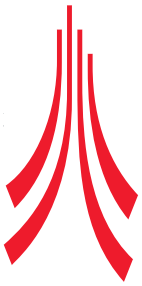
Dealing with Spelling Variation in Early Modern English Texts



Alistair Baron B.Sc. (Hons)

School of Computing and Communications

Lancaster University

A thesis submitted for the degree of

Doctor of Philosophy

February 2011

Abstract

Early English Books Online contains digital facsimiles of virtually every English work printed between 1473 and 1700; some 125,000 publications. In September 2009, the Text Creation Partnership released the second instalment of transcriptions of the EEBO collection, bringing the total number of transcribed works to 25,000. It has been estimated that this transcribed portion contains 1 billion words of running text. With such large datasets and the increasing variety of historical corpora available from the Early Modern English period, the opportunities for historical corpus linguistic research have never been greater. However, it has been observed in prior research, and quantified on a large-scale for the first time in this thesis, that texts from this period contain significant amounts of spelling variation until the eventual standardisation of orthography in the 18th century.

The problems caused by this historical spelling variation are the focus of this thesis. It will be shown that the high levels of spelling variation found have a significant impact on the accuracy of two widely used automatic corpus linguistic methods – Part-of-Speech annotation and key word analysis. The development of historical spelling normalisation methods which can alleviate these issues will then be presented. Methods will be based on techniques used in modern spellchecking, with various analyses of Early Modern English spelling variation dictating how the techniques are applied. With the methods combined into a single procedure, automatic normalisation can be performed on an entire corpus of any size. Evaluation of the normalisation performance shows that after training, 62% of required normalisations are made, with a precision rate of 95%.

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university.

Alistair Baron

Acknowledgements

Without the help of friends and family, it would have been impossible for me to even come close to completing this thesis. First and foremost, I'd like to thank my supervisor, Paul Rayson, who has been unbelievably supportive and patient throughout. Paul provided the spark I needed during my undergraduate project back in 2005 and gave me the confidence I needed to apply for the PhD process – he has put up with me and inspired me ever since.

Many others have given a helping hand with my research and the writing of this thesis. In particular, I'd like to thank Dawn Archer who has provided invaluable advice at all stages of my PhD. Many people at Infolab21 have made it such a brilliant place to work, special thanks to Eddie Bell, Yehia El-khatib, John Mariani, Andy Molineux, Chris Paice, Nick Smith and Gaz Tyson for their help, as well as Awais, James and Phil for their patience during the write-up! I'd also like to thank all of the people who have taken the time to try out the VARD 2 software, your feedback has been invaluable.

Just as important, I'd like to thank all those who have supported me at home: My parents, brothers, sisters, future in-laws and other family members for their love and support – particularly Judith for taking the time to read my thesis and provide brilliant feedback – and to all of my friends for the fun times when I've needed a break.

Leaving the most important until last, with me every step of the way has been Mel, my wonderful fianceé. Good things come to those who wait – I've finished now, so we can get married!

Publications

The following have been published as part of the research presented in this thesis. Where appropriate, portions of this thesis are based on my contributions to these publications without citation. Where research and text should be credited to a co-author, rather than myself, the work has been cited accordingly.

Baron, A. (2006). Standardising Spelling in Early Modern English Texts. Poster at the *Faculty of Science and Technology Research Conference*, Lancaster University, UK, 20th December 2006.

Rayson, P., Archer, D., **Baron, A.** and Smith, N. (2007). Tagging historical corpora - the problem of spelling variation. In *Proceedings of Digital Historical Corpora, Dagstuhl-Seminar 06491*, International Conference and Research Center for Computer Science, Schloss Dagstuhl, Wadern, Germany, December 3rd-8th 2006. ISSN 1862-4405.

Rayson, P., Archer, D., **Baron, A.**, Culpeper, J. and Smith, N. (2007). Tagging the Bard: Evaluating the accuracy of a modern POS tagger on Early Modern English corpora. In Davies, M., Rayson, P., Hunston, S. and Danielsson, P. (eds.) *Proceedings of the Corpus Linguistics Conference: CL2007*, University of Birmingham, UK, 27-30 July 2007.

Rayson, P., Archer, D., **Baron, A.** and Smith, N. (2008). Travelling Through Time with Corpus Annotation Software. In Lewandowska-Tomaszczyk, B. (ed.) *Corpus Linguistics, Computer Tools, and Applications State of the Art: PALC 2007*. Peter Lang, Frankfurt am Main.

