Natural Language Engineering **1** (1): 1–45. Printed in the United Kingdom

UNLT: Urdu Natural Language Toolkit

JAWAD SHAFI[1,2], HAFIZ RIZWAN IQBAL[3], RAO MUHAMMAD ADEEL NAWAB[2], and PAUL RAYSON[1]

[1] *Lancaster University, Lancaster, U.K.*
[2] *COMSATS University Islamabad, Lahore Campus, Pakistan*
[3] *Information Technology University, Lahore, Pakistan*
E-mails: p.rayson,j.shafi@lancaster.ac.uk, adeelnawab,jawadshafi@cuilahore.edu.pk, rizwan.iqbal@itu.edu.pk

(*Received 31 December 2017; revised 15 November 2021*)

Abstract

This study describes a Natural Language Processing (NLP) toolkit, as the first contribution of a larger project, for an under-resourced language - Urdu. In previous studies, standard NLP toolkits have been developed for English and many other languages. There is also a dire need for standard text processing tools and methods for Urdu, despite it being widely spoken in different parts of the world with a large amount of digital text being readily available. This study presents the first version of the *UNLT (Urdu Natural Language Toolkit)* which contains three key text processing tools required for an Urdu NLP pipeline; word tokenizer, sentence tokenizer and Part-Of-Speech (POS) tagger. The UNLT word tokenizer employs a morpheme matching algorithm coupled with a state-of-the-art stochastic n-gram language model with back-off and smoothing characteristics for the space omission problem. The space insertion problem for compound words is tackled using a dictionary look-up technique. The UNLT sentence tokenizer is a combination of various machine learning, rule-based, regular-expressions and dictionary look-up techniques. Finally, the UNLT part-of-speech taggers are based on Hidden Markov Model and Maximum Entropy based stochastic techniques. In addition, we have developed large gold standard training and testing datasets to improve and evaluate the performance of new techniques for Urdu word tokenization, sentence tokenization and POS tagging. For comparison purposes we have compared the proposed approaches with several methods. Our proposed UNLT, the training and testing datasets, and supporting resources are all free and publicly available for academic use.

1 Introduction

A natural language toolkit is a library or framework used to analyse human language in a statistical, rule-based or hybrid Natural Language Processing (NLP). These toolkits have been used in the development of a range of applications from various domains. For instance, life sciences and medicine (Cunningham et al., 2013), bio-informatics (Ferrucci and Lally, 2004), computer science (Maynard et al., 2015;

Rush et al., 2015), linguistics (Gries and John, 2014), Machine Learning (ML) (Bird et al., 2008), and the analysis of social media (Dietzel and Maynard, 2015).

The majority of existing NLP toolkits are for English with many other languages supported (Cunningham et al., 2002; Bird et al., 2009; Kwartler, 2017; Manning et al., 2014). However, there is a lack of standard text processing tools and methods for South Asian languages, particularly Urdu, which has 300 million speakers around the world (Riaz, 2009) for which a large amount of digital text is available through online repositories. Urdu is an Indo-Aryan[1] (or Indic) language derived from Sanskrit/Hindustani language (Bögel et al., 2007), has been heavily influenced by Arabic, Persian (Bögel et al., 2007) and less by Turkic (Chagatai[2]) languages for literary and technical vocabulary (Sharjeel et al., 2017), and is written from right to left in Nastaliq style (Shafi, 2020). Urdu is a morphologically rich language (Saeed et al., 2012), including many multi-word expressions and letters which may change their shape based on context, which makes the tokenization task very complex and challenging. Moreover, it is a free word order language (Daud et al., 2016; Mukund et al., 2010; Riaz, 2012).

In the previous literature, a small amount of work has been carried out to propose systematic text processing NLP approaches for Urdu including word tokenization (Rashid and Latif, 2012; Durrani and Hussain, 2010; Lehal, 2010; Rehman and Anwar, 2012), sentence tokenization (Raj et al., 2015; Rehman and Anwar, 2012), and POS tagging (Hardie, 2004; Anwar et al., 2007b,a; Sajjad and Schmid, 2009; Muaz et al., 2009; Ahmed et al., 2014) (see Section 2 for details). However, there are a number of limitations of these existing Urdu NLP studies: (i) most of them are not formed into NLP tools, and the ones that are implemented are not publicly and freely available, (ii) training/testing datasets along with developed resources are not always freely and publicly available to improve, compare, and evaluate new and existing methods (Daud et al., 2016), (iii) they have been trained and tested on very small datasets (Durrani and Hussain, 2010; Anwar et al., 2007a; Rehman and Anwar, 2012), (iv) the efficiency of dictionary based word tokenization approach (Rashid and Latif, 2012) is entirely dependent on a dictionary of complete Urdu words, which is not practically possible to produce for the Urdu language, (v) statistical n-gram based word tokenizers (Durrani and Hussain, 2010; Rehman et al., 2013) cannot handle unknown words or back off to a lesser contextual models, (vi) sentence tokenization methods require datasets to train machine learning algorithms and these are unavailable (Raj et al., 2015; Rehman and Anwar, 2012), (vii) current rule-based POS tagging methods (Hardie, 2004) are closely tailored to a particular dataset, therefore, not portable across different domains, (viii) smoothing and other features to handle unknown words in statistical POS taggers have not been thoroughly explored, (ix) less contextual POS tagging techniques have been proposed, and (x) POS tagsets which have been used to train/test statistical POS

[1] `https://en.wikipedia.org/wiki/Indo-Aryan_languages#cite_note-ethnologue-4` - Last checked: 20-March-2019

[2] `https://en.wikipedia.org/wiki/Urdu` - Last checked: 15-September-2020

taggers (Anwar et al., 2007b; Hardie, 2004; Sajjad and Schmid, 2009) have several shortcomings (see Section 4.3.1).

To overcome the limitations of existing Urdu NLP studies, this study presents the UNLT (Urdu Natural Language Toolkit), initially with text processing methods and three NLP tools including a word tokenizer, sentence tokenizer, and POS tagger. Our proposed Urdu word tokenizer is based on a novel algorithm which makes use of rule-based morpheme matching, n-gram statistical model with backoff and smoothing characteristics, and dictionary look-up. It is trained on our proposed dataset of 1,361K tokens and evaluated on our proposed test dataset containing 59K tokens. The UNLT sentence tokenizer is a combination of rule-based, regular expressions, and dictionary lookup techniques which are evaluated on our proposed test dataset of 8K sentences. Furthermore, we have also proposed a ML based sentence tokenizer. Finally, the UNLT POS tagger is mainly based on Hidden Markov and Maximum Entropy models, with multiple variations based on smoothing, suffix length, context window, word number, lexical and morphological features. The set of UNLT POS taggers were trained on our proposed training dataset of 180K tokens and evaluated on our proposed test dataset of 20K tokens.

The UNLT and our training/testing datasets will be crucial in: (i) fostering research in an under-resourced language i.e. Urdu, (ii) the development and evaluation of Urdu word tokenizers, sentence tokenizers and POS taggers, (iii) facilitating comparative evaluations of existing and future methods for Urdu word tokenization, sentence tokenization and POS tagging, (iv) the development of various NLP tools and applications in other areas such as information retrieval, corpus linguistics, plagiarism detection, semantic annotation etc. and (v) providing a framework in which other Urdu NLP tools can be integrated.

The rest of the article is organised as follows: Section 2 describes related work and Section 3 presents challenges for word tokenization, sentence tokenization, and POS tagging methods. Section 4 explain the proposed UNLT modules. Whereas, Section 5 present the proposed datasets. Section 6 introduces evaluation measures, results and their analysis. Section 7 presents a summary and future directions of our research.

2 Related work

2.1 Existing Urdu word tokenization approaches

In the existing literature, we find only a few studies which have addressed the problem of word tokenization for the Urdu language, these are (Rashid and Latif, 2012; Durrani and Hussain, 2010; Lehal, 2010; Rehman and Anwar, 2012). The study in (Rashid and Latif, 2012) performs Urdu word tokenization in three phases. First, Urdu words are tokenized based on spaces, thus returning the cluster(s) of valid (single word) and invalid (merged word(s)[3]) Urdu words. Next, a dictionary is checked against valid/invalid words to assure the robustness of the word(s). If the

[3] Combination of many words

word is present in the dictionary then it will be considered as a valid Urdu word, returning all single words. However, if the word is not matched in the dictionary then it is considered as a merged word, hence, needing further segmentation. In the second phase, the merged words are divided into all possible combinations, to check the validity of each produced combination through dictionary lookup. If it is present in the dictionary it will be considered as a valid word. The first two phases solve the problem of space omission (see Section 4.1), the third phase addresses the space insertion problem by combining two consecutive words and checking them in the dictionary. If the compound word is found in the dictionary, then it will be considered as a single word. This technique of word tokenization was tested on 11,995 words with a reported error rate of 2.8%. However, the efficiency of this algorithm is totally dependent on the dictionary (used to check whether a word is valid or not) and it is practically not possible to have a complete dictionary of Urdu words. Furthermore, if a valid word is not present in the dictionary then this technique will mark it as invalid, which will be wrong.

Durrani and Hussain (2010) proposed a hybrid Urdu word tokenizer which works in three phases. In the first phase, words are segmented based on space, thus, returning a set of an orthographic word(s)[4]. Further, a rule-based maximum matching technique is used to generate all possible word segmentations of the orthographic words. In the second phase, the resulting words are ranked using minimum word heuristics, *uni*, and *bi*-grams based sequence probabilities. In the first two phases, the authors solved the space omission problem (see Section 4.1). In the third phase, the space insertion problem is solved to identify compound words by combining words using different algorithms. The proposed Urdu word tokenizer is trained on 70K words, whereas it is tested on a very small dataset of 2,367 words reporting an overall error rate of 4.2%. Although the authors have reported a very low error rate, this study has some serious limitations: (i) the evaluation is carried out on a very small dataset, which makes the reported results less reliable in terms of how good the word tokenizer will perform on real-world data, (ii) using a statistical n-gram technique which may ultimately lead to data sparseness, and (iii) it does not tokenize Urdu text correctly even for short texts.

Another online CLE Urdu word tokenizer is available through a website[5], which allows tokenization of up to 100 words. Its implementation details are not provided. It reports an accuracy of 97.9%. However, the link is not always available[6], and its API is not freely available[7]. We applied the CLE online Urdu work tokenizer on three randomly selected input short texts and they all were incorrectly tokenized with many mistakes.

The research cited in (Lehal, 2010) takes an approach to Urdu word tokenization, based on the Hindi language. The authors tokenized Urdu words after transliterat-

[4] One orthographic word may eventually give multiple words and multiple orthographic words may combine to give a single word.
[5] `http://182.180.102.251:8080/segment/` - Last checked: 24-June-2018
[6] See: `http://182.180.102.251:8080/segment/` - As on: 11-April-2018
[7] `http://www.cle.org.pk/clestore/segmentation.htm` - Last checked: 24-June-2018

ing them from Hindi, as the Hindi language uses spaces consistently as compared to its counterpart Urdu. They also addressed and resolved the space omission problem for Urdu in two phases. In the first phase, Urdu grammar rules have been applied to decide if the Urdu adjacent words have to be merged or not. If the grammatical rules analyser provides a definite answer that two adjacent words can be joined or not, then no further processing is required. However, if the rule-based analyser is not confident about two words either it can be joined or not, then the second phase is invoked. In the second phase, Urdu and Hindi *uni*-gram and *bi*-gram bilingual lexical resources are used to make the final decision i.e. either we need to join the two adjacent words or not. This technique of Urdu word tokenization used 2.6 million words as training data, whereas, it was tested on 1.8 million tokens. The results show an error rate of 1.44%. The limitations of this study are: (i) the problem of space insertion has not been addressed, (ii) this approach requires large bilingual corpus which is difficult to create particularly for under-resourced languages like Urdu and Hindi.

Rehman et al. (2013) proposed an Urdu word tokenizer by using rule-based (maximum matching) with n-gram statistical approach. This approach to Urdu word tokenization uses several different algorithms to solve the problem of space omission and insertion. Firstly, the forward maximum matching algorithm is used to return the list of individual tokens of Urdu text. Secondly, the Dynamic Maximum Matching (DMM) algorithm returns all the possible tokenized sequences of the Urdu text, segments are ranked and the best one is accepted. Thirdly, DMM is combined with the *bi*-gram statistical language model. These three algorithms are used to solve the space omission problem, whereas, for the space insertion problem, six different algorithms were used. The authors used 6,400 tokens for training and 57,000 tokens for testing. This approach has produced up to 95.46% F_1 score. Furthermore, the algorithms are based on probabilities which may result in zero probability being assigned to some unknown words. The authors have not handled such cases with either back off or other smoothing estimators.

To overcome the limitations of the existing studies, our study proposes a novel Urdu word tokenization algorithm using a rule-based morpheme matching approach, with off-the-shelf statistical *tri*-gram language model with back-off and smoothing characteristics for the space omission problem, whereas, the space insertion problem has been solved using dictionary lookup technique (see Section 4.1.2). To train and test our proposed word tokenizer we developed benchmark training (contains 1.65 million tokens) and testing (contains 59K tokens) datasets. Our word tokenizer and training/testing datasets are freely and publicly available for download (see Section 7 for details).

2.2 Sentence tokenization approaches

The problem of Urdu sentence tokenization has not been thoroughly explored and we found only two studies (Rehman and Anwar, 2012; Raj et al., 2015) which address the issue. Rehman and Anwar (2012) used a hybrid approach that works in two stages. First, a *uni*-gram statistical model was trained on annotated data. The

trained model was used to identify word boundaries on a test dataset. In the second step, the authors used heuristic rules to identify sentence boundaries. This study achieved up to 99.48% precision, 86.35% recall, 92.45% F_1, and 14% error rate, when trained on 3,928 sentences, however, the authors did not mention any testing data. Although this study reports an acceptable score, it has some limitations; (1) the error rate is high (14%), (2) the evaluation is carried out on a very small dataset, which makes the reported results less reliable and it is difficult to tell how well the sentence tokenizer will perform on real test data, and (3) the trained model along with training/testing data are not publicly available.

In another study, Raj et al. (2015) used an Artificial Neural Network along with POS tags for sentence tokenization in two stages. In the first phase, a POS tagged dataset is used to calculate the word-tag probability (P) based on the general likelihood ranking. Furthermore, the POS tagged dataset along with probabilities was converted to bipolar descriptor arrays[8], to reduce the error as well as training time. In the next step, these arrays along with frequencies were then used to train feed forward Artificial Neural Network using back propagation algorithm and delta learning rules. The training and testing data used in this study are 2,688 and 1,600 sentences, respectively. The results show 90.15% precision, 97.29% recall and 95.08% F_1-measure with 0.1 threshold values. The limitations of this study are that the evaluation is carried out on a small set of test data, and the trained model, as well as the developed resources, are not publicly available.

Again, similar to the Urdu word tokenization problem (see Section 2.1), the developed Urdu sentence tokenizers along with training/testing datasets are not publicly available. To fill this gap, our contribution here is an Urdu sentence tokenizer which is a combination of rule-based, regular expressions and dictionary lookup techniques, along with training (contain 12K sentences) and testing (containing 8K sentences) datasets, all of which are free and publicly available for research purposes. Moreover, we have also proposed a novel supervised ML based sentence tokenizer by extracting various features.

2.3 Part-Of-Speech tagging approaches

Similar to Urdu word (see Section 2.1) and sentence tokenization (see Section 2.2), the problem of Urdu POS tagging has not been thoroughly explored. We found only six studies (Hardie, 2004; Anwar et al., 2007b,a; Sajjad and Schmid, 2009; Muaz et al., 2009; Ahmed et al., 2014) which addressed the issue.

