that is, the proportion of word tokens assigned some analysis, was earlier used in the evaluation of MorphInd, whose coverage was only 89%. To measure the quality of the analysis, rather than a single accuracy score, I used precision and recall. These measures are appropriate because SANTI-morf allows some ambiguities to be retained in the final output (thus, there may be more than one analysis per token).

8.2 Limitations

The problems with SANTI-morf that I identified during the evaluation exercise (Chapter 7) show that there is still much room for improvement. At the moment, SANTI-morf's text-file output is an XML document, which cannot be indexed directly by most corpus analysis software. Further reformatting is required. However, in the XML document, each element of the morphological analysis is clearly and unambiguously presented with explicit delineators. This consistency allows easy conversion to formats acceptable to different programs.

In the current MAS, certain actual features of running text (e.g. roman numerals, variant spellings of proper nouns, date and time expressions) are not fully covered. This did not cause a significant drop in SANTI-morf's performance in the evaluation, because these elements are infrequent in the testbed. Nevertheless, the required fixes remain an urgent need, as from any given user's perspective, infrequent features might be of central interest, or might be frequent in the texts they wish to work with; consider the word *organ* 'organ', which will be fairly infrequent in most general corpora, but is likely to be of high frequency in a corpus of biomedical documents.

The SANTI-morf MAS, though fine-grained overall, ignores certain distinctions. For example, *peN-* and *pe-* are both currently annotated as nominaliser prefixes. This analysis would be more fine-grained if they could be differentiated by semantic function as *agentive*, *patientive* or *instrumental*. Implementing this distinction would require lexical constraints to be added to the rules. The simple category *NYA* does not distinguish pronominal clitics from definite suffixes (see 4.2.2.2), but a more sophisticated disambiguation module might have made this distinction workable. However, the lack of distinctions like these is an acceptable limitation to the present system because such distinctions can be considered to be semantic or syntactic, rather than morphological, features. This follows my principle 15 for MAS design (see 4.1.15), which allows the annotation to dismiss categories whose disambiguation requires linguistic information from beyond the morphological level (morphosyntactic, syntactic, semantic, or any combination thereof). Fine-grained semantic and syntactic analyses can therefore be integrated at a subsequent stage of research; I project future work to develop SANTI-POS (POS tagger), SANTI-sense (semantic tagger), and/or SANTI-parse (syntactic annotator).

The relatively small size of the testbed is a further limitation; a larger testbed would arguably have guaranteed a more reliable evaluation. However, as mentioned in section 2.2, Voutilanen (1999:19) shows that enlarging the EngCG testbed, which was three times smaller than my testbed, led to essentially similar performance scores. Thus, there is reason to suspect that testbed size is not a serious limitation on the accuracy of my evaluation. Nevertheless, an expanded testbed may well prove useful for future work (see below).

A final limitation of this thesis project is that certain SANTI-morf-related activities that I had scheduled to take place in Indonesia during 2020 could not be carried out as a result of the global COVID-19 pandemic. These activities included preliminary work on: seeking input from Indonesian grammarians on the shape and future development of the MAS; inter-rater agreement evaluation of the MAS; sample analyses using SANTI-morf-annotated data as illustrations of its utility; and an initial acceptance or uptake analysis. All these required in-person work with Indonesian linguists, and had been planned for my trip to the 2020 KIMLI (*Konferensi Ilmiah Masyarakat Linguistik Indonesia*) 'Scientific Conference of the Linguistic Society of Indonesia', the largest such event. However, the conference was cancelled due to the pandemic, with no chance of being rescheduled within the thesis project's timeline; thus these activities had to be omitted from the present thesis.

That said, even without the pandemic, there would have been little space for these elements in my thesis, given the extensive discussion of the detail of MAS and its *actual implementation,* which needed major attention. In mitigation of the impact of the absence of these elements, I would argue that the extensive literature reviews of both MAS creation (Chapter 3) and system implementation (Chapter 4) have functioned as proxy means by which I have benefitted from the knowledge of other scholars. I also carried out multiple experiments in the process of creating the SANTI-morf resources and architecture to determine an optimum approach (see Chapter 7, which for reasons of space could not detail *every* such experiment or *every* possible approach that turned out to be suboptimal). The results of the evaluation show that SANTI-morf is in any case an improvement upon its predecessors, in spite of these limitations.

8.3 Directions for future work

In addition to SANTI-morf, which performs annotation at morpheme level, I plan to develop companion systems for other linguistic levels, as previously mentioned: SANTI-POS for POS annotation at word level; SANTI-sense for semantic sense annotation at word level; and SANTI-parse for syntactic analysis/parsing at phrase/clause level. My hope is that, together with SANTI-morf, these will make up *SANTI-ling*, a multi-level linguistic annotation system for Indonesian.

Once these systems are available, extending the SANTI-morf MAS to incorporate morphological features that intersect with (morpho-)syntactic or semantic features will be feasible, as SANTI-morf will be able to reuse analyses from other sub-systems for more sophisticated annotation and disambiguation. Thus, when a SANTI-ling-annotated corpus is indexed in a suitable analysis program, users will be able to build queries that combine morphological, POS, semantic, and syntactic criteria.