Baron, A. and Rayson, P. (2008). VARD 2: A tool for dealing with spelling variation in historical corpora. *Proceedings of the Postgraduate Conference in Corpus Linguistics*, Aston University, Birmingham, 22 May 2008.

Baron, A., Rayson, P. and Archer, D. (2009). The extent of spelling variation in Early Modern English. Presented at *ICAME 30*, Lancaster University, UK, 27-31 May 2009.

Baron, A., Rayson, P. and Archer, D. (2009). Automatic Standardization of Spelling for Historical Text Mining. In *Proceedings of Digital Humanities 2009*, University of Maryland, USA, 22-25 June 2009, pp. 309–312

Baron, A. and Rayson, P. (2009). Automatic standardization of texts containing spelling variation, how much training data do you need? In Mahlberg, M., Gonzlez-Daz, V. and Smith, C. (eds.) *Proceedings of the Corpus Linguistics Conference, CL2009*, University of Liverpool, UK, 20-23 July 2009.

Baron, A., Rayson, P. and Archer, D. (2009). Word frequency and key word statistics in historical corpus linguistics. *Anglistik: International Journal of English Studies*, 20 (1), pp. 41–67.

Lehto, A., **Baron, A.**, Ratia, M. and Rayson, P. (2010). Improving the precision of corpus methods: The standardized version of Early Modern English Medical Texts. In Taavitsainen, I. and Pahta, P. (eds.) *Early Modern English Medical Texts: Corpus description and studies*, pp. 279–290. John Benjamins, Amsterdam.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents the research undertaken to build a software solution which can be used to normalise spelling variation, which, as will be shown, exists in large quantities in texts from the Early Modern English (EModE) period. It will be made evident through the research presented that this spelling variation has a negative effect on the accuracy of software used in the field of corpus linguistics. The research process will involve taking steps to better understand the characteristics of EModE text, particularly in terms of its orthography, and to also quantify the precise effect historical spelling variation has on corpus linguistic methods. This will aid the development of an interactive tool which will utilise techniques from modern spell checking to normalise EModE spelling variation. This introductory chapter will first give an overview of the problem in hand, then three central research questions will be introduced and discussed. The chapter will end with a description of the structure for the remainder of the thesis.

1.1 Problem Overview

The computer-aided analysis of natural language text through corpus linguistic techniques is a well established area of research. Automated methods have been developed for tasks such as analysing word frequency, finding key words or key word clusters, finding collocations, and annotating texts with additional levels of detail such as part-of-speech tags and semantic categories. The majority of studies within the field of corpus linguistics, and other forms of language analysis, have focused on the examination of modern 'clean' texts. Problems occur, however,

when using the same techniques with more 'noisy' texts containing considerable amounts of spelling variation; the problems being due to the fact that most automated methods rely upon consistent spelling.

Early Modern English (EModE) is the most recent period of the English language for which the general written word contained a large amount of spelling variation[1]. Spelling in English was not standardised until the 18th century, before which the spelling of a word could change depending upon the author, scribe or publisher – as well as numerous other factors. It is thought that this spelling variation causes a considerable barrier to accurate and robust analysis in the field of historical corpus linguistics; however, very few studies have quantified the level of the problem. One study by Archer *et al.* (2003) evaluated the effect of spelling variation when semantically annotating texts from the 17th century, and found an increase in accuracy when spelling variation was partially normalised. The precise effect spelling variation has on other corpus linguistic methods needs to be established.

The EModE period (1500–1700) is of particular research interest due to it being the earliest period for which a relatively large corpus can be built, largely due to William Caxton's introduction of the printing press in 1476. The size and number of corpora available from the period has increased greatly over the last 20 years: from corpora such as the *Lampeter* corpus (Schmied, 1994) at 1.2 million words and the *Corpus of Early English Correspondence (CEEC)* (Nevalainen, 1997) at 5.1 million words to recent digitisation initiatives such as the Text Creation Partnership (TCP)'s transcriptions of 25,000 books from ProQuest's *Early English Books Online (EEBO)*[2], which is estimated to contain 1 billion words. Researchers can avoid the issue of spelling variation by utilising modernised versions of texts; Culpeper (2002), for example, used a modern edition in his study of Shakespeare's *Romeo and Juliet*. However, modernised versions of historical texts are more often not available, and moreover, researchers have questioned the value of some modernised versions, especially for Shakespeare's work (see e.g. de Grazia & Stallybrass, 1993). Another potential solution is

[1]Recent specific forms of language, such as SMS text messaging and Internet chatroom discussions, contain abnormally large amounts of inconsistent spelling for (mostly) different reasons.