A pioneering piece of research on Urdu POS tagging is described in Hardie (2004). This work focused on the development of a uni-rule POS tagger, which consists of 270 manual crafted rules. The author used a POS tagset with 350 tags (Hardie, 2003). The training data consists of 49K tokens, whereas, testing was carried out on two different datasets containing 42K and 7K tokens. The reported average accuracy for the 42K tokens is 91.66%, whereas, for the 7K corpus the average accuracy is

[8] If P $>0.1 \implies$ P$\equiv -1$, If P $== 0.1 \implies$ P$\equiv 0$, If P $<0.1 \implies$ P$\equiv +1$.

89.26% with a very high ambiguity level (3.09 tags per word). However, the POS tagset used in this study has several limitations (see Section 4.3), and therefore, cannot be used for a grammatical tagging task, and having a large number of POS tags with a relatively small training data will affect the accuracy, and manually deducing rules is a laborious and expensive task.

The first stochastic POS tagger for the Urdu language was developed in 2007 (Anwar et al., 2007a). They have proposed a POS tagger based on a *bi*-gram Hidden Markov Model with back off to *uni*-gram model. Two[9] different POS tagsets were used. The reported average accuracies for the 250 POS tagset and 90 POS tagset are 88.82% and 92.60% respectively. Both were trained on a dataset of 1,000 words, however, the authors have not provided any information about the test dataset. As before, this study has several limitations; the POS tagset of 250 tags has several grammatical deficiencies (see Section 4.3), the information about the proposed tagset of 90 tags is not available, the system was trained and tested on a very small dataset, which shows that it is not feasible for morphological rich and free word order language i.e. Urdu, and used less contextual *bi/uni*-gram statistical models.

Anwar et al. (2007b) have developed an Urdu POS tagger using *bi*-gram Hidden Markov Model. The authors proposed six *bi*-gram Hidden Markov based POS taggers with different smoothing techniques to resolve data sparseness. The accuracy of these six models varies from 90% to 96%. For each model, they used a POS tagset of 90 tags. However, the authors have not mentioned the size of training/test datasets. This study has several limitations as, like the one in (Anwar et al., 2007a) the authors have used a 350 POS tagset, which has several misclassifications (see Section 4.3), the training/testing data split is unknown to readers, and limited smoothing estimators have been used, it used *bi*-gram language model (i.e. less contextual), and suffix information has not been explored.

The authors in (Sajjad and Schmid, 2009) trained Trigrams-and-Tag (TnT) (Brants, 2000), Tree Tagger (TT) (Schmid, 1994b), Random Forest (RF) (Schmid and Laws, 2008) and Support Vector Machine (SVM) (Giménez and Marquez, 2004) POS taggers, using a tagset containing 42 POS tags. All these stochastic Urdu POS taggers were trained on a 100K word dataset, whereas for testing only 9K words were used. The reported accuracy for TnT, TT, RF, and SVM are 93.40%, 93.02%, 93.28% and 94.15% respectively. In terms of limitations, they used a POS tagset of 42 tags which has several grammatical irregularities (see Section 4.3).

In another study (Muaz et al., 2009), stochastic Urdu POS taggers are presented i.e. TnT and TT tagger. These taggers are trained and tested on two different datasets with the following statistics: (i) First dataset consists of 101,428 tokens (4,584 sentences) and, 8,670 tokens (404 sentences) for training and testing respectively, and (ii) the second dataset consists of 102,454 tokens (3,509 sentences) and 21,181 tokens (755 sentences) for training and testing respectively. The reported accuracy for the first dataset is 93.01% for TnT tagger, whereas 93.37% for TT tagger. For the second dataset, TnT tagger produced 88.13% accuracy and TT had

[9] The first POS tagset contains 250 POS tags (Hardie, 2003), whereas, the second one consists of 90 tags (details are not given)

90.49% accuracy. Similar to other studies, it employed a POS tagset which has several grammatical problems (see Section 4.3), meaning that it is no longer practical for Urdu text.

The authors in (Ahmed et al., 2014) have proposed an Urdu POS tagger[10] which is based on Decision Trees and smoothing technique of Class Equivalence, using a tagset of 35 POS tags. It is trained and tested on the CLE Urdu Digest corpus[11], training and test data split is 80K and 20K tokens, respectively. However, this POS tagger is only available through an online interface, which allows tagging of 100 words. It is trained on a relatively small dataset that is not freely available. The Decision Tree statistical models are less accurate for Urdu text as compared to HMM etc. Sajjad and Schmid (2009) (see Section 6.3).

In contrast, our study contributes a set of Urdu POS taggers along with large training (containing 180K POS tagged tokens) and testing (containing 20K POS tagged tokens) datasets. Our proposed POS taggers are based on two machine learning techniques, *tri*-gram Hidden Markov Model (HMM), and Maximum Entropy (MaEn) based models. Each of our proposed HMM and MaEn models is a combination of different backoff, smoothing estimators, suffix and other types of binary valued features. To the best of our knowledge, these models along with smoothing, backoff, suffix and binary valued features have not been explored previously for the Urdu language.

3 Challenges of Urdu NLP tools

3.1 Challenges for word tokenization

Is a challenging and complex task for the Urdu language due to three main problems (Durrani and Hussain, 2010): (i) the space omission problem - Urdu uses *Nastalique* writing style and *cursive script*, in which Urdu text does not often contain spaces between words, (ii) the space insertion problem - *irregular* use of spaces within two or more words and (iii) ambiguity in defining Urdu words - in some cases Urdu words lead to an ambiguity problem because there is no clear agreement to classify them as a single word or multiple words.

The first two problems stated above, mostly arise due to the nature of Urdu characters, which are divided into: (i) *joiner* (non-separators), and (ii) *non-joiner* (separators). Non-joiner characters,[12] only merge themselves with their preceding character(s). Therefore, there is no need to insert space or Zero Width Non Joiner (ZWNJ; an Urdu character which is used to keep the word separate from their following) if a word ends with such characters. These can form isolated shapes besides final shape, whereas, joiner characters[13] can form all shapes (isolated, initial,

[10] `http://182.180.102.251:8080/tag/` - Last checked: 09-July-2018

[11] `http://www.cle.org.pk/clestore/urdudigestcorpus100k.htm` - Last checked: 09-July-2018

[12] آ، ا، د، ڈ، ذ، ر، ز، ڑ، ژ، و، ے (Transliteration: alif_mad, alif, daal, ddaal, Zaal, ray, zay, rray, jay, wao, bari_ye). All Urdu characters and word are transliterated as given in (Tafseer, 2009).

[13] ب، پ، ت، ٹ، ث، ج، چ، ح، خ، س، ش، ص، ض، ط، ظ، ع، غ، ف، ق، ک، گ، ل، م، ن، ہ، ء، ہ، ی" (Translitera-

medial and final) (Bhat et al., 2012) with respect to their neighbouring letter(s). For instance, the Urdu character خ (khay) is a joiner and has four shapes: (i) isolated خ (khay) e.g. خوخ (KHOKH 'peach') i.e. it can be seen that at the end of a word, if the character is a joiner and its preceding character is non-joiner, it will form an isolated shape, (ii) final ـخ (khay) e.g. مـخ (MKH 'brain'), it can be observed that at the end of a word, if the character is a joiner, it acquires the final shape when leading a joiner, (iii) medial ـخـ (khay) e.g. بـخَـار (BKHAR 'fever'), in other words, it shows that in the middle of a word, if the character is a joiner, it will form the medial shape when the preceding character is a joiner, (iv) initial خـ (khey) e.g. خوف (KHOF 'fear'), it shows that at the start of a word, if the character is a joiner, it acquires the initial shape when following a non-joiner. Furthermore, the Urdu character ذ (zaal) is a non-joiner, thus has only two shapes: (i) isolated ذ (zaal) e.g. ذاکر ('Zakir'), it can be noticed that at the beginning of a word, if the character is a non-joiner, it acquires isolated shape when following a joiner, (ii) final ـذ (zaal) e.g. لـذْيـذْ (LZYZ 'delicious'), it can be examined that at the end of a word, if the character is non-joiner, it acquires final shape when preceding a joiner character. The shapes that these characters (joiner or non-joiners) acquire totally depend upon the context.

A reader can understand a text if a word which ends on a joiner character is separated by a space وہ شہر (OH SHHR, 'that city') or ZWNJ character[14] نئی سائکل ہي (NYY SAYYKL HE, 'is new bicycle'). Likewise, the dropping of either of them (space or ZWNJ) will result in a visual incorrect[15] text, وہشہر (OH SHHR, 'that city') and نئسائکلهي (NYY SAYYKL HE, 'is new bicycle'), thus being perceived as a single word even though they are two and three different words, respectively. On the other hand, a word which ends on a non-joiner character does not merge itself with other words, for instance, کمپوٹر انٹرنٹ (KMPYOTR ANTRNYT, 'computer internet') and مددکرو (MDDKRO, 'help him'), even if we remove space or ZWNJ character. Note that the کمپوٹرانٹرنٹ (KMPYOTR ANTRNYT, 'computer internet') and مددکرو (MDDKRO, 'do help') are also incorrect text, each of them is a combination of two words. As, مددکرو (MDDKRO, 'do help') is مدد (MDD, 'help') and کرو (KRO, 'do'), whereas, کمپوٹرانٹرنٹ (KMPYOTR ANTRNYT, 'computer internet') have کمپوٹر (KMPYOTR, 'computer') and انٹرنٹ (ANTRNYT, 'internet') words. However, omitted space(s) between all ambiguous text results in a space omission problem, which can be overcome by inserting a space at the end of

tion: bay, pay, tay, ttay, say, jeem, chay, hay, khay, seen, sheen, suad, zuaad, tuay, zuay, ain, ghain, fay, qaaf, kaaf, laam, meem, noon, hay_gol, hamza, hey_dochasmi, chooti-ye) for such characters, it is needed to insert a space between words or ZWNJ at the end of the first word, otherwise it will join itself with the following word.

[14] Non-printing character (U+200C) is used for computer writing systems.

[15] Human readable but words that are merged into a single token.

the first word so that two or three distinct words can be detected. For example, نئسائىكلهي (NYY SAYYKL HE, 'is new bicycle) are three distinct words, written without spaces, in order to tokenize them properly we need to insert spaces at the end of نئى (NYY, 'new') , and سائىكل (SAYYKL 'bicycle') so that three different tokens can be generated: (i) نئى (NYY, 'new'), (ii) سائىكل (SAYYKL, 'bicycle'), and (iii) ہي (HY, 'is'). As it can be noted from the above discussion, space omission problems are complex thus making the Urdu word tokenization task particularly challenging.

In the space insertion problem, if the first word ends either on a joiner or non-joiner, a space at the end of the first word (see Table 1, Correct column– incorrect multiple tokens with space (-), but correct shape) can be inserted for several reasons: (i) affixes can be separated from their root, (ii) to keep separate Urdu abbreviations when transliterated, (iii) increase readability for Urdu proper nouns and English/foreign words are transliterated, (iv) compound words and reduplication morphemes do not visually merge and form a correct shape and (v) to avoid making words written incorrectly or from combining (see Table 1, incorrect column– single token but incorrect shape). For example, خوش اخلاق (KHOSH AKHLAK, 'polite') is a compound word of type affixation, however, space was inserted between خوش (KHOSH, 'happy') i.e. a prefix (literally 'happy') and اخلاق (AKHLAK, 'ethical') i.e. root to increase the readability and understandability. To identify خوش اخلاق (KHOSH AKHLAK, 'polite') as a single word/token the tokenizer will need to ignore the space between them. This also serves to emphasise the fact that the space insertion problem is also a very challenging and complex task in Urdu word tokenization.

As discussed earlier, in some cases Urdu words are harder to disambiguate. There is no clear agreement on word boundaries in a few cases (sometimes they are considered as a single word even by a native speaker). For example the compound word, وزیر اعلی (OZYR AALY, 'chief minister'), بہن بھائی (BHN BHAYY, 'sibling', literally 'brother sister'). The same is the case for reduplications, فر فر (FR FR, 'fluent') and affixation, بد اخلاق (BD AKLAK, 'depravedly'). Certain function words (normally case markers, postpositions, and auxiliaries) can be written jointly e.g. اسمیں (ASMYN, 'herein'), یہوقت (YHOKT, 'this time') or ہوگی (Ho GEE). Alternatively, the same function words can be written separately such as اس میں (AS MYN, 'herein'), یہ وقت (YH OKT, 'this time') and ہو گی (HO GEE) (i.e two auxiliaries) respectively. These distinct forms of the same word(s) are visually correct and may be perceived as single or multiple words. These types of cases are ambiguous i.e. can be written with or without spaces and can be treated as a single unit or two different words. Consequently, this changes the perception of where the word boundary should sit. A possible solution to handle such words is to use a knowledge base. To conclude, the space insertion problem, space omission problem and ambiguity in tokenizing multi-words makes the Urdu word boundary detection

Table 1. Example text for various types of space omission problems

Type	Correct	Incorrect	Translation
Affixation	خوش ۔ اخلاق	خوشاخلاق	Polite
	KHOSH AKHLAK	KHOSHAKHLAK	
Abbreviations	ابن ۔ ابل ۔ ای	اناہلای	NLE
	AYN AYL AY	AYNAYLAY	
Compound word	تغیر ۔ پذیر	تغیرپذیر	Variable
	TGHYR PZYR	TGHYR PZYR	
English word	نٹ ۔ ورک	نٹورک	Network
	NYT ORK	NYTORK	
Proper noun	ویسٹ ۔ انڈیز	ویسٹانڈیز	West Indies
	OYST ANDYZ	OYSTANDYZ	
Reduplication	آنن ۔ فانن	آننفانن	Quickly
	AANN FANN	AANNFANN	

a complex and challenging task. This may be a possible explanation for the fact that no standard efficient Urdu word tokenizer is publicly available. An efficient Urdu word tokenization system would be needed to deal with these issues and to properly tokenize Urdu text.

To conclude, the space insertion problem, space omission problem and ambiguity in tokenizing multi-words makes the Urdu word boundary detection a complex and challenging task. This may be a possible explanation for the fact that no standard efficient Urdu word tokenizer is publicly available. An efficient Urdu word tokenization system would be needed to deal with these issues and to properly tokenize Urdu text.

3.2 Challenges for Sentence boundary detection

Sentence boundary detection is a non-trivial task for Urdu text because: (i) it does not use any special distinguishing characters between upper and lower case, (ii) punctuation markers are not always used as sentence separators, (iii) sentences are written without any punctuation markers, and (iv) there is a lack of standard evaluation and supporting resources. For English and other languages, the difference in upper and lower case is helpful in identifying sentence boundaries. Furthermore, in English language there is a convention that if a period is followed by a word starting with a capital letter then it is more likely to be a sentence marker, whereas, in Urdu, there are no upper and lower-case distinctions. Punctuation such as "-", ".", "؟" and "!"' are used as sentence terminators and these can also be used inside the sentence.

The Table 2 shows example Sentence Boundary Markers (SBM) (such as sen-

tences at index i, ii, iii, and iv, in all these sentences question, period, exclamation, and double quotes marker are used at the end of sentences to represent a sentence boundary) and Non-Sentence Boundary Markers (NSBM) for Urdu text. It can be observed from these examples that the NSBM are also frequent because they are being used between dates (such as sentence at index vii, in this sentence a period mark is used with in a sentence which is actually not a sentence boundary), abbreviations (index v, this sentence is composed of several period markers, however first two are not indicating a sentence boundary marker), emphatic declaration (index vi, here exclamation marker is used with in a sentence i.e. not a sentence boundary mark), names and range (index viii i.e. a first period and double quote marker is used within a sentence but both are not a sentence ending marker), and sentences without any SBM (index ix). Consequently, these kind of examples makes the sentence tokenization of Urdu text a challenging task.