In other work, I aim to make SANTI-morf (and later SANTI-ling) able to run from a command prompt or terminal, rather than just the NooJ graphical interface. This will allow other researchers to use them as support systems for NLP tools such as sentiment analysis, question-answering, or document summarisation, as well as for corpus annotation for linguistic research. For less technical end-users, I hope to build a web-based interface for SANTI-morf (and SANTI-ling) so that Indonesian linguists can annotate their data without any of the installation and setup steps required to use SANTI-morf in NooJ.

Another direction for future work concerns annotation speed. I plan to experiment on modifying NooJ's *inflectional-derivational* grammar affordance, designed to implement morphosyntactic annotation, to instead implement the morpheme-level annotation which SANTI-morf currently accomplishes via NooJ lexicons and rules (following Kartunnen et al.'s 1992 lexical transducer approach; see 6.6.4). This would allow speedier text processing. The current SANTI-morf annotates 60 word-tokens per second. Compared to EngCG, which 20 years ago could annotate 3,000 words per second (Voutilanen 1999:18), this speed is very poor (50 times slower), and thus, needs improvement.

Finally, I plan to build a formatter program to convert SANTI-morf output to a format acceptable to online corpus indexer platforms, such as CQPweb (Hardie 2012). Correctly formatting the output requires me to combine NooJ with a conversion program. My recommendation to NooJ's author (Silberztein, personal communication) is that it ought to be possible for such formatter programs to be incorporated as NooJ plugins, so that data exported from NooJ will be ready for CQPweb without any intermediate program at all.

8.4 Contributions of this thesis

This study's primary contributions to the fields of Indonesian linguistics and computational morphology are the SANTI-morf MAS and the SANTI-morf system. The SANTI-morf MAS is the first Indonesian MAS that permits full morphological annotation at morpheme level (see Chapter 4); MorphInd's MAS, by contrast, leaves affixes unannotated.

SANTI-morf outperforms MorphInd in terms of coverage. Its MAS is also more fine-grained than MorphInd's. Using a SANTI-morf annotated corpus, users can search for morphemes with any relevant grammatical property using an actual morphological tagset. This is made easier by the fact that SANTI-morf lacks the 'unknown' category expressed as tag X for words left unanalysed. SANTI-morf can also search using the actual morph form. Users can search based on both orthographic and citation forms, and on both formal and functional categories (whereas MorphInd annotation only allows queries based on citation forms and functional categories). Affixes, which in MorphInd are left unannotated, are fully annotated in

SANTI-morf. The SANTI-morf MAS also incorporates functional analytic categories absent from the MorphInd MAS, including outcome POS and reciprocal, applicative and causative voice. The fuller representation of the morphology of Indonesian makes possible more robust searching.

To my knowledge, SANTI-morf is the first system that uses NooJ to implement morpheme-level annotation for Indonesian. So far, I am the only contributor for Indonesian language resources to the NooJ language resource repository[67]. In this way, I have contributed to the field by bringing use of NooJ use to a language for which it did not previously exist.

Another accomplishment of this thesis is the testbed (see Chapter 2 and Chapter 5). I have brought into existence a gold-standard morpheme-level annotated corpus, which did not exist before. This corpus can be reused, for instance as training or testing data for a probabilistic or a hybrid MA system.

The creation of SANTI-morf has been a small PhD project, not comparable to the projects that led to well-known corpus annotation software such as EngCG or CLAWS. SANTI-morf is still in its infancy compared to these programs. This means on the one hand that it still needs further testing, but on the other that it has much potential for future development, as I have indicated in this concluding chapter.

Although SANTI-morf has been designed primarily for linguists, it may also have applications in NL:P. For instance, the morphological information it generates could potentially be used to improve other annotation programs such as lemmatisers, POS taggers, or syntactic parsers; or even higher-level NLP applications such as question answering systems, spelling and grammar checkers, or automatic translation software.

Nonetheless, I am confident that SANTI-morf constitutes a contribution to the field of Indonesian linguistics, particularly corpus linguistics. Indonesian linguists can now use SANTI-morf to morphologically analyse texts and corpora, index them (in NooJ or other corpus analysis programs), and thus search their data by morpheme citation forms and analytic tags as well as the original orthographic form. Applying corpus methods such as concordance, collocation, keywords or key tags, tag frequency lists, and various advanced statistical analyses based on these, is now possible. On this basis, it will become possible to ask new research questions, or

[67] http://www.NooJ-association.org/resources.html (last accessed 11/06/2021)

answer existing questions in new ways. For instance, one live issue in the study of history of Indonesian is change in the use of numeral classifiers over time. With a historical corpus tagged by SANTI-morf, analysed in today's powerful corpus software, information on the diachronic frequency distribution of these morphemes can be gathered to bring to bear on this area of inquiry. Likewise, linguists interested in morphophonology will be able to utilise the corpus annotation to test their hypotheses. One example of a present debate that this might inform relates to the citation form of the active voice prefix; some scholars argue that *meng-* should be understood to be the underlying (citation) form of this morpheme, instead of *meN-*, on the basis of its supposedly greater productivity relative to other allomorphs. Claims about productivity are, without morphologically annotated corpora, inevitably somewhat speculative; but the productivity of *meng-* may easily be measured and quantified using a corpus annotated by SANTI-morf.