[2]`http://www.lib.umich.edu/tcp/eebo/description.html`

to manually normalise the spelling variation within texts; this may be possible for small amounts of data but for studies using larger corpora of 1–5 million words, this would be restrictively time-consuming and with the very large datasets becoming available for research, such as EEBO, a fully manual approach would be clearly unworkable.

A solution is required which can be used to, at least partially, automatically normalise the spelling in EModE corpora, supplementing texts with modern equivalents which automatic corpus linguistic tools should recognise with greater ease. This pre-processing of texts should result in the corpus linguistic analysis of the historical data being more accurate and robust. It is important to note from the outset why dealing with historical spelling variation is not equivalent to two well researched tasks, namely translation and spelling correction. Firstly, EModE spelling normalisation does not equate to a translation task whereby words are generally found in another fixed form, as between, for example, English and French[3]. The task is more like modern spellchecking in this respect whereby a word could be spelt in a variety of forms, some potentially being unique to a particular author, text or passage. Secondly, it is important to make the distinction that texts would not be 'corrected' per se; it is generally considered that there was no real notion of a 'correct spelling' during the EModE period, especially before the first dictionaries were published (e.g. Samuel Johnson's dictionary of 1755). Moreover, the original spelling forms found should be retained, or at least be easily retrievable, as the choice of spellings itself can be an important point of interest – a common criticism of modernised texts is their lack of authenticity, in that the original orthography is lost. The task is similar to annotation in this respect as information is to be added to the text, not taken away; the modern equivalent spellings are for the primary purpose of assisting automatic corpus linguistic tools.

Computer-based methods for dealing with spelling errors in modern texts have been the subject of research for over 40 years (Damerau, 1964; Kukich, 1992; Mitton, 2010). Techniques have been researched and developed to deal with problems such as word processing errors, Optical Character Recognition (OCR) post-processing, Information Retrieval (IR) with noisy texts and the normalisation of Computer-Mediated Communication (CMC) based texts (e.g.

[3]That is not to say that automatic translation of texts is an easy task; the research area of machine translation has received substantial focus itself (e.g. Dorr *et al.*, 1999).

SMS, emails, etc). However, limited research has been completed to develop techniques to deal with the spelling issues in historical texts, and even less for specifically EModE spelling variation. Therefore, investigation is required to establish whether modern spellchecking techniques can be applied to EModE spelling normalisation and how the characteristics of EModE spelling variation will dictate which methods are required and how these methods should be implemented.

1.2 Research Questions

The research presented in this thesis aims to address the problems caused by EModE spelling variation, particularly in terms of improving the accuracy of corpus linguistic tools when applied to historical corpora. This will centre around the development of a spelling normalisation tool, for which the applicability of methods from modern spellchecking will be investigated. The resulting tool will act as a pre-processor for corpus linguistic techniques. Research is required to assess the extent of the problems caused by spelling variation, discover how modern spellchecking techniques can be applied to EModE spelling variation and evaluate the effectiveness of the developed software. More formally, there are three principal research questions to be addressed:

RQ 1 *How extensive is Early Modern English spelling variation in terms of the levels of variation appearing in Early Modern English corpora and how large an impact does this spelling variation have on corpus linguistic methodology?*

RQ 2 *What are the characteristics of Early Modern English spelling variation and how will these affect the application of modern spellchecking techniques to historical spelling normalisation?*

RQ 3 *What levels of performance can the developed normalisation tool achieve with different levels of training, particularly in terms of precision and recall, when automatically normalising spelling variation in Early Modern English corpora?*

The first research question is designed to help better understand the problem in hand, particularly the scale of the effect spelling variation has. The hypothesis is that spelling variation will have a considerable detrimental effect on the accuracy of corpus linguistic tools. Presuming that in answering *RQ 1* we prove the correctness of this hypothesis, the need for the remainder of the research to be undertaken will be justified. The first stage in addressing this research question will be to look at the ratios between spelling variants and modern spellings in different Early Modern English texts. Obtaining these ratios will quantify how much work is required to normalise texts and, presuming that the hypothesis regarding spelling variation having an impact holds, it naturally follows that more spelling variation equates to a larger impact. This leads onto the second stage where the hypothesis will be directly tested by comparing corpus linguistic methods' results before and after normalisation, thus assessing the impact of spelling variation on corpus linguistic tools. The research to answer these questions should also be of wider interest to researchers working with EModE corpora; the results will assist in better understanding of the corpora being used and provide the ability to assess the risk of not normalising spelling variation.