3.3 Challenges for POS Tagging

POS tagging for the Urdu language is also challenging and difficult task due to four main problems (Naz et al., 2012; Mukund et al., 2010): (i) free word order (general word order is SOV), (ii) polysemous words, (iii) Urdu is highly inflected and morphologically rich, and (iv) the unavailability of gold-standard training/testing dataset(s). We briefly discuss these issues here.

Firstly, Urdu sentences have a relatively complex syntactic structure compared to English. Anwar et al. (2007b) have shown examples of the free word order and its semantic meaningfulness in the Urdu language. Secondly, as with other languages, Urdu also has many polysemous words, where a word changes it meaning according to its context. For example, the word باسی (BASY) means 'stale' if it is an adjective and 'resident' when it is a noun. Thirdly, Urdu is also a highly inflected and a morphologically rich language because gender, case, number and forms of verbs are expressed by the morphology (Hardie, 2003; Sajjad and Schmid, 2009). Moreover, Urdu language represents case with a separate character after the head noun of the noun phrase (Sajjad and Schmid, 2009). They are sometimes considered as postpositions in Urdu due to their place of occurrence and separate occurrence. If we consider them as case markers, then Urdu has accusative, dative, instrumental, genitive, locative, nominative, and ergative cases (Butt, 1995: Pg 10). Usually, a verb phrase contains a main verb, a light verb (which is used to describe the aspect) and a tense verb (describes the tense of the phrase) (Hardie, 2003; Sajjad and Schmid, 2009). Finally, there is a lack of benchmark training/testing datasets that can be used for the development and evaluation of Urdu POS taggers.

4 Urdu natural language toolkit

This study aims to develop a natural language toolkit for the Urdu language. The UNLT consists of word tokenization, sentence tokenization, and POS tagging modules. The following sections discuss these modules in more detail.

Table 2. Examples showing Sentence Boundary Markers (SBM) and Non-Sentence Boundary Markers (NSBM) for Urdu text

index	**Marker**[a]	**Text**
i	QM-SBM	مشرف کو باہر کیوں جانے دیا گیا ؟
		؟GYA DYA JANE KYON BAHR KO MSHRF
		Why was Musharraf let to go abroad?
ii	PM-SBM	انڈیا میں آئی سی سی ورلڈ ٹی ۲۰ کا آغاز ۔
		۔AAGHAZ KA 20 TY ORLD SY SY AAYY MY ANDYA
		Inauguration ceremony of ICC world T 20 held in India.
iii	EM-SBM	اس پر بھی عوام نہ سمجھے تو !
		! TO SMJHY NH AOAM BHY PR IS
		Even then if public do not understand then!
iv	DQ-SBM	« میرے خیال میں ان کے وزیر خارجہ ۲۱ اگست کو آرہے ہیں »
		"HYN RHY AA KO AGST 21 KHARJH OZYR KE AN MY KHYAL MYRE"
		"In my opinion the foreign minister is visiting on August 21st"
v	PM-NSBM	یو۔ ای ۔ اے میں کافی پاکستانی بستے ہیں ۔
		HYN BSTE PAKSTANY KAFY MY AY- AE -YO
		Many Pakistanis are living in U.A.E.
vi	EM-NSBM	حضور والا ! آپ پورے ملک کے بادشاہ ہیں ۔
		- HYN BADSHH KE MLK PORE AAP ! OALA HZOR
		My lord! You are the king of this country.
vii	PM-NSBM	آج ۳-۶-۲۰۱۵ ہے ۔
		- HYN 3-6-2016 AAJ
		Today is 3rd of May 2015.
viii	PM-NSBM	« »پاکستان» ۲ - ۳ سے جیت رہا ہے ۔
	DQ-NSBM	- HE RHA JET SE 3 - 2 "PAKSTAN"
		"Pakistan" is winning by 2-4.
ix	SBM	ابھی چند سال کا وقت لگے گا
	Missing	- ABHEY CHAND SAL KA WAQET LAGEY GA
		Still this will take few years

[a] QM: Question Mark, PM: Period Mark, EM: Exclamation Mark, DQ: Double Quotes

4.1 Urdu word tokenizer

4.1.1 Generating supporting resources for Urdu word tokenizer

For our proposed Urdu word tokenizer, we have developed two dictionaries: (i) a complex words dictionary - to address space insertion problem and (ii) a morpheme dictionary - to address the problem of space omission.

Complex words dictionary: To address the space insertion problem, a large complex words dictionary was created using the UMC Urdu dataset (Jawaid

et al., 2014), which contains data from various domains including Sports, Politics, Blogs, Education, Literature, Entertainment, Science, Technology, Commerce, Health, Law, Business, Showbiz, Fiction and Weather. From each domain, at least 1,000 sentences were randomly selected and pre-processed to remove noise (see Section 5.4). After noise removal, to speed up the dictionary creation process a basic space-based tokenization approach was implemented in Java to split sentences into words. Space based tokenization resulted in some incorrect word generation, e.g., complex words such as the prefix ان گنت (AN GNT'countless') is incorrectly split into a morpheme, ان (AN, literally 'this') and a stem, گنت (GNT, literally 'count'), postfix حملہ آور (HMLH AAOR, 'assailant') is incorrectly split as حملہ (HMLH, 'attack') i.e. a root and آور (AAOR, literally 'hour') i.e. a morpheme

Compound words which can be categorised into three types with respect to their formation: (i) *AB formation*– two roots and stems join together, (ii) *A-o-B formation*– two stems or roots are linked to each other with the help of و (wao) (a linking morpheme), and (iii) *A-e-B formation*– 'e' is the linking morpheme which shows relation between A and B. (for more detailed discussion see (Rehman et al., 2011)). In this research all three types have been used without any classification e.g. A-o-B formation type of compound word غور و فکر (GHOR O FKR, 'contemplation') is incorrectly split as غور (GHOR, literally 'ponder') a root, و (O) a linking morpheme, and a stem فکر (FKR, literally 'worry'). Reduplication which have two types: (i) *full reduplicated word*– two duplicate words are used to form a word and (ii) *echo reduplication*– the onset of the content word is replaced with another consonant (detailed information can be found in (Bögel et al., 2007)). Echo reduplication word, دن بدن (DN BDN, 'day by day') is incorrectly split as دن (DN, literally 'day') i.e. content word and بدن (BDN, literally 'body'), a consonant. One million space-based tokenized words list (henceforth UMC-Words) has been used to form a large complex words dictionary containing: (i) affixes, (ii) reduplications, (iii) proper nouns, (iv) English words, and (v) compound words.

To collect affixes (prefixes and postfixes) complex words from the UMC-Words list (Jawaid et al., 2014), a two-step approach is used. In the first step, a list of prefixes and postfixes are manually generated. In the second step, an automatic routine is used to extract words containing affixes from the large UMC-Words list. Using prefixes and postfixes, the previous and next words are extracted respectively from the UMC-Words list.

Reduplications complex words are collected using two methods: (i) full extraction and (ii) partial extraction. The full extraction method is used to extract the full reduplicated words such as جسي جسي (JYSY JYSY, 'as'). To extract such full reduplicated words, we compared each word in the UMC-Words list to the next word, if both are the same then concatenate both to form a full reduplicated compound word. The partial extraction method is used to collect the words of echo reduplication i.e. in which a consonant word is a single edit distance away from the

first content word. The echo reduplication words can be further collected using two methods: (i) one insertion extraction and (ii) single substitution extraction.

One insertion extraction method extracts the one insertion reduplicated words, in which the consonant word has one insertion in its content word e.g. دن بدن (DN BDN, 'day by day'). It can be noted that the consonant word بدن (BDN, literally meaning 'body') has one more character (three) as compared to the content word دن (DN, literally 'day') (which have two characters). Furthermore, the last two characters of the consonant word are identical to the content word. To extract one insertion reduplicated words, we used the UMC-Words list. The extraction process works as follows: after excluding the first character, if the remaining characters of consonant word are identical as well as having the same character count to the content word, they are one insertion reduplicated word(s) we concatenated them to form a single word.

The single substitution extraction method extracts the single substituted reduplicated word(s) - here the consonant word has single substitution in its content word e.g. خلط ملط (KHLT MLT, 'intermixed'). It is worth noting that both words content خلط (KHLT, literally 'bad') and consonant ملط (transliteration: MLT) has three characters and the final two characters are overlapping. To extract one substituted reduplicated word(s) we used automatic routine and applied the following process over the UMC-Words list as: if the length of the content word is matched with the length of the consonant word and the length of content word is greater than two[16] characters, and if one character is dissimilar after comparing character by character, then it will form a single substitution reduplicated complex word.

To automatically extract abbreviations (91) and proper nouns (2K), regular expressions are used and further supplemented by manual checking to increase the size of the proper nouns (3K) and abbreviations lists (187). The remaining 65K proper noun list was generated in another NLP project and are used in this study for Urdu word tokenization. In addition to this, manual work[17] was also carried out to remove noisy affix entries. Moreover, compound words (of formation AB and A-e-B) and English words are added to increase the size of the complex words dictionary. However, to collect words of A-o-B formation automatically, a linking morpheme (و, O) has been used. While using a linking morpheme both previous and next words are extracted from the UMC-Words list to form a A-o-B compound words.

The complete statistics of the complex words dictionary are as follows: there are

[16] To make sure the two character words or auxiliaries could not be erroneously identified as reduplication such as كر كي (KR KE, literally 'by doing')

[17] Five undergraduate NLP students have been employed to carry out manual tasks, all are native Urdu speakers and have an interest in Urdu NLP and literature. Furthermore, each student undertook a practical training session on annotation tasks. Each student was given an annotation assignment of 80 random sentences from the UPC dataset and requested to extract affixes, compound words, abbreviations and English words. These assignments were marked and each student was awarded with a score. Students having scored 85% or above were thus selected for annotation tasks.

in total 80,278 complex words (7,820 are affixes, 278 are abbreviations, 10,000 are MWEs, 1,480 are English words, 60,000 are proper nouns and 700 are reduplication words).

Morpheme segmentation process: To address the space omission problem (see Section 4.1.2), a large-scale morpheme dictionary is automatically compiled from the HC dataset (Christensen, 2014). Before we proceed further towards the approach used to generate the morphemes dictionary, it is worth describing the morpheme types. Urdu language morphemes can be categorized into: (i) free and (ii) bound morphemes. Proposed word tokenizer has to rely on both categories. The bound or functional morphemes such as affixes include prefixes, e.g., "کو ، لا ، گا" (GA, LA, KO), linking morphemes, for e.g., ا ،و (A, O) or suffixes, for e.g., شدہ ، زدہ (transliteration: SHDH, ZDH), can only expose their meanings if they are attached to other words, i.e. they cannot stand alone. Whereas, free or lexical morphemes can stand alone, for example, غم ، علم ،چست ، مقبول (MKBOL, CHST, ALM, GHM, 'grief, knowledge, clever, famous').

There are two further categories of free morphemes: (i) true free morphemes and (ii) pseudo-free morphemes. True free morphemes can be either standalone (for e.g., دل (DL, 'heart')) or form part of other words (e.g. درد دل (DRD DL, 'angina pectoris')). Pseudo-free morphemes can be a character, affix or word.

The preceding discussion summarizes the various types of morphemes. However, from a computational linguistics view, free and bound morphemes play a vital role in Urdu word formations (Khan et al., 2012), hence, they will be used without any further classification in our proposed UNLT word tokenizer module.

In order to generate the morpheme dictionary, the 1,000 most frequent words which have more than 20 occurrences in the HC dataset are used (Christensen, 2014); the selected words were split to form a morpheme dictionary. The whole chopping process is completed in two steps: (i) Crude-Morphemes (CM) chopping and (ii) Ultra-Crude-Morphemes (UCM) chopping.

In the first step, the first n character(s) of each word are kept while the rest are discarded. For example, in case of $n = 1$, we kept only the first character and discarded all others, thus words such as واقفیت (OAKFYT, 'awareness') will return و (wao). Such single character morphemes are helpful to formulate compound words, for instance خوش و خرم (KHSH O KHRM, 'canty'). Furthermore, we keep chopping all the words repeatedly with the following values of $n = 2, 3.4, 5, 6$[18]. This process returns وا، واق، واقف، واقفی، واقفیت (transliterations are: OA, OAK, OAKF, OAKFY, OAKFYT) morphemes for the word واقفیت (OAKFYT, 'awareness'). There may be a situation where we may lose several valuable morpheme(s), if the length of $n > 6$. Nevertheless, this is a rare case. Henceforth, we will call this method Crude-Morpheme chopping.

[18] An assumption made by us after analysing Urdu text that a word is formed of a maximum of six morphemes

To generate entirely different morphemes from the same word, we further applied a modified version of CM chopping, i.e. UCM. In which, we skipped the first character and then applied the CM chopping with length $n = 2, 3, 4$. Thus, UCM chopping resulted with these morphemes, اق، اقف، اقفی، اقفت (transliterations are: AK, AKF, AKFY, AKFYT) for the word واقفت (OAKFYT, 'awareness'). Furthermore, we iterate the UCM chopping method by skipping the first two characters (as well as three, four etc.), until we meet the last two characters. Thus, the following morphemes are returned by UCM, in the third قف، قفی، قفت (transliterations are: KF, KFY, KFYT), in the fourth فی، فت (transliterations are: FY, FYT) and in the last ت (transliteration, YT) iterations.

Repeating CM and UCM chopping on the entire list of words will return all possible morphemes. The two chopping methods used in this study will result in erroneous morphemes. However, we manually examined the morpheme dictionary and removed these. The number of morphemes generated by the CM and UCM chopping methods were 5,089 and 7,376 respectively.

It can be observed from the above discussion that two different large-scale dictionaries i.e. the complex words dictionary and the morphemes dictionary are generated with distinct approaches and with various statistics. These dictionaries will be used to solve the space omission and space insertion problems with the word tokenizer module of UNLT. To the best of our knowledge, no such large complex words (a study (Hautli and Sulger, 2009) just proposed a scheme to extract location and person name) and morpheme dictionaries have been previously compiled semi-automatically for Urdu, to perform Urdu word tokenization.

4.1.2 Proposed Urdu word tokenizer

To investigate an effective approach for UNLT Word Tokenization (henceforth UNLT-WT approach), our method (see Algorithm 1) is a combination of state-of-the-art approaches: rule-based maximum matching, dictionary lookup, statistical *tri*-gram Maximum Likelihood Estimation (MLE) with back-off to *bi*-gram MLE. Furthermore, smoothing is applied to avoid data sparseness. A step by step working example of the proposed algorithm can be seen at[19]. However, this section just presents the statistical approach used to solve space omission problem.