Many publications on the creation of annotation systems foreground the computational or statistical elements of the system. The linguistic work involved – that is, the tagset as a reflection of the MAS the system implements – is, by contrast, often downplayed. The complexities of the MAS and its creation are rarely described in great detail. Indeed, much annotation software, including for instance xfst (see 6.10), is agnostic of any linguistic knowledge and implements whatever MAS its language resources define. This underplays the importance of the linguistic side of creating an annotation system. However, as this thesis has demonstrated (in Chapters 3), discounting the linguistic work involved in building an annotation system is misleading, because the backbone of a tagging system is its MAS.

Thus, another contribution of this thesis has been to proportionately expose the linguistic work, i.e. the creation of the SANTI-morf MAS and its lexicon and rule resources, which constituted the major part of the SANTI-morf undertaking. Most annotation systems for Indonesian (across linguistic levels) have been built by programmers rather than linguists (as I demonstrated in 3.5 and 5.2). I hope that my thesis, and SANTI-morf, will inspire more Indonesian linguists to contribute to the development of annotation systems, not only for Indonesian itself but also for other languages of Indonesia such as Javanese, Sundanese, or Balinese.

Moreover, given that corpus linguistics is still fairly new territory for Indonesian linguistics, I hope that SANTI-morf will offer a new lens for Indonesian linguists to approach their data, quantitatively or qualitatively, making the large-scale analysis of textual data via computer-assisted methods an integral part of the toolbox that researchers in many areas of study can bring to bear.

REFERENCES

Alwi, H., Dardjowidjojo, S., Lapoliwa, H., & Moeliono, M. (1998). *Tata Bahasa Baku Bahasa Indonesia (3rd Edition).* Jakarta: Balai Pustaka.

Antworth, E. L. (1990). *PC-KIMMO: A Two-Level Processor for Morphological Analysis.* Dallas: Summer Institute of Linguistics.

Aranoff, M., & Fudeman, K. (2011). *What is Morphology.* Cambridge: Cambridge University Press.

Arka, I. W. (1993). *Morpholexical Aspects of Causative -kan in Indonesian.* [Unpublished M.Phil Thesis]. University of Sydney.

Arka, I. W., Dalrymple, M., Mistica, M., Mofu, S., Andrews, A., & Simpson, J. (2009). A Linguistic and Computational Morphosyntactic Analysis of the Applicative -i in Indonesian. In M. Butt, & H. T (Eds.), *LFG09 Conference* (pp. 85-105). Stanford: Stanford University Press.

Aronoff, M. (1976). *Word Formation in Generative Grammar.* Cambridge, MA: MIT Press.

Aston, G., & Burnard, L. (1998). *The BNC handbook: exploring the British National Corpus with SARA.* Birmingham: Birmingham University Press.

Barsky, R. (2011). *Zellig Harris: from American Linguistics to Socialist Zionism.* Massachusetts: MIT Press.

Beesley, K. R., & Karttunen, L. (2003). *Finite State Morphology.* Stanford: CSLI.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python.* California: O'Reilly Publishing.

Booij, G. (2007). *The Grammar of Words: An Introduction to Linguistic Moprhology (2nd Edition).* Oxford: Oxford University Press.

Brill, E. (1995). Transformation Based Error Driven Learning and Natural Language Processing. *Computational Linguistics, 21(4)*, 543-565.

Brill, E. (1999). Corpus Based Rules. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 247-284). Dordrecht: Kluwer Academic Publishers.

Buckwalter, T. (1999). *Morphology Analysis.* www.qamus.com.

Buckwalter, T. (2001). *Buckwalter Arabic Morphological Analyzer 1.0.* Pennsylvania: Linguistic Data Consortium LDC2004L02.

Calzolari, N., McNaught, J., & Zampolli, A. (1996). *EAGLES Final Report: EAGLES Editors' Introduction.* Pisa: EAGEB-FR. https://www.issco.unige.ch/downloads/ewg96.pdf.

Carnstair-McCarthy, A. (2002). *Introduction to English Morphology: Words and Their Structure.* Edinburgh: Edinburgh University Press.

Chaer, A. (2008). *Morfologi Bahasa Indonesia.* Bandung: Rhinneka Cipta.

Chomsky, N. (1957). *Syntactic Structure.* Berlin: Walter de Gruyter.

Chomsky, N. (1965). *Aspects of the Theory of Syntax.* Cambridge, MA: MIT Press.

Chomsky, N. (2015). *The Minimalist Program.* Massachussets: MIT Press.

Chomsky, N., & Halle, M. (1968). *The Sound Pattern of English.* USA: MIT Press.

Chomsky, N., & Schützenberger, M. P. (1963). The Algebraic Theory of Context Free Languages. In P. Braffort, & D. Hirschberg (Eds.), *Computer Programming and Formal Languages* (pp. 118–161). Amsterdam: North Holland.