Maximum likelihood and smoothing estimation: In our proposed UNLT-WT approach (see Algorithm 1) at step 11.1, we used *tri*-gram MLE and smoothing estimations, because there can be multiple tokenized sequences for which *flag_bit=false* and *word_count* are equal. For instance, there are two given texts, (i) اسي باہر جا کي پڑھني دو (transliteration: ASE BAHR *JA* KE PRHNE DO, 'let him *go* abroad for higher studies'), and (ii) اسي باہر جی کي پڑھني دو (transliteration: ASE BAHR *JY* KE PRHNE DO, literally meaning 'let him *yes* abroad for higher

[19] https://doi.org/10.17635/lancaster/thesis/831- Page 88 - 98

Algorithm 1 UNLT-WT approach

Step 1: Initialize flag_bit=false, row=1, column=1, word_counter=0;
Step 2: Create array words_list[row][column], array morphemes_list, array compound_words_list;
Step 3: Remove all white spaces and ZWNJ, to form a space free input text.
Step 4: Read *bi*-gram of input text.
Step 5: Match this *bi*-gram with each word of *morphemes_list*
Step 6: Extract all those morphemes from *morphemes_list*, which matched with *bi*-gram.
Step 7: Store each extracted morpheme on a separate row/column of *words_list*
Step 7.1: For each row, copy the flag_bit, word_counter++
Step 8: If no match is found in *morphemes_list*, split *bi*-gram into *uni*-gram.
Step 8.1: Store the first *uni*-gram with previous morpheme (column) except و (character O) and ا (character A) (use in compound words) and turn the flag_bit=true. For و and ا, store it on separate column of array*words_list*[row][column] and increment word_counter.
Step 9: Repeat the steps 4 to 8, until sentence ending marker, and for each row of *words_list*.
Step 10: Select the row having minimum *word_counter* value and flag_ bit=false.
Step 11: If multiple rows are qualified in step 10 then
Step 11.1: Calculate *tri*-gram MLE for each row.
Step 11.1.1: Select the one having highest value of *tri*-gram MLE.
Step 11.2: If in step 11.1.1, any row having *tri*-gram MLE value equal to *Zero*, then calculate *bi*-gram MLE for each row.
Step 11.2.1: Select the one having highest value of *bi*-gram MLE.
Step 11.3: If in step 11.2.1, any row having *bi*-gram MLE value equal to *Zero* then, calculate *bi*-gram *smoothing* for each row.
Step 11.3.1: Select the one having highest value of smoothing.
Step 12: For final selected row, read each column and match in the compound word dictionary.
Step 12.1: If a match is found then read the next column of selected row in step 12 and repeat step 12 for the remaining part of selected row.
Step 12.1.1: If complete match is found then concatenate with the columns in step 12.1.
Step 12.1.2: Move each element of final selected row in step 12, decrease the array index.
Step 13: Finally, list of tokenized word will be produced.

studies'). Both have six tokens with flag_bit= false, but only the first text is semantically correct and meaningful. For such ambiguous cases, we calculate an N-gram language model with MLE for parameter and Laplace for smoothing estimation. The goal of these estimations is to find an optimized segmented sequence with the highest probability. This can be shown by a given mathematical expression, a general statistical model of our proposed UNLT-WT approach.

$$\hat{P}(m_j|m_{1-N+1}m_{n-N+2}...m_{j-1}) = \underset{M\in\varsigma(I|D)}{arg\ max} P(M) \tag{1}$$

Here, $\varsigma(I|D)$ denotes all possible tokenized words of the input string i.e. $I = i_1 i_2 ... i_l$ with l characters, and M denotes string concatenation of all possible tokenized sequences i.e. $M = m_1 m_2 ... m_n$, in terms of morphemes dictionary D. Theoretically, it is assumed that the n-gram model outperforms with a high value of N. However, practically the data sparseness restricts better performance with high order N. Therefore, in our UNLT-WT approach, we opted for *tri*-gram ($N = 3$) or *bi*-gram ($N = 2$) MLE. These have proved to be successful in several tasks for resolving

ambiguity (e.g. POS tagging (Brants, 2000), automatic speech recognition (Abdelhamid et al., 2012) and word tokenization (Fu et al., 2008)).

The task of resolving similar sequence ambiguities for the above two texts is accomplished by using *tri*-gram MLE (Jurafsky and Martin, 2014) as:

$$P\left(t_j|t_{j-2},t_{j-1}\right)=\frac{C\left(t_{j-2},t_{j-1},t\right)}{C\left(t_{j-2},t_{j-1}\right)} \tag{2}$$

Where t represents the individual token, C is a count of three ($t_{j-2}t_{j-1}t$) and two ($t_{j-2}t_{j-1}$) consecutive words in the dataset and P is the *tri*-gram contestant MLE value of each of the possible segmented sequences. The calculated probability for the first sequence is 3.2e-08 while for the second it is 0.

As *tri*-grams take account of more context, if this specific context is not found in the training data (see Section 5.1), we back-off to a narrower contextual *bi*-gram language model. *Bi*-gram cumulative probability values have been calculated as given by Jurafsky and Martin (2014):

$$P\left(t_j|t_{j-1}\right)=\frac{C\left(t_{j-1}t\right)}{C\left(t_{j-1}\right)} \tag{3}$$

Where t represents the individual token, C is a count of two ($t_{j-1}t$) and one (t_{j-1}) consecutive word(s) in the dataset and P is the *bi*-gram contestant MLE value of each of the possible segmented sequences. The calculated probability for the first sequence is 2.7e-6 for the former sequence and 0 for the latter one.

These *zero* probabilities are again an underestimation of the input string, ultimately a cause for the data sparseness. Even if a statistical language model is trained on a very large dataset, it will remain sparse in some cases. However, there is always a possibility that the input text occurs in the test dataset (Chen and Goodman, 1999), thus assigning them to zero made this an unstable, frail and specific estimator. Therefore, to overcome this, different *smoothing* techniques have been proposed in previous literature (Jurafsky and Martin, 2014) with different characteristics (such as smoothing the probability etc.). Hence, it is primarily aimed at making a robust and generalize language model by re-evaluating lower or zero probability upwards and vice-versa for high probabilities.

For this study, we employed Laplace (add-one) smoothing (Jeffreys, 1998), as one of the oldest, simplest and baseline estimations. This estimation adds one to all frequency counts, i.e. that all *bi*-gram probability counts have been seen one more time than actually exists in the training data as:

$$P_{add:1}\left(t_j|t_{j-1}\right)=\frac{1+C\left(t_{j-1},t\right)}{V+C\left(t_{j-1}\right)} \tag{4}$$

Where v represents the unique words (types), added to the total number of words $C(t_{j-1})$ in order to keep the probability normalized (Jurafsky and Martin, 2014). We have used Laplace smoothing to estimate the parameters required for data sparseness in order to increase the *bi*-gram MLE value for اسي باہر جی کي پڑھني دو (transliteration: ASE BAHR *JY* KE PRHNE DO, 'let him *go* abroad for higher studies'), from 0 to 1.9e-14, and decreased value for اسي باہر جا کي پڑھني دو (transliteration: ASE BAHR *JA* KE PRHNE DO, literally meaning 'let him *yes*

abroad for higher studies'), from 2.7e-6 to 3.8e-7. As the latter tokenized sequence has the highest smoothing MLE. Therefore, it will be selected by UNLT-WT as the best tokenized sequence, which is correct.

4.2 Urdu sentence tokenizer

4.2.1 Rule based approach

For our proposed rule-based approach, to manually extract rules for the sentence tokenization task, initially, a subset of the UMC dataset (Jawaid et al., 2014) comprised of 13K sentences is selected, which contains Urdu text from various domains or genres including News, Religion, Blogs, Literature, Science and Education. After pre-processing (see Section 5.4) we retained 10K sentences, which were used to extract rules to develop our proposed Urdu sentence tokenizer. The rules were devised to include sentence termination markers (۔, ؟, ؞ and !), regular expressions and supplementary dictionary lookup[20] (henceforth UNLT-ST-RB approach). These heuristics are applied as follows:

1. If the current character is a period marker (۔) *AND* the same mark appears after two or three characters, then consider it as an abbreviation and match it in the abbreviation list.
2. If within the next 9 characters (from any previous SBM marker), an exclamation mark (!) is found, then this is not a sentence boundary marker.
3. If the character before a double quote (؞) is a period (۔) or question (؟) mark, then it is a sentence boundary marker.
4. Apply regular expressions for detecting the date and hyphenated numeric values.
5. In addition to this all the above rules from 1 to 4, split sentences based on the question (؟), period (۔) and exclamation (!) markers.

4.2.2 ML based approach

In this approach, we are exploiting a Support Vector Machines (SVM) classifier (Hearst et al., 1998) to detect the sentence boundaries of the Urdu text - using the features described below another approach is formed i.e. UNLT-ST-ML. SVMs offer robust classification even with sparse vectors of large dimension (Akita et al., 2006), its good performance results on textual data and its suitability for binary classification (Kreuzthaler et al., 2015) task make this a suitable classifier for sentence boundary detection. Moreover, SVMs use a function (see Equation 5) for classifying sentence boundary label pairs $(x_j, y_j), j = 1, ..., m$ for all $x_j \in \mathbb{R}^n$ to a target value $y \in \{1, -1\}$. Where $w \in \mathbb{R}^n$ a weight coefficient and $b \in \mathbb{R}$ is a bias. We are using a Polynomial kernel implemented in Weka[21].

[20] We used the same dictionary compiled for the word tokenization task (see Section 4.1.1)
[21] `http://www.cs.waikato.ac.nz/ml/weka/` - last checked: 22-September-2020

$$f(X) = sgn(w^T \phi(x) + b) \tag{5}$$

Features for ML approach:

- Probability (UMC dataset[22] (Jawaid et al., 2014) is used) that a word with "؟, ۔ and !" occurs at the end of a sentence
- Probability (UMC dataset[23] is used (Jawaid et al., 2014)) that a word with "؟, ۔ and !" occurs at the beginning of a sentence
- Length of a word with "؟, ۔ and !"
- Length of a word after "؟, ۔ and !"
- Is a sentence contains an abbreviation
- Is a sentence contains a date/numeration
- *Bi-* and *tri-*grams words information (preceding "؟, ۔ and !") are used
- If a word before "؟, ۔ and !" markers contains any one of the tag (NN, NNP, JJ, SC, PDM, PRS, CD, OD, FR, Q, and CC. See Section 4.3.1 for POS tags) is not a sentence boundary

4.3 Urdu part of speech tagging

4.3.1 Existing Urdu POS tagset

The tagging accuracy of a POS tagger is not only dependent on the quality and amount of training dataset but also on the POS tagset used for annotation. In the prior literature, we found three commonly used POS tagsets for the Urdu language: (i) Hardie's POS tagset (Hardie, 2004), (ii) Sajjad's POS tagset (Sajjad, 2007) and (iii) Centre for Language Engineering (CLE) Urdu POS tagset (Ahmed et al., 2014).

Hardie's POS tagset (Hardie, 2004) was an early attempt to resolve the grammatical tag disambiguation problem for the Urdu language. This tagset follows the EAGLES[24] guidelines and consists of 350 morphosyntatic tags, which are divided into 13 main categories. Some grammarians (Platts, 1909) propose only three main categories whereas (Schmidt, 1999) used 10 main categories for Urdu text. There were a number of shortcomings observed in Hardie's POS tagset (Hardie, 2004). For example, the possessive pronouns like میرا (MYRA 'my'), تمہارا (TMHARA 'your') and ہمارا (HMARA 'our') are assigned to the category of *possessive adjective*, which is incorrect. Many grammarians marked them as *pronouns* (Platts, 1909; Javed, 1985). Moreover, the Urdu language has no *articles* but this tagset defined articles. Another issue with the tagset is the use of *locative* and *temporal adverbs* such as یہاں (YHAN 'here'), وہاں (OHAN 'there'), and اب (AB 'now'), which are treated as *pronouns*. The *locative* and *temporal* nouns such as صبح (SBH 'morning'), شام

[22] To calculate that a certain word occurs before a sentence boundary
[23] To calculate that a certain word occurs after a sentence boundary
[24] `http://www.ilc.cnr.it/EAGLES96/home.html` - Last checked: 07-December-2016

(SHAM 'evening'), and گھر (GHR 'home') appear in a very similar syntactic context. To conclude, these grammatical misclassifications as well as the large number of POS tags with relatively small training data will affect the accuracy of POS taggers developed for the Urdu language.

Another POS tagset (henceforth Sajjad's POS tagset) (Sajjad and Schmid, 2009), consists of 42 POS tags with finer grained categories for pronouns and demonstratives. However, it is lacking in terms of Urdu verb, tense and aspect.

A recently released CLE Urdu POS tagset (Ahmed et al., 2014) contains 35 tags and addresses most of the issues reported above. It is based on the critical analysis of several previous iterations of Urdu POS tagsets. Furthermore, it is built on the guidelines of the Penn Treebank[25] and a POS tagset for common Indian languages[26]. In the CLE Urdu POS tagset, a verb category has multiple tags based on the morphology of the verbs. Furthermore, it has shown promising results on Urdu text see Section2.3).

For this study, we selected the CLE Urdu POS tagset (Ahmed et al., 2014) for the following reasons: (i) it provides correct grammatical classifications, (ii) it provides purely syntactic categories for major word classes and (iii) provides reasonable performance on a small size test dataset.

4.3.2 Proposed Urdu POS tagging approaches

For this study, we applied two stochastic approaches for Urdu POS tagging: (i) *tri*-gram Hidden Markov Model and (ii) Maximum Entropy-based model. The reason for selecting these two methods for Urdu POS tagging is many fold, (a) they have proven to be effective for POS tagging not just for English (Yi, 2015) but also for other languages which are closely related to Urdu such as Hindi (Joshi et al., 2013; Dandapat, 2008), (b) both are well established stochastic models for automatic POS tagging task (Wicaksono and Purwarianti, 2010), (c) these methods have been primarily investigated for when dealing with languages with limited resources (Azimizadehet et al., 2008; Ekbal et al., 2008), and (d) these models have not been previously compared for the Urdu language.

Hidden Markov Model (HMM) for POS tagging: In general, the Urdu POS tagging task can be formulated as: given a sequence of words $w_1, ..., w_n$, find the sequence of POS tags $t_1, ..., t_n$ from a POS tagset T[27] using some statistical model. In this section we have used HMM stochastic learning model described by (Rabiner, 1989), while Thede and Harper (1999) redefined it for the POS disambiguation task. This model was implemented in Garside and Smith (1997); Bird et al. (2009) for POS tagging. For our experiments, we used a third order HMM learning model, also referred to as a *tri*-gram POS tagging. This model is composed

[25] `https://www.cis.upenn.edu/treebank/` - Last checked: 13-June-2017

[26] `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/I08-7013.pdf` - Last checked: 24-September-2016

[27] 35 tags as in CLE Urdu POS tagset: `http://www.cle.org.pk/software/langproc/POStagset.htm` - Last checked: 20-November-2016

of transitional (contextual) and lexical (emission) probabilities as:

$$\hat{T} = \underset{t \in T}{arg\ max} P(t_1, ..., t_n) | (w_1, ..., w_n) \tag{6}$$

Using Bayes' theorem, the above equation can be rewritten as for 3^{rd} order model as:

$$P(t_1, ..., t_n | w_1, ..., w_n) = \underset{t_1, ..., t_n}{arg\ max} \prod_{j=3}^{n} \underbrace{(P(t_j | t_{j-1}, t_{j-2})}_{Transition Probability} * \prod_{i=1}^{n} \underbrace{P(w_i | t_i))}_{Lexical Probability} \tag{7}$$

During the training process, the above *tri*-gram HMM language model computes two probability factors for the sequences: (i) emission probabilities, aimed at determining the probability of a particular tag conditioned on particular word, and (ii) transitional probabilities, used to find the probability of a particular tag on the basis of given preceding tag(s). Given a sentence, the aim of the HMM language model is to search the tagging sequence and choose the most likely sequence that maximises the dot product of lexical and transition probabilities. That can be computed by using a Viterbi algorithm (Viterbi, 1967).