Chun-Hsien, C., & Honavar, V. (2000). Neural Architecture for Information Retrieval and Database Query. In R. Dale, H. Moisl, & H. Somers (Eds.), *Handbook of Natural Language Processing* (pp. 873-888). New York & Basel: Marcel Dekker.

Cloeren, J. (1999). Standards for Tagsets. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 37-54). Dordrecht: Kluwer Academic Publishers.

Coates, S. (1999). *Words and Their Structure.* New York: Routledge.

Coltekin, C. (2010). A Freely Available Morphological Analyzer for Turkish. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, & D. Tapias (Eds.), *Proceedings of the LREC 2010* (pp. 19-28). Malta: ELRA.

Creutz, M. J., & Linden, B. K. (2004). Morpheme Segmentation Gold Standards for Finnish and English. *Publications in Computer and Information Science Report A77.* Helsinki: Helsinki University of Technology.

Crystal, D. (2008). *Dictionary of Linguistics and Phonetics: 6th Edition.* Oxford: Blackwell.

Daelemans, W. (1999). Machine Learning Approaches. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 285-302). Dordrecht: Kluwer Academic Publishers.

Dawson, J. (1974). Suffix Removal and Word Conflation. *Association for Literary and Linguistic Computing (ALLC) bulletin, 2(3)*, 33-46.

Denistia, K., & Bayeen, H. (2019). The Indonesian prefixes PE-and PEN-: A study in Productivity and Allomorphy. *Morphology 29(3)*, 385-407.

Djawanai, S. (1999). Diathesis in Indonesian: Active versus Passive Construction as Reflection of a Worldview. *NUCB journal of language culture and communication, 1(2)*, 27-41.

Drewek, R., & Erni, M. (1982). LDVLIB (LEM): A System for Interactive Lemmatizing and its Application. In J. Horecky (Ed.), *Proceedings of the Ninth International Conference on Computational Linguistics* (pp. 86-89). Prague: Academia Praha.

Dukes, K., & Habash, N. (2010). Morphological Annotation of Quranic Arabic. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, & J. Odijk (Eds.), *LREC 2010* (pp. 2530-2536). Valletta: European Language Resources Association (ELRA).

Durand, J., & Katamba, F. (2014). *Frontiers of Phonology: Atoms, Structures and Derivations.* London: Routledge.

El-Beze, M., & Merialdo, M. (1999). Hidden Markov Model. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 263-283). Dordrecht: Kluwer Academic Publishers.

Ewing, M. C. (2005). *Grammar and Inferences in Conversation: Identifying Clause Structure in Javanese.* Philadelphia: John Benjamins.

Fromkin, V., Rodman, R., & Hymes, N. (2011). *An Introduction to Language (9th Edition).* New York: Wadsworth Cengage Learning.

Garside, R. (1987). The CLAWS Word-tagging System. In R. Garside, G. Leech, & G. Sampson (eds.), *The Computational Analysis of English: A Corpus-based Approach* (pp. 31-41). London: Longman.

Goh, C. L., Asahara, M., & Matsumoto, Y. (2005). Chinese word segmentation by Classification of Characters. *International Journal of Computational Linguistics & Chinese Language Processing, 10 (3)*, 200-213.

Goksel, A., & Kerslake, C. (2005). *Turkish: A Comprehensive Grammar.* London and New York: Routledge.

Goldhahn, D., Eckart, T., & Quasthoff, U. (2012). Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. In N. Calzolari, K. Choukri, T.

Declerck, M. Doğan, B. Maegaard, J. Mariani, & S. Piperidis (Eds.), *Proceedings of LREC Vol. 29* (pp. 31-43). Istambul: European Language Resources Association (ELRA).

Graff, D., Maamourio, M., Bouziri, B., Krouna, S., & Kullick, S. B. (2009). *Standard Arabic Morphological Analyzer (SAMA) Version 3.1.* Pennsylvania: Linguistic Data Concortium LDC2009E73.

Grefenstette, G. (1999). Tokenization. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 127-134). Dordrecht: Kluwer Academic Publishers.

Gross, M. (1994). Constructing Lexicon-Grammars. In S. Atkins, & A. Zampolli (Eds.), *Computational Approaches to the Lexicon* (pp. 213-263). Oxford: Oxford University Press.

Gross, M. (1997). The Construction of Local Grammars. In E. Roche, & Y. Schabes (Eds.), *Finite State Language Processing* (pp. 329-354). Massachusetts: MIT Press.

Gulsen, E. (2014). ITU Turkish Web Service. In W. Shuly, & M. B. Tadic (Ed.), *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 1-4). Gothenburg: Association for Computational Linguistics (ACL).

Habash, N. (2010). *Introduction to Arabic Natural Language Processing.* Toronto: University of Toronto.

Habash, N., Rambow, O., & Roth, R. (2009). MADA+ TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. In K. Choukri, & B. Maegard (Eds.), *Proceedings of the 2nd International Conference on Arabic Language and Tools (MEDAR) Vol 41* (p. 62). Cairo: The MEDAR consortium.

Halle, M. (1973). Prolegomena to the Theory of Word Formation. *Linguistic Inquiry, Vol 4 (1)*, 3-16.

Haspelmath, M., & Sims, A. (2013). *Understanding Morphology.* London and New York: Routledge.

Heafield, K. (2011). KenLM: Faster and Smaller Language Model Queries. In C. Callison-Burch, P. Koehn, C. Monz, & O. F. Zaidan (Eds.), *Proceedings of the Sixth Workshop on Statistical Machine Translation* (pp. 187-197). Edinburgh: ACL.

Hellberg, S. (1972). Computerized Lemmatization Without the Use of A Dictionary: A Case Study From Swedish Lexicology. *Computers and the Humanities, 6(4)*, 209-212.

Hirschman, L., & Mani, I. (2003). Evaluation. In R. Mitkov (Ed.), *The Oxford handbook of Computational Linguistics* (pp. 414-429). Oxford: Oxford University Press.

Hripcsak, G., & Rothschild, A. S. (2005). Agreement, the F-Measure, and Reliability in Information Retrieval. *Journal of the American Medical Informatics Association, 12(3)*, 296-298.

Hulden, M. (2009). Foma: a Finite-State Compiler and Library. In A. Lascarides (Ed.), *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)* (pp. 29-32). Stroudsburg, PA: EACL.

Johns, Y., & Stokes, R. (1996). *Bahasa Indonesia Book 1.* Singapore: Periplus.

Johnson, C. D. (1972). *Formal Aspects of Phonological Description.* [Unpublished PhD Thesis]. University of California Berkeley.

Jurafsky, D., & Martin, J. H. (2007). *An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Cambridge: Pearson.

Kager, R. (2004). *Optimality Theory.* Cambridge: Cambridge University Press.

Kaplan, R. M., & Kay, M. (1981). Phonological Rules and Finite-State Transducers. *Linguistic Society of America Meeting Handbook, Fifty-Sixth Annual Meeting*, (pp. 27-30).

Karlsson, F. (1990). Constraint Grammar as a Framework for Parsing Unrestricted Text. In H. Karlgren (Ed.), *Proceedings of the 13th International Conference of Computational Linguistics, Vol. 3* (pp. 168-173). Helsinki: ACL.

Karlsson, F. (1999). *Finnish: an Essential Grammar.* New York: Routledge.

Karttunen, L. (1998). The Proper Treatment of Optimality in Computational Phonology. In L. Karttunen, & K. Oflazer (Ed.), *Proceedings of International Workshop on Finite-State Methods in Natural Language Processing 1998* (pp. 1-2). Ankara: FSMNLP.

Karttunen, L., & Beesley, K. R. (1992). *Two-level Rule Compiler: Technical Report ISTL-92-2.* Palo Alto: Xerox.

Karttunen, L., & Beesley, K. R. (2001). A Short History of Two-Level Morphology. *ESSLLI-2001 Special Event titled" Twenty Years of Finite-State Morphology.* Helsinki: FoLLI.

Retrieved 08 22, 2018, from http://www.helsinki.fi/esslli/evening/20years/twol-history.html

Karttunen, L., Kaplan, R. M., & Zaenen, A. (1992). Two-Level Morphology with Composition. In A. Zampolli (Ed.), *Proceedings of COLING 92* (pp. 141-148). Nantes: ACL.

Karttunnen, L. (1993). Finite State Constraints. In J. A. Goldsmith (Ed.), *The Last Phonological Rules* (pp. 175-193). Chicago: University of Chicago press.

Kartunnen, L. (1993). *Finite State Lexicon Compiler: Technical Report. ISTL-92-2.* Palo Alto: Xerox.

Katamba, F. (1993). *Morphology.* New York: St Martin.

Klein, D., & Manning, C. (2003). Accurate Unlexicalized Parsing. In E. Hinrich, & D. Roth (Ed.), *Proceedings of the 41st Meeting of the Association for Computational Linguistics* (pp. 423-430). Sapporo: ACL.

Koskenniemi, K. (1983). *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production.* Helsinki: University of Helsinki.

Krause, W., & Willée, G. (1981). Lemmatizing German Newspaper Texts with the Aid of an Algorithm. *Computers and the Humanities Vol 15 no 2*, 101-113.

Kridalaksana, H. (1989). *Pembentukan Kata dalam Bahasa Indonesia.* Jakarta: Gramedia.

Kridalaksana, H. (2007). *Kelas Kata dalam Bahasa Indonesia.* Jakarta: Gramedia Pustaka Utama.

Larasati, S. (2012). IDENTIC CORPUS: Morphologically Enriched Indonesian-English Parallel Corpus. In N. Calzolari, K. Choukri, T. Declerck, U. Doğan, B. Maegaard, & S. Piperidis (Eds.), *Proceedings of LREC 2012* (pp. 902-906). Istanbul: ELRA.

Larasati, S., Kuboň, V., & Zeman, D. (2011). Indonesian Morphology Tool (MorphInd): Towards an Indonesian Corpus. In C. Mahlow, & M. Piotorski (Eds.), *Systems and Frameworks for Computational Morphology* (pp. 119-129). Berlin: Springer.