Parameter estimation: We can estimate the HMM parameters by applying the simplest *tri*-gram MLE (see Section 4.1.2), used for computing relative frequencies. We have used a training dataset (see Section 5.3) to find tag frequency counts (C) for two or three consecutive tag pairs (t_{j-2}, t_{j-1}, t_j), (t_{j-2}, t_{j-1}). Where, t_j is the j_{th} tag of annotated dataset used during training process. The following equation requires frequencies count of $w_i t_i$, where w_i is the word and t_i is the tag assigned to i_{th} word. The *tri*-gram language model (see Section 4.1.2) and the following equation is used with these parameter settings, $1 \leq (i, j) \leq n$.

$$P(w_i | t_i) = \frac{C(w_i t_i)}{C(t_i)} \tag{8}$$

Smoothing: We have used the MLE for parameter estimation (see Section 4.3.2), consequently, such models may come across a situation where unseen events do not occur or have quite low frequencies in the trained model. Therefore, the zero probability of such occurrences produces problems in the multiplication of probabilities, eventually, leading to a data sparseness.

To avoid data sparseness, we need some estimators that automatically assign a part of the probability mass to unknown words and tag sequences, thus yielding an improvement for unseen events and overall accuracy improvement for the POS tagger. For this, different smoothing techniques have been cited in the literature with an objective to decrease the probability of seen events and assigning appropriate non-zero probability mass to unseen events. In this study, five different smoothing techniques were adopted including: (i) linear interpolation, (ii) Laplace, (iii) Lidstone's, (iv) Good-Turing, and (v) Kneser-Ney estimations. Adopting them with an HMM model thus alleviates sparse data issues.

Linear interpolation: A well-practised smoothing technique consists of linearly combined estimation for different order n-grams as:

$$P(t_i | t_{i-1}, t_{i-2}) = \lambda_1 \rho(t_i) + \lambda_2 \rho(t_i | t_{i-1}) + \lambda_3 \rho(t_i | t_{i-1}, t_{i-2}) \tag{9}$$

Where P is a valid probability distribution, ρ are maximum likelihood estimates of

the probabilities and $\lambda_1 + \lambda_2 + \lambda_3 = 1$ to normalise the probability. Although, there are different ways to estimate λs, for our experiments we adopted *deleted linear interpolation*, cited in (Brants, 2000).

The deleted linear interpolation successively removes each *tri*-gram from the training dataset. Moreover, this technique estimates the best value for the λs from all other n-grams in the dataset, making sure that the value of λ does not depend upon the particular n-gram. Further, it computes the weights depending on the counts of each i-gram, involved in the interpolation. Thus, our first HMM based proposed model is a combination of linear interpolation smoothing technique along with *tri*-gram HMM model (henceforth T-HMM-LI).

Laplace and Lidstone's estimation: Laplace estimation (one of the oldest and simplest smoothing techniques) updates the count by one of each *bi*-gram occurs compared to the actual frequency in training data (Jurafsky and Martin, 2014) (see Section 4.1.2). Whereas, Lidstone's smoothing estimation (Manning and Schütze, 1999) generalizes Laplace, by adding an arbitrary value to all (seen or unseen) the events. Although the values for λ can be calculated using different methods, for our experiments we used the same value cited in the research article (Manning and Schütze, 1999), i.e. a well-known Expected Likelihood Estimation (ELE). Thus, Lidstone's estimation (Manning and Schütze, 1999) can be calculated as:

$$P_{lidstone(x,\lambda)} = \frac{\lambda + C(X)}{V\lambda + N} \qquad \lambda = 0.5 \tag{10}$$

Where V represents the unique words (vocabulary) against the total number of words N to keep probabilities normalized (Jurafsky and Martin, 2014). The generalized formulation of Lidstone's and Laplace estimation in an HMM-based Urdu tagger is as follow:

$$\pi_i = \frac{C(s_i(t=0)) + \lambda}{C(tokens) + V_{tag}\lambda} \tag{11}$$

$$a_{ij} = \frac{C(s_i \rightarrow s_j) + \lambda}{C(tokens) + V_{tag}\lambda} \tag{12}$$

$$P(s_j) = \frac{C(s_j) + \lambda}{C(tokens) + V_{tag}\lambda} \tag{13}$$

$$P(w_k) = \frac{C(w_k) + \lambda}{C(tokens) + V_w\lambda} \tag{14}$$

Here, V_{tag} is the number of possible tags and V_w is the size of the approximated vocabulary. Our second POS tagging model is a combination of Laplace and *tri*-gram HMM model (henceforth T-HMM-LaE). The third POS tagger makes use of Lidstone's estimation and supervised *tri*-gram HMM model parameters (we shall call this T-HMM-LiE).

Good-Turing estimation: Uses the probability mass of n-grams (that occur $c+1$ times) which is seen once to re-estimate the count of n-grams (that are seen exactly c times) that were never seen. This can be described as:

$$c^* = (c+1)\frac{n_{c+1}}{n_c} \tag{15}$$

The fourth POS tagger makes use of Good-Turing estimation and supervised *tri*-gram HMM model parameters (hereafter, T-HMM-GT).

Kneser-Ney smoothing: This outperforms all other smoothing techniques. In this paper we have used the modified version of this smoothing (Christer, 1996), which is an interpolated variation of Kneser–Ney smoothing with an augmented version of absolute discounting, thus the transition probabilities $p(t_j|t_{j-1})$ are calculated as:

$$P(t_j|t_{j-1}) = \frac{f(t_{j-1}t_j) - D(f(t_{j-1}t_j))}{\sum_{t_j} f(t_{j-1}t_j)} + \gamma(t_{j-1}).\frac{N_{1+}(\cdot, t_i)}{N_{1+}(\cdot\cdot)} \tag{16}$$

Here D is estimated value and $D(f) = \{0 \text{ if } f = 0,\ D_1 \text{ if } f = 1, D_2 \text{ if } f = 2,\}$. Another POS tagger is formed using supervised *tri*-gram HMM model and and Kneser-Ney smoothing (henceforth T-HMM-KN).

Maximum Entropy (MaEn) model for POS tagging: The other adopted stochastic learning model is MaEn, and we aimed to compare this to the above described *tri*-gram HMM based models, to find the most optimal POS tagger for Urdu. The MaEn statistical assumption is a simplistic model, it assigns a probability distribution for every tag, given a word and its context as:

$$P(t_1, ..., t_n|w_1, ..., w_n) = \prod_{j=1}^{n} P(t_j|c_j) \tag{17}$$

Where, t is the individual tag in the set T of all possible tags i.e. $t_1, ..., t_n$ for a given a sentence, c is defined as the context, usually defined as the sequence of words $w_1, ..., w_n$ and the tag preceding the word. The maximum likelihood tag sequence is used for assigning probabilities to a string of input words.

The principle of estimating probabilities in MaEn model is to make as few assumptions as possible, other than the constraint imposed. Furthermore, these constraints are learned from the training data, which express some relation between features extracted and outcome. The probability distribution which satisfies the above property has the highest entropy, thus, it agrees with the maximum likelihood distribution, and has a general form as cited in (Ratnaparkhi, 1996):

$$P(t|c) = \frac{1}{N} exp \sum_{j=1}^{k} \alpha_j f_j(c, t) \tag{18}$$

Where N is the total number of training samples (normalization constant), f_j is feature function on the event (c, t). Feature functions used by MaEn model are binary valued and defined to capture relevant aspects of language. The α_j is a model parameter with k features, which is determined through the Generalize Iterative Scaling (GIS) algorithm (Curran and Clark, 2003). However, these model values and features, are primary ingredients of MaEn learning model.

Features selection in MaEn model: As described previously the MaEn is feature based probabilistic model, to obtain high accuracy we used two binary valued features that might be helpful for predicting POS tag, these are determined empirically for Urdu POS tagging along with MaEn model as: (i) context window, and (ii) word number. The best context window with five words has been identified, which is comprised of n-gram (W_{i-2}, W_{i-1}, W_i, W_{i+1}, W_{i+2}) and n-POS (t_{i-2}, t_{i-1},

and t_i) information. If the current word is a number such as "۲۵۰۳۱", another feature can be created:

$$f_j(c,t) = \begin{Bmatrix} 1 \; if \; WordReadIsNumber \; (w_j) = true \; and \; t_j = CD \\ else \; 0 \end{Bmatrix} \quad (19)$$

Using the above mentioned features with MaEn we formulated another Urdu POS tagging model (henceforth MEn). However, these suitable binary valued features are the same for other languages. We examine some other important feature sets for the Urdu language below.

Morphological information for HMM and MaEn models: To improve the tagging accuracy of the unknown words in the above models, we have developed an exclusive feature set after detailed analysis of the UNLT-POS training dataset (see Section 5.3). This feature set is intended to have the capability to capture lexical and morphological characteristics (features) of the Urdu language. The captured morphological features are based on information retrieved from a stemmer[28] and dictionary[29], assuming that information is complete[30]. Thus, we boosted the lexical probability of assigning restricted lexical (POS) tag to a word. Consequently, the integrated models are expected to perform better with such artificial weight (reduced set of possibilities) for a given word. All the above models (T-HMM-LI, T-HMM-LaE, T-HMM-LiE, T-HMM-GT-MA, T-HMM-KN, and MEn) are incorporated with such restricted POS tags features, henceforth, T-HMM-LI-MA, T-HMM-LaE-MA, T-HMM-LIE-MA, T-HMM-GT-MA, T-HMM-KN-MA and MEn-MA.

The above mentioned MA information is helpful to restrict the possible choice of POS tags for a given word, on the other hand, suffix and prefix (of current word) information can also help us to further improve the POS models. For HMM based POS models, suffix information has been used during the smoothing of emission probabilities. Whereas, for the MEn model the suffix and prefix information are used as another type of feature. It is extended using a prefix and suffix up to a length of four. It is also important to note, using prefix and suffixes of length $<= 4$ for all words in MEn gives better results instead of using only rare words as described by Ratnaparkhi (1996). The primary reason for much improved results based on prefix and suffix is that, a significant number of instances are not found for most of the word of the language vocabulary, with a small amount of annotated data. HMM based (T-HMM-LI, T-HMM-LaE, T-HMM-LiE, T-HMM-GT, and T-HMM-KN) and MEn models are incorporated with suffix information, we shall call them T-HMM-LI-Suf, T-HMM-LaE-Suf, T-HMM-LiE-Suf, T-HMM-GT-Suf, T-HMM-KN-Suf, and MEn-Suf POS taggers.

The last six POS models represent combinations of various statistical, smoothing and features as described above. The T-HMM-LI-Suf-MA is a combination of

[28] `http://www.cle.org.pk/software/langproc/UrduStemmer.htm` - Last checked: 17-November-2016

[29] `http://182.180.102.251:8081/oud/default.aspx` - Last checked: 09-October-2016

[30] If a word is unknown then it belongs to one of the open class lexical categories, i.e. all classes of Noun, Adjective, Verb, Adverb, and Interjection.

tri-gram HMM along with Linear interpolation, restricted POS tags feature and suffix information. T-HMM-LaE-Suf-MA is based on the *tri*-gram HMM model with further incorporation of Laplace smoothing, suffix and restricted POS tags. In T-HMM-LiE-Suf-MA, we have used *tri*-gram HMM along with Lidstone's estimation, with suffix and restricted POS tags. The T-HMM-GT-Suf-MA is a combination of *tri*-gram HMM along with Good-Turing, restricted POS tags and suffix information. In T-HMM-KN-Suf-MA, *tri*-gram HMM, Kneser-Ney estimation, suffix and restricted POS tags are used. MEn-Suf-MA POS tagging model is a collection of, MaEn, contextual window, suffix and restricted POS tags.

4.4 Deep learning approaches for comparison

We have performed an empirical comparison of our proposed word/sentence tokenizers and POS tagger, with the recently proposed Trankit NLP Toolkit (Nguyen et al., 2021). We have selected this deep learning toolkit for the evaluation process as this has reported good results as compared to other deep learning methods. This is a light-weight transformers-based (a deep neural network based new SOTA architecture in NLP (Vaswani et al., 2017)) tool for multi/mono-lingual text processing, with trainable and pre-trained pipelines for NLP tasks for 100 languages including Urdu. Trankit's recent version[31] NLP tools are built upon the transformer XLM-RoBERTa (Conneau et al., 2020) (a cross-lingual transformers based masked language model, trained on more than 2 terabytes of pre-processed Common Crawl data for 100 languages), extends the state-of-the-art in word/sentence tokenization, dependency parsing, part-of-speech tagging, and tagging of morphological features. It also attained high performance in various other basic NLP tasks including word tokenization, multi-word expansion, and lemmatization of the 90 treebanks.

Trankit's word and sentence tokenizers are based on word parts rather than character-based to exploit the contextual information (Kudo, 2018). It begins on a given input text t, by splitting it into sub-strings on the basis of space. Furthermore, a multi-lingual sentence splitter (Kudo and Richardson, 2018) further breaks each sub-string into word pieces, concatenated to obtain an overall sequence of word-parts $w = [w_1, w_2, w_3, ..., w_n]$ for t. In the next step, a pre-trained transformer takes w as input to produce corresponding representation vectors $v = [v_1, v_2, v_3, ..., v_n]$ for each word-parts in the sequence w. Each vector from v will be fed to a feed forward neural network with a *softmax* at the end to predict whether the w_i is the end of a single word, multi-word, or a sentence. Finally, all of the predictions for all word-parts w are accumulated to decide a single-word token, multi-word token, and sentence boundaries for the given input text t. Word and sentence tokenization using this approach is denoted by Trankit-WT and Trankit-ST, respectively.

Trankit uses the detected words/tokens and sentences for POS tagging at sentence-level. For a given input sentence s, the transformers-generated representation of each of its words is computed by aggregating the representation of its word

[31] `https://trankit.readthedocs.io/en/latest/` - Last visited: 21-October-2021

pieces. These distributed representational vectors are further fed to a *softmax* layer to predict the most probable POS tag for each word in the input sentence s. POS tagging using this approach is denoted by using Trankit-POST.

5 Proposed dataset for UNLT

5.1 Dataset for Urdu word tokenization

Testing data: Another key element of our research is to develop a large benchmark dataset, for the evaluation of our proposed UNLT-WT approach (see Section 4.1.2). The process of developing a benchmark test dataset is divided into three steps: (i) raw text collection, (ii) cleaning and (iii) annotation.

In the first phase, raw data is collected from various online sources (BBC Urdu[32], Express news[33], Urdu Library[34], Urdu Point[35], Minhaj Library[36], Awaz-e-Dost[37] and Wikipedia[38]) by using a Web crawler[39]. The collected raw data is free and publicly available for research purposes, and belongs to following genres: Commerce, Entertainment, Health, Weather, Science and Technology, Sports, Politics and Religion. This collected text contains 61,152 tokens.

In the next phase of the test dataset creation process, the collected raw text was pre-processed (see Section 5.4), which resulted in the removal of 2,152 tokens. The remaining cleaned data is composed of 59,000 tokens (3,583 sentences).

The quality of evaluation of an Urdu word tokenization approach depends on the annotation quality of the test dataset because inconsistent and noisy annotations deteriorate the model's performance. Thus, the annotations were performed by three different annotators (D, E and F). All the annotators are native speakers of Urdu. The annotation process is further divided into three phases: (i) training, (ii) annotation, and (iii) inter-rater agreement calculation and conflict resolution.