Lee, C.-M. (1999). Contrastive topic: A Locus of Interface Evidence from Korean and English. In K. Turner (Ed.), *The Semantics/Pragmatics Interface from Different Points of View* (pp. 317-342). Brighton: Elsevier.

Leech, G. (1997). Introducing Corpus Annotation. In R. Garside, & G. Leech (Eds.), *Corpus Annotation* (pp. 1-18). London: Longman.

Leech, G., & Smith, N. (2000). *The British National Corpus (Version 2) with Improved Word-class Tagging.* Lancaster: UCREL.

Leech, G., & Wilson, A. (1999). Standards for Tagsets. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 51-80). Dordrecht: Kluwer Academic Publishers.

Lehmann, C. (2004). Interlinear Morphemic Glossing. In G. Booij, J. Mugdan, & S. Skopeteas (Eds.), *Morphologie. Ein internationales Handbuch zur Flexion und Wortbildung* (pp. 1834-1857). Berlin & New York: W. de Gruyter.

Levelt, W. (2008). *An Introduction to the Theory of Formal Languages and Automata.* Amsterdam: John Benjamins.

Lewis, M. P. (2009). *Ethnologue: Languages of the World.* Dallas: SIL International.

Linden, K., & Pirinnen, T. (2009). Weighted Finite-State Morphological Analysis of Finnish Compounding with HFST-LEXC. In K. Jokinen, & E. Bick (Eds.), *Proceedings of the 17th Nordic Conference of Computational Linguistics NODALIDA 2009* (pp. 89-95). Odensk: Northern European Association for Language Technology (NEALT).

Lindén, K., Axelson, E., Hardwick, S., Silfverberg, M., & Pirinen, T. (2011). HFST Tools for Morphology–An Efficient Open-Source Package for Construction of Morphological Analyzers. In C. Mahlow, & M. Piotrowoski (Eds.), *Systems and Frameworks for Computational Morphology* (pp. 28-47). Berlin: Springer.

Linz, P. (2001). *An Introduction to Formal Languages and Automata.* Boston: Jones and Bartlett.

Lipka, L. (1992). *An Outline of English Lexicology.* Tubingen: Niemeyer.

Litkowsi, K. C. (2005). Computational Lexicons and Dictionaries. In K. Brown (Ed.), *Encyclopedia of Language and Linguistics* (pp. 753-761). Michigan: Elsevier.

Lovins, J. B. (1968). Development of a Stemming Algorithm. *Mechanical Translation and Computing 11 (1)*, 22-31.

Manning, C., & Schutze, H. (1999). *Foundations in Statistical Natural Language Processing.* Massachusetts: MIT Press.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics 19(2)*, 313–330.

McCarthy, J. J. (2002). *A Thematic Guide to Optimality Theory.* Cambridge: Cambridge University Press.

McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., & Bedini, C. (2013). Universal dependency annotation for multilingual parsing. In H. Schuetze, P. Fung, & M. Poesio (Eds.), *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: Vol 2* (pp. 92-97). Sofia, Bulgaria: ACL.

Meknavin, S., Charoenpornsawat, P., & Kijsirikul, B. (1997). Feature-based Thai Word Segmentation. In A. Kawtrakul, C. Pechsiri, T. Permpool, D. Thamvijit, & P. Sornpraser (Eds.), *Proceedings of Natural Language Processing Pacific Rim Symposium* (pp. 41-46). Phuket: ACL.

Meunier, J., Boisvert, J., & Denis, F. (1976). The Lemmatization of Contemporary French. In A. Jones, & R. F. Churchhouse (Eds.), *Proceedings of the Third International Symposium in the Computer in Literary and Linguistic Studies* (pp. 208-214). Cardiff: ALLC.

Mikheev, A. (2003). Text Segmentation. In R. Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics* (pp. 201-218). Oxford: Oxford University Press.

Mille, S., Burga, A., Ferraro, G., & Wanner, L. (2012). How Does the Granularity of an Annotation Scheme Influence Dependency Parsing Performance? In M. Kay, & C. Boitet (Eds.), *Proceedings of COLING 2012* (pp. 839-852). Mumbai: COLING.

Mistica, M., Andrews, A., Arka, I.-W., & Baldwin, T. (2009). Double-Double, Morphology and Trouble: Looking into Reduplication in Indonesian. In L. Pizzato, & R. Schwitter (Eds.), *Proceedings of the Australasian Language Technology Association Workshop 2009* (pp. 44-52). Canberra: ALTA.

Monachini, M., & Calzolari. (1999). Standardization in the Lexicon. In. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 149-174). Dordrecht: Kluwer Academic Publishers.

Mueller, F. (2007). Indonesian Morphology. In A. Kaye (Ed.), *Morphologies of Asia and Africa* (pp. 1207-1230). Winnona: Eisenbraums.

Neme, A. A. (2011). A Lexicon of Arabic Verbs Constructed on the Basis of Semitic Taxonomy and Using Finite-State Transducers. In B. Sagot (Ed.), *ESSLLI International Workshop on Lexical Resources* (pp. 79-86). Ljublana: WoLeR.