In the training phase, two annotators (D and E) annotated a subset of 58 sentences. After that, the inter-annotator agreement was computed for these sentences and conflicting tokens were discussed to further improve the annotation quality. In the annotation phase, the remaining test dataset comprising of 3,525 sentences was annotated by annotators D and E. After the annotation phase, the inter-rater reliability score was computed for the entire test dataset of 59,000 tokens. We obtained the inter-annotator agreement of 86.3% as the annotators had agreement on 50,917 pairs. The Kappa Coefficient was computed to be 78.09%, which is considered as good, considering the levels of difficulty for classifying the merge (space

[32] `http://www.bbc.com/urdu` terms of use: `https://www.bbc.com/urdu/institutional-37588278` - Last checked: 29-June-2019
[33] `http://www.express.pk/` - Last checked: 28-October-2016
[34] `http://www.urdulibrary.org/` - Last checked: 02-November-2016
[35] `http://www.urduweb.org/planet/` - Last checked: 08-November-2016
[36] `http://www.minhajbooks.com/urdu/control/` - Last checked: 08-November-2016
[37] `http://awaz-e-dost.blogspot.co.uk/` - Last checked: 08-November-2016
[38] `https://ur.wikipedia.org/wiki/` - Last checked: 08-November-2016
[39] `https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0023-65A9-5` - Last Checked: 18-December-2016

omission) and compound words (space insertion) into single or multiple tokens (see Section 3.1). Furthermore, the conflicting tokens were annotated, and decisions resolved by the third annotator F, which resulted in a gold standard UNLT-Word Tokenizer-Test (UNLT-WT-Test) dataset.

The Table 3 shows the type-token ratio of the UNLT-WT-Test dataset, that have a total of 59,000 tokens and 5,849 types. The UNLT-WT-Test dataset is stored in the standard "txt" format and is free and publicly available for research purposes (for license and URL see Section 7).

Table 3. Domain wise statistics of the UNLT-WT-Test dataset

Domain	**Tokens**	**Types**	**Domain**	**Tokens**	**Types**
Commerce	7,254	663	Sports	6,868	691
Entertainment	8,578	937	Politics	9,627	777
Health	6,765	651	Religion	5,553	556
Weather	6,606	756	S&T[a]	7,749	823

[a] Science and Technology

Training data: The training dataset for our proposed Urdu word tokenizer was created by using a subset of the HC Corpus (Christensen, 2014). To develop a gold standard training dataset, two million tokens were randomly selected from the following domains: Politics, Culture, Crime & Law, Fashion, Religion, Business & Economy, Science & Technology, Sports, Weather, Education, Health, Entertainment.

After pre-processing (see Section 5.4) the collected raw data, the resulting dataset contained 1.65 million tokens. The pre-processed text is used to create the gold standard training dataset. In the first step, the text is tokenized on the basis of space. After that, a human annotator manually corrected the improperly tokenized words generated in the first step. The final benchmark training dataset (hereafter called UNLT-WT-Train dataset) is comprised of 1.65 million tokens.

The UNLT-WT-Train dataset was used to generate N-grams using the approach described in (Jurafsky and Martin, 2014). Furthermore, we count the occurrences of each unique N-gram type, resulting in a total of 1,335,263 N-gram pairs with the following statistics: *tri*-grams: 636,765, *bi*-grams: 494,988 and *uni*-grams: 203,510.

5.2 Dataset for Urdu sentence tokenization

For the evaluation of our proposed Urdu sentence tokenizer, we created a benchmark dataset (hereafter called UNLT-ST dataset) by following three steps: (i) raw Urdu text collection, (ii) pre-processing of raw data and (iii) annotation. To con-

struct the UNLT-ST dataset, in the first step, we used a Web crawler[40] to extract raw Urdu text of 12.5K sentences from online sources (see Section 5.1) including: BBC Urdu, Express news, Urdu Library, Urdu Point, Minhaj Library, Awaz-e-Dost and Wikipedia. These sources allow their text (content) to be freely used for research purposes. To make the dataset more realistic, we extracted the raw text of different domains and genres including Sports, Politics, Blogs, Education, Literature, Entertainment, Science, Religion, Fashion, Weather, Entertainment, Fiction, Health, Law and Business. BBC Urdu is our largest source of text collection, which contains 3,358 sentences, while the Urdu Point is the smallest one, containing 1,157 sentences. Statistics of sentences collected from other sources are: Awaze-e-Dost: 1,457, Express news: 1,557, Minhaj library: 1,657, Urdu library: 1,357, and Wikipedia: 1,957 sentences.

In the second step, the raw data was pre-processed (see Section 5.4), which resulted in the removal of 2,500 sentences. The remainder of the 10,000 clean sentences are distributed as follows: Awaz-e-Dost: 1,200, BBC Urdu: 2,606, and Express News: 1,297, Minhaj Library: 1,303, Urdu Library: 1,119, Urdu Point: 948 and Wikipedia: 1,527 sentences.

In the third step, the pre-processed text containing 10,000 cleaned sentences was manually tokenized by three annotators (G, H and I). All the annotators are native speakers of Urdu and have good knowledge about Urdu sentence tokenization task. Furthermore, the annotation process was split into three phases: (i) training, (ii) annotation and (iii) inter-rater agreement and conflict resolution.

During the training phase, two annotators (G and H) annotated 1,500 sentences. Subsequently, the inter-annotator agreement was computed for these sentences and conflicting sentences were discussed to further improve the annotation quality. Further, during the annotation phase, the remaining 8,500 sentences were manually annotated by annotators (G and H). In the third phase, the inter-rater agreement score was computed for all 10,000 sentences. We achieved an inter-rater agreement of 89.34%, as the annotators agreed upon 8,934 sentences. Moreover, the Kappa Coefficient was computed to be 81.83% (Cohen, 1968). The conflicting 1,066 sentences were annotated by the third annotator (I) for conflict resolution and this judgement was considered as decisive, resulting in the gold standard UNLT-ST Training/Test dataset.

The UNLT-ST Training/Test dataset consists of 10,000 sentences. In our proposed test dataset, 6,469 period markers are SBM, while 536 are NSBM; 531 exclamation marks are SBM and 198 are NSBM; 421 question marks are SBM and 17 are NSBM; 197 double quotes, are SBM and 9 are NSBM; 382 SBM markers are #, @, $, *; the remaining 2,000 sentence are without any SBM. As can be noted from these statistics, our proposed UNLT-ST Training/Test dataset contains both SBM and NSBM for different characters as well as sentences without any SBM, which makes the dataset much more realistic and challenging. The UNLT-

[40] `https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0023-65A9-5` - Last checked: 19-October-2016

ST Training/Test dataset is saved in standard "txt" format (for licensing and URL see Section 7)

5.3 Dataset for Urdu POS tagging

This section describes the creation of a large dataset (hereafter called UNLT-POS dataset) for the training and testing of the Urdu POS taggers. The dataset creation process was accomplished in three steps: (i) raw text collection, (ii) cleaning process and (iii) annotation process.

To construct a gold-standard Urdu POS tagging dataset, in the first step, a Web crawler (see Section 5.1) was used to extract Urdu text of 239,834 words (14,137 sentences) from various online sources (see Section 5.1) including BBC Urdu, Express news, Urdu library, Urdu point, Minhaj library, Awaz-e-Dost and Wikipedia. To make the dataset more realistic the raw data is from various domains: Sports (23,153), Politics (33,944), Blogs (10,976), Education (12,845), Literature (9,045), Entertainment (13,946), Science and Technology (17,683), Fashion (10,463), Weather (9,459), Business (17,328) and Commerce (10,496), Showbiz (19,503), Fictions (8,678), Health (12,783), Law (8,185), and Religion (21,347).

The raw data was pre-processed (see Section 5.4), which resulted in 200,000 words. The domain and genre distribution of these words is: Sports (20,128), Politics (26,145), Blogs (9,428), Education (10,742), Literature (8,756), Entertainment (10,560), Science and Technology (13,143), Fashion (9,758), Weather (8,996), Business (14,418) and Commerce (9,710), Showbiz (16,228), Fictions (8,084), Health (11,584), Law (6,952), and Religion (15,368).

The UNLT-POS dataset was created using a manual approach. In the first step, a total of 2,000 tokens were POS tagged using the CLE online POS tagger[41] to train annotators. Manual inspection of the tagged data showed that a reasonable number of words are incorrectly tagged, particularly proper nouns, common nouns, verbs, auxiliaries, pronouns, adjectives, cardinal nominal modifiers, adverbs, conjunctions, participles, interjections and foreign fragment. In the second training step, three annotators (A, B and C) manually annotated[42] the automatically tagged data. Annotators A and B initially annotated 2,000 tokens. An inter-annotator agreement was calculated for these tokens and conflicting tagged tokens were discussed to further improve the annotation quality. After the training phase, the 200,000 words was manually annotated by annotators A and B and the inter-annotator agreement was computed on the entire dataset. An inter-annotator agreement of 85.7% was obtained. The Kappa Coefficient was computed to be 77.41% (Cohen, 1968). The conflicting tokens were annotated by the third annotator, resulting in a gold-standard UNLT-POS training/testing dataset saved in "txt" format. As far as we are aware, our UNLT-POS training/testing dataset is the largest manually POS

[41] `http://182.180.102.251:8080/tag/` - Last checked: 06-August-2016

[42] In the training annotation process, the tag assigned by the CLE online POS tagger is retained if the annotator determines that it is correct, otherwise the annotator replaces it with the correct POS tag.

tagged Urdu dataset, free and publicly available for research purposes (for license and URL see Section 7).

For experiments presented in this study, the UNLT-POST gold-standard dataset is randomly divided into two different datasets: (i) consisting of 60K training and 20K of test data (henceforth UNLT-POS-Small training/testing dataset respectively), (ii) consisting of 120K training and 20K for testing (henceforth UNLT-POS-Moderate training/testing dataset respectively).

The detailed statistics of different train/test datasets are shown in Table 4. The rows "Unknown Tokens" and "Unknown Types" of the Table 4 represent the count of total tokens and types (unique tokens) respectively, not seen in the different UNLT-POS training/testing datasets. It has been observed that each test dataset holds 9% to 11% words that are unknown with respect to the training data. These figures are a little higher as compared to the several European languages (Evangelos and George, 1995). However, Table 5 shows the detailed statistics of most frequent POS tags of the UNLT-POS testing dataset.

Table 4. Statistics of three different training/testing datasets for evaluating the performance of Urdu POS taggers

Dataset		**Training set**	**Testing set**
UNLT-POS	Tokens	180,000	20,000
	Types	16,742	2,124
	Unknown Tokens	–	1,948
	Unknown Types	–	246
UNLT-POS-Moderate	Tokens	120,000	20,000
	Types	14,843	2,457
	Unknown Tokens	–	2,078
	Unknown Types	–	273
UNLT-POS-Small	Tokens	60,000	20,000
	Types	9,538	2,801
	Unknown Tokens	–	3,024
	Unknown Types	–	311

5.4 Pre-processing

In this study, various datasets have been used, all these datasets (see Section 5.1 5.2 5.3) are pre-processed as follows. Text in a dataset is cleaned by removing multiple spaces, duplicated text, diacritics as they are optional (only used for altering pronunciation (Mukund et al., 2010)) and HTML tags. Moreover, noise from the data is removed by discarding ASCII and invalid UTF-8 characters, emoticons, asterisks, bullets, right and left arrows (Jawaid et al., 2014). Further,

Table 5. Statistics of most frequent POS tags of UNLT-POS testing dataset

POS Tag[a]	**TC**[b]	**UT**[c]	**POS Tag**	**TC**	**UT**
NN	1,764	123	AUXA	1,023	0
PSP	1,572	0	NNP	1,243	398
VBF	1,129	192	RB	826	63
JJ	1,315	91	AUXT	639	3

[a] NN: Common Noun, PSP: Postposition, VBF: Main Verb Finite, JJ: Adjective, AUXA: Aspectual Auxiliary, NNP: Proper Noun, RB: Common Adverb, AUXT: Tense Auxiliary
[b] Token Count
[c] Unknown tokens

only sentences with three or more words were kept[43]. A language detection tool[44] is used to discard foreign words and a text normalization tool[45].

6 Results and analysis of proposed UNLT tools

6.1 Results of Word tokenizer

Table 6 presents precision, recall, F_1 and accuracy results when training on UNLT-WT-Train dataset, and testing on the UNLT-WT-Test (see Section 5.1) for Urdu word tokenization task by using various approaches. UNLT-WT-SP refers to results obtained using the space-based tokenization approach. UNLT-WT refers to results obtained using our proposed approach (see Section 4.1.2) for Urdu word tokenization. Duranni's word tokenizer (see Section 2.1). Whereas, CLE's word tokenizer refers to an online tokenizer[46]. The Trankit-WT is a deep learning model which implements a light-weight XLM-Roberta transformer (see Section 4.4).

Overall, the best results are obtained by using our proposed UNLT-WT approach (precision = 0.91, recall = 0.87, $F_1 = 0.89$, and accuracy = 0.92). These results show that UNLT-WT is the most appropriate method for Urdu word tokenization on the UNLT-WT-Test dataset. Furthermore, this also shows that combining maximum matching, dictionary lookup and statistical N-gram MLE along with smoothing estimation are helpful in getting good performance on UNLT-WT-Test dataset for Urdu word tokenization task. However, the highest F_1 score of 0.89 for the word

[43] This is calculated by dividing the total words in dataset by the total number of sentence disambiguation markers.
[44] `https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0023-65A9-5` - Last checked: 15-October-2016
[45] Text normalization tool can be downloaded from `http://www.cle.org.pk/software/langproc/urdunormalization.htm-Lastchecked:15-October-2016` is used to keep the Unicode of the characters consistent.
[46] `http://www.cle.org.pk/clestore/segmentation.htm` - Last visited: 18-Dec-2019. Tokenized up-to 100 words at one time and implementation details are not available. The online link refers three papers but does not describe which one of them is used for the creation of CLE Urdu word tokenizer.

tokenization task indicates that Urdu word tokenization is a challenging task leaving room for further improvement.

As expected, the overall results are lower for the baseline space-based tokenization UNLT-WT-SP (precision = 0.55, recall = 0.52, F_1 = 0.54, and accuracy = 0.61) and approach, on UNLT-WT-Test dataset. Durani's word tokenizer reports an accuracy of 0.49, precision of 0.18, recall of 0.20, and F_1 = 0.19. Furthermore, the CLE's Urdu word tokenizer has shown precision = 0.58, recall = 0.56, F_1 = 0.57, and accuracy = 0.73. Whereas, the Trankit-WT has produced the following results: precision = 0.73, recall = 0.69, F_1 = 0.71, and accuracy = 0.80. This highlights the fact that the UNLT-WT-SP, Durrani's, CLE's approaches are not appropriate for Urdu word tokenization tasks.

The comparison of our proposed methods significantly outperformed the recently reported deep neural network based word-piece based tokenization (Trankit-WT). There could be various possible reasons for this notable difference in the scores including: simple space-based splitting of the raw input text as a pre-processing step in Trankit-WT, seems to be a major contributor in producing low scores for word tokenization. This pre-process space-splitting could be efficient for many languages like English, Spanish, etc. but not suitable for the Urdu language. As has been described previously that spaces play a major role in word tokenization (see Section 3.1). We have further observed that Trankit-W is very poor in capturing multi-words.

Table 6. Results obtained on UNLT-WT-Test dataset using various techniques

Technique	**Precision**	**Recall**	**F$_1$**	**Accuracy**
UNLT-WT-SP	0.55	0.52	0.54	0.61
UNLT-WT	0.91	0.87	0.89	0.92
Durani's	0.18	0.20	0.19	0.49
CLE's	0.58	0.56	0.57	0.73
Trankit-WT	0.73	0.69	0.71	0.80

While analysing the errors of the proposed UNLT-WT approach, we observed that it does not explicitly handle unknown words for space omission, and this resulted in splitting an unknown Urdu morpheme into smaller morphemes. For instance, the word کثراللسان (KSYR ALLSAN, 'multilingual') erroneously split into کثی (KSY), را, (RA) للس (LLS), and ان (AN). Likewise, it might be less appropriate when a word is a combination of known and unknown morphemes, for instance, شہبازکو جانیدو (SHBAZ KO JANE DO, 'let the Shahbaz go'). For space insertion, some compound words were not present in compound words dictionary, another major cause of incorrect word tokenization.

6.2 Results of Sentence tokenizer

Table 7 presents precision, recall, F_1 and error rate results[47] on the UNLT-ST Train/Test dataset[48] (see Section 5.2) for various Urdu sentence tokenization approaches[49] (for other two, see Section 4.2.2). Trankit-ST is is a light-weight XLM-Roberta transformer for sentence tokenization. (see Section 4.4)

Overall, the best results are obtained using our proposed UNLT-ST-ML approach, precision = 0.90, recall = 0.92, F_1 = 0.91, error rate = 0.09. The lowest results are obtained using a baseline UNLT-ST-PQEQM (precision = 0.89, recall = 0.19, F_1 = 0.31, error rate = 0.83). Whereas, another proposed rule based UNLT-ST-RB approach shows a precision of 0.81, recall of 0.74, F_1 of 0.77 and error rate of 0.29. This shows that combining various features in SVM is helpful in producing a good performance on the UNLT-ST Train/Test dataset. Trankit-ST a deep learning based approach has produced results as follows: precision = 0.86, recall = 0.83, F_1 = 0.85, error rate = 0.14. It is worth mentioning here that Trankit-S has produced more accurate results than two other sentence tokenization approaches i.e. UNLT-ST-RB / PQEQM. However, the highest F_1 score of 0.91 for sentence tokenization task indicates that Urdu sentence tokenization is a challenging task and there is still room for further improvement.

For the UNLT-ST-PQEQM approach, which uses different characters as sentence boundary indicators, the performance is high in term of precision of 0.89. The likely reason for this is the majority of sentences in Urdu text are terminated using these characters. However, other evaluation measures show very low results. This highlights the fact that these characters alone are not suitable for the Urdu sentence tokenization task. As far as the rule base (UNLT-ST-RB) approach is concerned it shows good results as compared to the baseline approach. This shows that combining various heuristics, regular expressions and dictionary lookup is helpful in producing a good performance on the UNLT-ST Test dataset. However, this method also fails to detect sentence boundary where sentences are ended with out any SBM. While manually analysing the errors of the proposed UNLT-ST-RB approach, we came across some scenarios where our proposed approach failed to accurately tokenize sentences. It was found that NSBM including: ':', '||', '$', '∗', '@' and '#' are the major reasons for incorrect tokenization of sentences. Moreover, the period used between different abbreviations also caused misclassification. Finally, all those sentences which end without any sentence boundary marker are also declassified.

[47] For ML approach we have used 10-fold cross-validation.
[48] For baseline and rule based approaches we have used the entire UNLT-ST Train/Test dataset as a test dataset.
[49] Baseline UNLT-ST-PQEQM– tokenization on the basis of "period", "question mark", "exclamation mark", and "double quotes" characters.

Table 7. Results obtained by using various sentence tokenization approaches on UNLT-ST-Train/Test dataset

Technique	**Precision**	**Recall**	**F_1**	**Error rate**
UNLT-ST-PQEQM	0.89	0.19	0.31	0.83
UNLT-ST-RB	0.81	0.74	0.77	0.29
UNLT-ST-ML	0.90	0.92	0.91	0.09
Trankit-ST	0.86	0.83	0.85	0.14

6.3 Results of POS tagger

Table 8 presents accuracy results when trained and tested on the UNLT-POS-Small, UNLT-POS-Moderate, UNLT-POS test datasets (see Section 5.3) for the Urdu POS tagging tasks by using different models (see Sections 4.3.2 and 4.4).

Overall best results are obtained using our proposed T-HMM-GT-Suf-MA followed by T-HMM-GT-Suf-MA, and MEn-Suf-MA POS tagging models, 95.59%, 95.14% and 94.20% respectively. This shows that combining various stochastic and smoothing techniques with language dependent features are helpful in producing a very good performance on the UNLT-POS test dataset. The highest accuracy score of 95.59% indicates that the Urdu POS tagging task is challenging and there is still room for improvement. It can also be noted from these results that our proposed POS tagging approach (T-HMM-GT-Suf-MA) outperforms both baseline approaches BL-MFT[50] (accuracy = 84.72%) and BL-CLE[51] (Ahmed et al., 2014) (which uses Decision Trees along with a smoothing technique of Class Equivalence) (accuracy = 88.45%) on UNLT-POS test dataset. Furthermore, Trainkit-POST has produced an accuracy of 92.67%. This has good accuracy as compared to baseline and several other approaches. The reason for using the BL-CLE model as a baseline approach is that, currently, this is the only POS tagger available for Urdu which uses CLE Urdu POS tagset (see Section 4.3.1). Therefore, we can compare the results of CLE Urdu POS tagger with our proposed UNLT-POS tagger.

We can further observe that the *tri*-gram HMM based models can produce good results if incorporated with linear interpolation, suffix as well as Morphological Information (MI). Certainly, using MI along with linear interpolation gives better results as compared to suffix, but what is significant to note, using all the information together improved the accuracy of the models, T-HMM-LI-Suf-MA: 95.14%, T-HMM-LaE-Suf-MA: 93.74%, T-HMM-LiE-Suf-MA: 93.97%, T-HMM-GT-Suf-MA: 94.48%, T-HMM-KN-Suf-MA: 95.59% and MEn-Suf-MA: 94.20%. Furthermore, it can be observed, T-HMM-LI, T-HMM-LaE, T-HMM-LiE, T-HMM-GT, and T-HMM-KN produce accuracies of 87.34%, 85.92%, 86.89%, 87.01%, and 87.51% re-

[50] In this each word in the test data will be assigned the POS tag based on the most frequent POS tag in the training data.
[51] `http://182.180.102.251:8080/tag/` - Last checked: 17-July-2018

Table 8. Results obtained using various POS tagging models based on several approaches on different POS test datasets

Approaches[a]	**Model**	**Accuracy**[b]		
		D1	**D2**	**D3**
Most frequent tag	BL-MFT	-	-	84.72
Decision Tree	BL-CLE	-	-	88.45
Trankit-POST	T-DL	-	-	92.67
tri-gram HMM, LI	T-HMM-LI	67.14	80.34	87.34
tri-gram HMM, LI, suffix	T-HMM-LI-Suf	83.23	87.91	91.53
tri-gram HMM, LI, MI	T-HMM-LI-MA	88.37	90.39	92.27
tri-gram HMM, LI, suffix, MI	T-HMM-LI-Suf-MA	90.87	93.76	95.14
tri-gram HMM, LaE	T-HMM-LaE	65.97	79.14	85.92
tri-gram HMM, LaE, suffix	T-HMM-LaE-Suf	80.42	86.39	89.98
tri-gram HMM, LaE, MI	T-HMM-LaE-MA	87.88	89.74	90.19
tri-gram HMM, LaE, suffix, MI	T-HMM-LaE-Suf-MA	89.04	91.64	93.74
tri-gram HMM, LiE	T-HMM-LiE	66.98	80.02	86.89
tri-gram HMM, LiE, suffix	T-HMM-LiE-Suf	82.78	87.13	90.93
tri-gram HMM, LiE, MI	T-HMM-LiE-MA	88.13	90.02	91.69
tri-gram HMM, LiE, suffix, MI	T-HMM-LiE-Suf-MA	90.23	92.59	93.97
tri-gram HMM, GT	T-HMM-GT	67.02	80.10	87.01
tri-gram HMM, GT, suffix	T-HMM-GT-Suf	82.98	87.27	91.05
tri-gram HMM, GT, MI	T-HMM-GT-MA	88.19	90.07	91.90
tri-gram HMM, GT, suffix, MI	T-HMM-GT-Suf-MA	90.57	93.03	94.48
tri-gram HMM, KN	T-HMM-LiE	67.24	80.36	87.51
tri-gram HMM, KN, suffix	T-HMM-LiE-Suf	83.23	87.98	91.70
tri-gram HMM, KN, MI	T-HMM-LiE-MA	88.42	90.52	92.39
tri-gram HMM, KN, suffix, MI	T-HMM-LiE-Suf-MA	91.02	93.99	95.59
MaEn, CW, WN	MEn	80.59	84.92	88.31
MaEn, CW, WN, suffix	MEn-Suf	84.43	88.06	92.56
MaEn, CW, WN, MI	MEn-MA	88.32	89.49	93.11
MaEn, CW, WN, suffix, MI	MEn-Suf-MA	90.26	93.31	94.20

[a] LI: Linear Interpolation, MI: Morphological Information, LaE: Laplace Estimation, LiE: Lidstone's Estimation, GT: Good-Turing, KN-Kneser-Ney MaEn: Maximum Entropy, CW: Context Window, WN: Word Number

[b] D1: UNLT-POS-Small training/testing dataset, D2: UNLT-POS-Moderate training/testing dataset, D3: UNLT-POS training/testing dataset

spectively, on the UNLT-POS dataset. For the case of MEn, the reported accuracy is 88.31%. One important observation here is that by using smoothing and language dependent features, the proposed Urdu POS tagging accuracies can be improved as compared to BL-MFT and BL-CLE models.

It can also be observed from the Table, that T-HMM-GT performs better than the other four models T-HMM-LI, T-HMM-LaE, T-HMM-LiE, T-HMM-GT, and T-HMM-KN on UNLT-POS, UNLT-POS-Small, and UNLT-POS-Moderate test datasets. Moreover, the accuracy of T-HMM-LaE model is slightly poorer than the other HMM based models (T-HMM-LI, T-HMM-LiE, T-HMM-GT, and T-HMM-KN), with UNLT-POS-Small data due to model overfitting. However, such discrepancies are alleviated with the increase of training data (UNLT-POS-Moderate and UNLT-POS training datasets).

It has been further observed that language dependent features increased the accuracy of the models to a certain extent, even if trained on a UNLT-POS-Moderate training dataset. However, with different features along with smoothing, the increase in the model accuracy is higher when training data is smaller. For instance, T-HMM-LI-MA and T-HMM-LI-Suf models improved around 16%, 7% and 4%, and 21%, 10% and 5% respectively over the T-HMM-LI models, for UNLT-POS, UNLT-POS-Small, and UNLT-POS-Moderate test datasets.

From the above observations, it can be concluded that using MI and suffix, increases in the model accuracy are higher for UNLT-POS-Small and UNLT-POS-Moderate training datasets. It is also important to note, the T-HMM-KN-MA models give an approximate improvement of around 5%, 7% and 1% over the T-HMM-KN-Suf model for UNLT-POS-Small, UNLT-POS-Moderate and UNLT-POS training dataset respectively. However, integrating all of them, an improvement has been observed in T-HMM-KN-Suf-MA models which are 8%, 6%, and 3% improved with respect to T-HMM-KN-Suf model in case of UNLT-POS-Small, UNLT-POS-Moderate and UNLT-POS training dataset. It can also be noticed that similar results have been observed for the other two (T-HMM-LaE, T-HMM-LiE, T-HMM-GT and T-HMM-LI) HMM based models. However, T-HMM-LiE performed better than the T-HMM-LaE model, but with the higher training data, the performance of these models are somewhat comparable.

MEn models outperform all others with smaller training data but contrasting results have been observed with large training data. It is worth noting that MEn along with suffix and morphological information has positive effects with poor resources. Our results show the T-HMM-GT-Suf-MA and MEn-Suf-MA are more accurate than others, providing support for our further analysis based on such models.

Table 9 shows cases where the MEn-Suf-MA model performs better than T-HMM-KN-Suf-MA, by comparing the accuracies of open class tags for known and unknown words on the UNLT-POS testing dataset. Our results show that the T-HMM-KN-Suf-MA model shows poor accuracy while predicting proper nouns (NNP) over the MEn-Suf-MA model. Mostly the proper nouns (NNP) in T-HMM-KN-Suf-MA model are erroneously classified as an adjective (JJ). Furthermore, it is worth noting again that in Urdu, there is no discrimination between upper and lower-case characters, also using an adjective as a proper noun is frequent in Urdu

e.g. کبیر (KBYR, 'big') and صغیر (SGHYR, 'small'). Another reason for misclassification in tagging of the proper nouns is that many of them end with negation marker or pronoun e.g. the ناگینہ ('Nagyna') end with the نہ (NH, 'no') or the NNP ناذیہ ('Nazyh') which end with the یہ (YH, 'this'), a pronoun. These errors needs further investigation.

Table 9. Accuracies of open class tags on UNLT-POS testing dataset using T-HMM-KN-Suf-MA and MEn-Suf-MA

Tag	**T-HMM-KN-Suf-MA**		**MEn-Suf-MA**	
	Known	**Unknown**	**Known**	**Unknown**
NN	95.10	80.34	92.32	78.23
NNP	73.98	56.09	76.56	70.74
JJ	92.02	63.58	89.54	61.97
RB	81.98	57.98	84.45	64.33
VBF	93.14	73.01	92.47	72.03

Similarly, in the case of common adverb (RB), the accuracy of the MEn-Suf-MA model is approximately 2% and 6% higher for known and unknown words respectively. However, the performance of MEn-Suf-MA for all other open class tags did not improve over the T-HMM-KN-Suf-MA. It is further observed that with the increase of unknown words, the accuracy of the MEn model has reduced.

The most prominent causes for Urdu POS tag misclassification are that there is no clear distinction between noun and proper noun, dropping of words is also frequent, if a noun in a noun phrase is dropped the adjective becomes a noun in that phrase, the highly inflected nature of Urdu, and ambiguity between noun and verb is due to verbal nouns.

7 Conclusion and future directions

In this paper, we have described a novel Urdu Natural Language Toolkit by integrating word and sentence tokenizers as well as a POS tagger. Words are tokenized by coupling a rule-based morpheme matching method with a *tri*-gram stochastic language model, backed-off to *bi*-gram Maximum Likelihood Estimation supplemented by smoothing technique for unseen words. We also developed large compound word and morpheme dictionaries, which were used in our proposed Urdu word tokenizer. A large benchmark training and testing datasets are also created. The training dataset comprises 1,361,179 N-grams (1.65 million tokens) whereas, test dataset contains 59,000 manually tokenized words. Moreover, we have compared our results with state-of-the-art deep learning methods for comparison purposes. The results show that our proposed Urdu word tokenizer obtained precision

of 0.96, recall of 0.92, F_1 of 0.94, and accuracy of 0.97. The sentence tokenization approach is formed by using rules extracted from a large raw Urdu corpus, regular expressions and dictionary lookup. Our proposed Urdu sentence tokenizer obtained promising results on a large and new generated gold-standard test dataset (precision = 91.08%, recall = 94.14%, F_1 = 92.59%, and error rate = 6.85%). We have described 24 different stochastic models (and two baseline models) for the Urdu POS tagging task. Each of these models has a unique stochastic scheme supplemented with various language features and smoothing estimations. In addition, a large gold-standard training/testing dataset was generated. Results show that the best performance is achieved by the T-HMM-KN-Suf-MA POS tagger which is a combination of *tri*-gram HMM, Kneser-Ney, suffix, and morphological information, achieving an accuracy of 95.59%. These resources have been made publicly available to the research community at `https://github.com/UCREL/UNLT` and `https://doi.org/10.17635/lancaster/researchdata/494` under the terms of the Creative Commons Attribution 4.0 International License[52] and GNU General Public License v3.0[53].