Neme, A., & Laporte, É. (2013). Pattern-and-Root Inflectional Morphology: The Arabic Broken Plural. *Language Sciences vol 40*, 221-250.

Nomoto, H. (2013). On the Optionality of Grammatical Markers: A Case Study of Voice Marking in Malay/Indonesian. *NUSA: Linguistic Studies of Languages in and Around Indonesia Vol.54*, 121-143.

Oflazer, K. (1994). Two-level Description of Turkish Morphology. *Literary and Linguistic Computing, 9(2)*, 137-148.

Oflazer, K. (1999). Morphological Analysis. In H. van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 175-205). Dordrecht: Kluwer Academic Publishers.

Oflazer, K. (2018). Morphological Processing for Turkish. In K. Oflazer, & M. Saraclar (Eds.), *Turkish Natural Language Processing* (pp. 21-52). New York: Springer.

Oflazer, K., & Gokhan, T. (1997). Morphological Disambiguation by Voting Constraints. In P. R. Cohen, & W. Wahlster (Eds.), *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics* (pp. 222-229). Madrid: ACL.

Oflazer, K., & Kuruoz, L. (1994). Tagging and Morphological Disambiguation of Turkish Texts. In P. Jacobs (Ed.), *the Fourth Conference on Applied Natural Language Processing (ANLP94)* (pp. 144-149). Stuttgart: ACL.

Oflazer, K., Yeniterzi, R., & Durgar-El-Kahlout, I. (2018). Statistical Machine Translation and Turkish. In K. Oflazer, & M. Saraclar (Eds.), *Turkish Natural Language Processing* (pp. 207-237). New York: Springer.

O'Halloran, M., & Waite, W. M. (1966). Note on Rapid Instruction Analysis by Table Lookup. *The Computer Journal, 9(3)*, 248-249.

Oztaner, R. M. (1996). *A Word Grammar for Turkish with Morphophonemic Rules.* Ankara: Middle East Technical University.

Palmer, D. (2000). Tokenisation and Sentence Segmentation. In R. Dale, H. Moisl, & H. Somers (Eds.), *Handbook of Natural Language Processing* (pp. 11-36). New York & Basel: Marcel Dekker.

Palmer, D. D. (1997). A Trainable Rule-Based Algorithm for Word Segmentation. In P. R. Cohen, & W. Wahlster (Eds.), *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 21-328). Madrid: ACL.

Pasha, A., Al-Badrashiny, M., Diab, M.-T., El Kholy, A., Eskander, R., Habash, N., & Roth, R. (2014). In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, & D. Tapias (Eds.), *MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic* (pp. 1094-1101). Rejkavic: European Language Resources Association (ELRA).

Paumier, S. (2014). *Unitex Manual ver 3.1.* Paris: Universite Paris Est Marne La Valee & LADL.

Payne, T. E. (1997). *Describing morphosyntax: A Guide for Field Linguists.* Cambridge: Cambridge University Press.

Pisceldo, F., Mahendra, R., Manurung, R., & Arka, I. W. (2008). A Two Level Morphological Analyser for the Indonesian Language. In N. Stokes, & D. Powers (Eds.), *Proceedings of Australasia Technology Association Workshop* (pp. 142-150). Hobart: ACL.

Porter, M. (1980). Algorithm for Suffix Stripping. *Program 13(3)*, 130-137.

Prentice, D. J. (1987). Malay (Indonesian and Malaysian). In B. Comrie (Ed.), *The Major Languages of East and Southeast Asia* (pp. 185-208). London: Routledge.

Prince, A., & Smolensky, P. (2004). *Optimality Theory: Constraint Interaction in Generative Grammar.* Oxford: Blackwell.

Rashel, F., Dinakaraman, A., & Luthfy, A. (2014). In M. Dong, Y. Lu, R. E. Banchs, & B. Ranaivo-Malancon (Eds.), *Building an Indonesian Rule-Based Part-of-Speech Tagger* (pp. 70-73). Sarawak: IALP.

Roark, B., & Sproat, M. (2001). *Computational Approaches to Morphology and Syntax.* Oxford: Oxford University Press.

Ryding, K. C. (2005). *A Reference Grammar of Modern Standard Arabic.* Cambridge: Cambridge university press.

Ryding, K. C. (2014). *Arabic: A Linguistic Introduction.* Cambridge: Cambridge University Press.

Sahin, M., Sulubacak, U., & Eryigit, G. (2013). Redefinition of Turkish morphology using flag diacritics. *Proceedings of The Tenth Symposium on Natural Language Processing.* Phuket: Thammasat University.

Sak, H., Güngör, T., & Saraçlar, M. (2008). Turkish Language Resources: Morphological Parser, Morphological Disambiguator and Web Corpus. In B. Nordström, & A. Ranta (Eds.), *International Conference on Natural Language Processing* (pp. 417-427). Berlin, Heidelberg: Springer.

Samuelsson, C., & Wiren, M. (2000). Parsing Techniques. In R. Dale, & H. S. Moisl (Eds.), *Handbook of Natural Language Processing* (pp. 59-92). New York & Basel: Marcell Dekker.