Our next primary target for follow-up research work will be to extend the toolkit by developing and integrating various NLP tools. As far as the word tokenization is concerned we aim to adopt machine learning approaches (conditional random field, maximum entropy etc.) to learn the morphological pattern of the valid morphemes (instead of morphemes lookup) and to handle out-of-vocabulary words in morpheme matching process of space omission problem. For the sentence tokenizer we plan to develop a hybrid approach for Urdu sentence tokenization task i.e. rule-based along with artificial neural network. Finally, in the future, we aim to deal with unknown words and will adopt further statistical methods (CRF, deep learning etc.) along with heuristic rules to increase the POS tagging accuracy for Urdu text.

Acknowledgements

This work has been supported by the COMSATS University Islamabad, Lahore Campus, Pakistan and Lancaster University, U.K. under the Split-Site Ph.D. programme.

References

Abdelhamid, A. A., Abdulla, W. H., and MacDonald, B. 2012. WFST-based large vocabulary continuous speech decoder for service robots. In *Proceedings of the International Conference on Imaging and Signal Processing for Healthcare and Technology (ISPHT'12), Baltimore, USA*, pp. 150–4.

Akita, Y., Saikou, M., Nanjo, H., and Kawahara, T. 2006. Sentence boundary detection of spontaneous Japanese using statistical language model and support vector machines. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'06), Pennsylvania, USA*, pp. 1033–6.

[52] `https://creativecommons.org/licenses/by/4.0/-` Last checked: 15-September-2020
[53] `https://www.gnu.org/licenses/gpl-3.0.en.html-` Last checked: 11-November-2021

Albared, M., Omar, N., Aziz, M. J., and Nazri, M. Z. 2010. Automatic part of speech tagging for Arabic: an experiment using bigram hidden Markov model. In *International Conference on Rough Sets and Knowledge Technology, (RSKT'10), Beijing, China*, pp. 361–70.

Ahmed, T., Urooj, S., Hussain, S., Mustafa, A., Parveen, R., Adeeba, F., Hautli, A., and Butt, M. 2014. The CLE Urdu POS tagset. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland*, pp. 2920–5.

Anwar, W., Wang, X. L. Li., and Wang, X. L. 2007a. A statistical based part of speech tagger for Urdu language. In *IEEE International Conference on Machine Learning and Cybernetics (ICMLC'07), Hong Kong, China*, Volume 6, pp. 3418–24.

Anwar, W., Wang, X. L. Li., and Wang, X. L. 2007b. Hidden Markov model based part of speech tagger for Urdu. *Information Technology Journal* **6**(8): 1190–8.

Azimizadeh, A., Arab, M.M., and Quchani, S.R. 2008. Persian part of speech tagger based on Hidden Markov Model. In *Proceedings of the 9th International Conference on the Statistical Analysis of Textual Data (JADT'08), Lyon, France*, pp. 121-8.

Bhat, R.A, and Sharma, D.M. 2012. A dependency treebank of Urdu and its evaluation. In *Proceedings of the Sixth Linguistic Annotation Workshop (LAW VI'12), Jeju, Republic of Korea*, pp. 157-65.

Bird, S., Klein, E., and Loper, E. 2009. *Natural language processing with Python.* "O'Reilly Media, Inc.".

Bird, S., Klein, E., Loper, E., and Baldridge, J. 2008. Multidisciplinary instruction with the natural language toolkit. In *Proceedings of the 3rd Workshop on Issues in Teaching Computational Linguistics (TeachCL'08), Ohio, USA*, Volume 13, pp. 62–70.

Bögel, T., Butt, M., Hautli, A., and Sulger, S. 2007. Developing a finite-state morphological analyzer for Urdu and Hindi. In *Proceedings of the Finite-state methods and natural language processing : 6 th International Workshop (FSMNLP' 07), Potsdam, Germany,*, pp. 86–96.

Brants, T. 2000. TnT: a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000, Seattle, Washington, USA*, pp. 224–31.

Butt, J. M. 1995. The structure of complex predicates in Urdu. *Ph.D. thesis, Center for the Study of Language (CSLI), department of linguistics, Stanford University.*

Chen, S. F., and Goodman, J. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* **13**(4): 359–94.

Christensen, H. 2014. HC corpora. *http://www.corpora.heliohost.org/ (Last checked: 05-March-2017).*

Christer, S. 1996. Handling sparse data by successive abstraction. In *Proceedings of the the 16th International Conference on Computational Linguistics (COLING'96), Copenhagen, Denmark*, pp. 895–900.

Cohen, J. 1968. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* **70**(4): 213–20.

Conneau, A., Khandelwal, K., Goyal. N., Chaudhary, V., Wenzek, G., Guzman, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. 2020. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116* **1**: 1–12.

Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. 2002. GATE: a framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, USA*, pp. 168–75.

Cunningham, H., Tablan, V., Roberts, A., and Bontcheva, K. 2013. Getting more out of biomedical documents with GATE's full lifecycle open source text analytics. *PLoS Computational Biology* **9**(2): e1002854.

Curran, J. R., and Clark, S. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th conference on European chapter of the Association for Computational Linguistics (EACL'03), Budapest, Hungary*, Volume 1, pp. 91–8.

Dandapat, S. 2008. Part of specch tagging and chunking with maximum entropy model. In *Proceedings of the IJCAI Workshop on Shallow Parsing for South Asian Languages (IJCAI'08), Hyderabad, India.*, pp. 29–32.

Daud, A., Khan, W., and Che, D. 2016. Urdu language processing: a survey. *Artificial Intelligence Review* **47**(3): 279–311.

Dietzel, A., and Maynard, D. 2015. Climate change: a chance for political re-engagement? In *Proceedings of the Political Studies Association 65th Annual International Conference (PSA'15), Sheffield, UK*, pp. 1–19.

Durrani, N., and Hussain, S. 2010. Urdu word segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, Los Angeles, California, USA*, pp. 528–36.

Ekbal, A., Haque, R., and Bandyopadhyay, S. 2008. Maximum entropy based Bengali part of speech tagging. *Advances in Natural Language Processing and Applications, Research in Computing Science (RCS) Journal* **33**(8): 67–78.

Evangelos, D., and George, K. 1995. Automatic stochastic tagging of natural language texts. *Computational Linguistics* **21**(2): 137–63.

Ferrucci, D., and Lally, A. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering* **10**(3-4): 327–348.

Fu, G., Kit, C., and Webster, J. J. 2008. Chinese word segmentation as morpheme-based lexical chunking. *Information Sciences* **178**(9): 2282–96.

Garside, R., and Smith, N. 1997. A hybrid grammatical tagger: CLAWS4. *Corpus Annotation: Linguistic Information from Computer Text Corpora, Longman, London* **102**(1): 102–21.

Giménez, J., and Marquez, L. 2004. SVMTool: a general POS tagger generator based on Support Vector Machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04), Lisbon, Portugal*, pp. 43–6.

Gries, ST,. and John, N. 2014. Research methods in linguistics. pp. 257–87. Cambridge University Press, UK

Hardie, A. 2003. Developing a tagset for automated part-of-speech tagging in Urdu. In *In Archer, D, Rayson, P, Wilson, A, and McEnery, T (eds.) Proceedings of the Corpus Linguistics 2003 conference. UCREL Technical Papers, Lancaster, UK*, Volume 16, pp. 298–307.

Hardie, A. 2004. The computational analysis of morphosyntactic categories in Urdu. *Ph. D. thesis, Lancaster University, UK.*

Hautli, A., and Sulger, S. 2011. Extracting and classifying Urdu multiword expressions. In *Proceedings of the the ACL-HLT Student Session (ACL-HLT'11), Portland, OR, USA.*, pp. 24–9.

Hearst, MA., Dumais, ST., Osuna, E., Platt, J., and Scholkopf, B. 1998. Support vector machines. *IEEE Intelligent Systems and their applications* **13**(4): 18–28.

Javed, I. 1985. Nai Urdu qawaid. *Urdu Development Board, New Delhi*.

Jawaid, B., Kamran, A., and Bojar, O. 2014. A tagged corpus and a tagger for Urdu. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'09), Reykjavík, Iceland.*, pp. 2938–43.

Jeffreys, H. 1998. *The theory of probability*, Volume 3rd. Oxford University Press.

Joshi, N,. Darbari, H., and Mathur, I. 2013. HMM based POS tagger for Hindi. In *Proceedings of the 2013 International Conference on Artificial Intelligence and Soft Computing (AISC'13), Bangalore, India*, pp. 341-9.

Jurafsky, D., and Martin, J. 2014. *Speech & language processing, 2nd edition*, Volume 3. Pearson London.

Khan, S. A., Anwar, W., Bajwa, U. I., and Wang, X. 2012. A light weight stemmer for Urdu language: a scarce resourced language. In *Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP-COLING'12), Mumbai, India*, pp. 69–78.

Kreuzthaler, M., and Schulz, S. 2015. Detection of sentence boundaries and abbreviations in clinical narratives. In *BMC medical informatics and decision making* **15**(2): 1–13.

Kudo, T., and Richardson. J. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), Brussels, Belgium*, pp. 66-71.

Kudo, T. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), Melbourne, Australia.*, pp. 66–75.

Kwartler, T. 2017. Text mining in practice with R. pp. 237–69. John Wiley & Sons, Ltd, Chichester, UK

Lehal, G. S. 2010. A word segmentation system for handling space omission problem in Urdu script. In *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), the 23rd International Conference on Computational Linguistics, Beijing, China*, pp. 43–50.

Malik, A. 2009. A hybrid model for Urdu Hindi translation. In *Proceedings of the Named Entities WorkShop (NEWS'09), Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, (ACL-IJCNLP'09) Singapore*, pp. 177–85.

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*, Volume 999. Cambridge Massachusetts:MIT Press.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, Maryland, USA*, pp. 55–60.

Maynard, D., Greenwood, M.A., Roberts, I., Windsor, G., and Bontcheva, K. 2015. Real-time social media analytics through semantic annotation and linked open data. In *proceedings of the ACM Web Science Conference (WebSci'15), Oxford, United Kingdom*, pp. 46–8.

Muaz, A., Ali, A., and Hussain, S. 2009. Analysis and development of Urdu POS tagged corpus. In *Proceedings of the 7th Workshop on Asian Language Resources (ALR'7), Suntec, Singapore*, pp. 24–9.

Mukund, S., Srihari, R., and Peterson, E. 2010. An information-extraction system for Urdu—a resource-poor language. *ACM Transactions on Asian Language Information Processing (TALIP)* **9**(4): 1–43.

Naz, F., Anwar, W., Bajwa, U.I., and Munir, E. 2012. Urdu Part of Speech Tagging Using Transformation Based Error Driven Learning. *World Applied Sciences Journal (WASJ)* **16**(3): 437–48.

Nguyen, M., Lai, V., Veyseh, A.P.B., and Nguyen, T.H. 2021. Trankit: A Light-Weight Transformer-based Toolkit for Multilingual Natural Language Processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, Online*, pp. 80–9.

Platts, J. T. 1909. *A grammar of the Hindustani or Urdu language.* London: Crosby Lockwood and Son, republished in 2002 by Sang-e-Meel Publications, Lahore.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**(2): 257–85.

Raj, S., Rehman, Z., Rauf, S., Siddique, R., and Anwar, W. 2015. An artificial neural network approach for sentence boundary disambiguation in Urdu. *The International Arab Journal of Information Technology* **12**(4): 395–400.

Rashid, R., and Latif, S. 2012. A dictionary based Urdu word segmentation using maximum matching algorithm for space omission problem. In *Proceedings of the International Conference on Asian Language Processing (IALP'17), Hanoi, Vietnam*, pp. 101–4.

Ratnaparkhi, A. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'96), New Jersey, USA*, Volume 1, pp. 133–42.

Rehman, Z., Anwar, W., and Bajwa, U.I. 2011. Challenges in Urdu text tokenization and sentence boundary disambiguation. In *Proceedings of the 2nd Workshop on South Southeast Asian Natural Language Processing (WSSANLP'11), Chiang Mai, Thailand*, pp. 40-5.

Rehman, Z., and Anwar, W. 2012. A hybrid approach for Urdu sentence boundary disambiguation. *The International Arab Journal of Information Technology* **9** (3): 250–5.

Rehman, Z., Anwar, W., Bajwa, U. I., Xuan, W., and Chaoying, Z. 2013. Morpheme matching based text tokenization for a scarce resourced language. *PloS One* **8**(8): e68178.

Riaz, K. 2012. Comparison of Hindi and Urdu in computational context. *International Journal of Computational Linguistics and Natural Language Processing (IJCLNLP)* **1**(3): 92–7.

Riaz, K. 2010. Rule-based named entity recognition in Urdu. In *Proceedings of the 2010 Named Entities WorkShop (NEWS'10), Uppsala, Sweden*, pp. 126–35.

Rush, A.M., Chopra, S., and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'15), Lisbon, Portugal*, pp. 1–11.

Saeed, A., Nawab, R.M.A., Stevenson, M., and Rayson, P. 2018. A word sense disambiguation corpus for Urdu. *Language Resources and Evaluation (LRE)* **1**(3): 1–22.

Sajjad, H. 2007. Statistical part of speech tagger for Urdu. *Unpublished MS Thesis, National University of Computer and Emerging Sciences, Lahore, Pakistan.*

Sajjad, H., and Schmid, H. 2009. Tagging Urdu text with parts of speech: a tagger comparison. In *Proceedings of the 12th Conference of the European Chapter of the ACL, Athens, Greece*, pp. 692–700.

Schmid, H. 1994b. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the international conference on new methods in language processing (NeMLaP), Manchester, UK.*, Volume 12, pp. 44–9.

Schmid, H. and Laws, F. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08), Manchester, UK*, Volume 1, pp. 777–84.

Schmidt, R. L. 1999. Urdu, an Essential Grammar *(Routledge Essential Grammars)*, Volume 1. Psychology Press.

Shafi, J. 2020. An Urdu Semantic Tagger–Lexicons, Corpora, Methods, and Tools *Ph. D. thesis, Lancaster University, UK.*

Sharjeel, M., Nawab, R.M.A., and Rayson, P. 2017. COUNTER: corpus of Urdu news text reuse. *Language Resources and Evaluation* **1**(3): 777–803.

Tafseer, A. 2009. Roman to Urdu transliteration using wordlist. In *Proceedings of the Conference on Language and Technology (CLT'09), Lahore, Pakistan.*, pp. 1–8.

Thede, S. M., and Harper, M. P. 1999. A second-order hidden Markov model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics (ACL'99), College Park, Maryland*, pp. 175–82.

Vaswani, A., Shazeer, N., Parmar, Uszkoreit, N., Jones, L., Gomez, A.N., Kaiser, L., and Polosukhin, I. 2017. Attention Is All You Need. *arXiv preprint arXiv:1706.03762* **1**: 1–5.

Viterbi, A. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory* **13**(2): 260–9.

Wicaksono, A.F., and Purwarianti. A. 2010. HMM based part-of-speech tagger for Bahasa Indonesia. In *Proceedings of the Fourth International MALINDO Workshop, Jakarta, Indonesia*, pp. 1-7.

Yi, C. 2015. An English pos tagging approach based on maximum entropy. In *Proceedings of the International Conference on Intelligent Transportation, Big Data and Smart City (ICITBS'15), Halong Bay, Vietnam.*, pp. 81–4.