Sawalha, M., Atwell, E., & Abushariah, M.-A. (2013). SALMA: Standard Arabic Language Morphological Analysis. *Proceedings of 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)* (pp. 1-6). Sharjah: IIEE.

Scalise, S. (1986). *Generative Morphology.* New York: Walter de Gruyter.

Schiller, A., & Kartunnen, L. (1999). Lexicons for Tagging. In H. Van-Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 135-147). Dordrecht: Kluwer Academic Publishing.

Selkirk, E.-O. (1983). *The Syntax of Words.* Massachusetts: MIT Press.

Shiohara, A. (2012). Applicatives in Standard Indonesian. In W. Nakamura, & R. Kikusawa (Eds.), *Senri Ethnological Studies 77* (pp. 59-76). Osaka: National Museum of Ethnology.

Siegel, D. (1979). *Topics in English morphology.* New York: Garland.

Silberztein, M. (1993). *Dictionnaires Electroniques et Analyse Automatique de Textes: Le Systeme INTEX.* Paris: Masson.

Silberztein, M. (1997). *The INTEX manual, 1997.* Université de Franche-Comté: available at: http://intex.univ-fcomte.fr/.

Silberztein, M. (2003). *NooJ Manual.* Available at Download: www.nooj-association.org.

Silberztein, M. (2016). *Formalizing Natural Languages: Nooj Approach.* London: Wiley.

Sinclair, J. (1996). *EAGLES Preliminary Recommendations on Corpus Typology'.* Available at: http://www.ilc.cnr.it/EAGLES96/corpustyp/corpustyp.html.

Sneddon, J. N. (1996). *Indonesian: A Comprehensive Grammar.* New York: Routledge.

Sneddon, J. N., Adelaar, A., Djenar, D.-N., & Ewing, M.-C. (2010). *Indonesian Reference Grammar:2nd Edition.* New South Wales: Allen & Unwin.

Soderberg, C. D., & Olson, K. S. (2008). Indonesian. *Journal of the International Phonetic Association 38(2)*, 209-213.

Spencer, A. (1991). *Morphological Theory.* Oxford: Blackwell.

Sproat, R. (2000). Lexical Analysis. In R. Dale, H. Moisl, & Somers (Eds.), *Handbook of Natural Language Processing* (pp. 37-57). New York: Marcell Decker.

Sproat, R. W., Chilin, S., Gale, W., & Chan, N. (1996). A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. *Computational Linguistics, 22(3)*, 377–404.

Uzawa, H. (2007). A study on the Pragmatic Function of ialah and adalah in Malay. In Y. Kawaguchi, M. Minegishi, & J. Durand (Eds.), *Corpus Analysis and Variation* (pp. 315-338). John Benjamins: Amsterdam.

Van Halteren, H. (1999). Performance of Taggers. In H. Van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 81-94). Dordrecht: Kluwer Academic Publisher.

Van Halteren, H., & Voutilanen, A. (1999). Automatic Taggers: An Introduction. In H. Van Halteren (Ed.), *Syntactic Wordclass Tagging* (pp. 109-116). Dordrecht: Kluwer Academic Publishers.

Veronis, J., & Khouri, L. (1995). Etiquetage Grammatical Multilingue: Le Projet MULTEXT. *TAL. Traitement Automatique Des Langues 36*, 233-248.

Voutilanen, A. (1999). Hand-Crafted Rules. In H. Van Halteren (Ed.), *Syntactic wordclass tagging* (pp. 217-247). Dordrecht: Kluwer Academic Publishers.

Voutilanen, A. (1999). Orientation. In H. Van Halteren (ed.), *Syntactic Wordclass Tagging* (pp. 3-9). Dodrecht: Kluwer Academic Publishers.

Voutilanen, A. (2003). Part of Speech Tagging. In R. Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics* (pp. 219-232). Oxford: Oxford University Press.

Voutilanen, A., Heikilla, J., & Antilla, A. (1992). *Constraint Grammar of English: A Performance Oriented Introduction.* Helsinki: University of Helsinki.

Voutilanen, A., Purtonen, T., & Muhonen, K. (2012). *FinnTreeBank2 Manual.* Helsinki: University of Helsinki.

Wilcox, G. (2009). *Introduction to Linguistic Annotation.* Helsinki: Morgan & Claypool.

Wintner, S. (2013). Formal Language Theory. In A. Clark, C. Fox, & S. Lappin (Eds.), *The Handbook of Computational Linguistics and Natural Language Processing.* (pp. 11-42). Oxford: Blackwell.

Wissler, L., Almashraee, M., Díaz, D.-M., & Paschke, A. (2014). The Gold Standard in Corpus Annotation. *IIEE Graduate Student Conference (GSC).* Passau: IIEE.

Wong, P.-K., & Chan, C. (1996). Chinese Word Segmentation Based on Maximum Matching and Word Binding Force. In J. Tsujii (Ed.), *COLING 1996 Volume 1: The 16th International Conference on Computational* (pp. 200-203). Copenhagen: Coling.