RQ 2 is key to the development of a historical spelling normalisation tool. The literature contains many issues to consider and many potential solutions for modern spellchecking – these will be discussed in detail in Section 2.2. In order to narrow these down to the important issues and likely useful solutions for dealing with EModE spelling variation, research is required to better understand common features of the spellings used. The specific characteristics which require analysis will be presented in the background literature (Chapter 2), where they can be discussed in more detail. In order to perform the different analyses, spelling variants linked to their modern equivalents will be required. This is a problem because very few EModE texts exist containing both original spellings and normalised forms. The research presented in this thesis results in a tool which will alleviate this problem; therefore, during the incremental development of the software these spelling characteristics are studied in tandem to help make decisions for the next stage of development. Again, these results should be of interest to the wider research community for better understanding of historical spelling trends and how these compare to other forms of spelling errors and variation.

The final research question, *RQ 3*, can be answered through the evaluation of the developed normalisation tool. It is important for any solution to be able to deal with a large proportion of the spelling variation present in a given text (i.e. recall[4]), thus having the desired effect of improving the accuracy of subsequently applied corpus linguistic tools. However, it is likely to be of greater importance that high precision[5] is maintained, in that any normalisations made are, in the vast majority of cases, correct. Failure to achieve high precision would introduce additional noise to the texts being normalised and potentially have a detrimental effect on the accuracy of corpus linguistic tools – this would clearly be unacceptable as the overall aim of the research is to improve historical corpus linguistic accuracy. Due to the importance of recall and precision here, and with the measures being widely used in Natural Language Processing evaluation (see e.g. Reynaert, 2008a), it seems sensible to evaluate the performance of the developed software in these terms. The developed software should be able to deal with spelling variation in different EModE corpora and a user should be able to customise and train the tool for a specific corpus. Therefore, the effectiveness of training the tool should also be evaluated in terms of its effect on both precision and recall.

The research questions stated here shall be referred to throughout this thesis in order to discuss further specifics, highlight their necessity and address the questions raised. This is emphasised in the following section, where the structure of the thesis is presented.

1.3 Structure of Thesis

There are four main chapters of this thesis in addition to the current Introduction and a final Conclusions chapter. Chapter 2 presents the background literature related to the research undertaken, which is split into two main areas. Firstly, the EModE period is introduced with particular focus falling on its inherent spelling variation and the application of corpus linguistics to EModE texts. Secondly, the field of modern spellchecking is discussed in detail, with various related issues highlighted and techniques for dealing with spelling problems considered. The

[4]Recall: Ratio of correct normalisations to normalisations necessary.

[5]Precision: Ratio of correct normalisations to normalisations made.

characteristics of different types of spelling errors that need to be investigated in order to address *RQ 2* will also be introduced. Finally, the chapter will include a critical analysis of specific previous research which has attempted to deal with the problem of historical spelling variation, both in English and other languages.

Whilst Chapter 2 will introduce some of the potential issues arising from EModE spelling variation, particularly in terms of its effects on corpus linguistic methodology, Chapter 3 will quantify the levels of variation in various EModE corpora and evaluate the precise impact of this variation on two automated and advanced corpus linguistic techniques; thus addressing *RQ 1*. Various analyses aimed at addressing *RQ 2* will also be described in Chapter 3, identifying and describing some specific characteristics of EModE spelling variation to take into consideration when developing an EModE spelling normalisation tool.

With an understanding of the size and properties of the problem in hand, Chapter 4 will describe how modern spellchecking techniques can be applied to EModE spelling normalisation. Particular focus will be given to how the characteristics of EModE spelling variation (as established in Chapter 3) influence the priorities and decisions made when choosing and adapting methods for detecting and normalising variants. The specific algorithms developed will be detailed and discussed before a description is given of a piece of software which utilises these methods in the form of a spelling normalisation tool.

Chapter 5 will evaluate the performance of the developed normalisation methods on EModE texts, hence addressing *RQ 3*. The developed normalisation tool's ability to both detect and automatically normalise EModE spelling variants shall be investigated in terms of precision and recall. A case study will focus specifically on the tool's performance in normalising a single EModE corpus. This will highlight the tool's effectiveness and usefulness in a real research project for which it is aimed.

The thesis will end with Chapter 6, the conclusions. Here, the thesis will be summarised and the research questions revisited in order to establish how successfully each has been addressed. The chapter will also explore the various contributions made and highlight any potential areas for future work following on from the research presented.

Chapter 2

Background and Related Work

The research presented in this thesis concerns two broad subjects. Firstly, Computer Science, due to the research centering around the development of an interactive piece of software which uses methodology well established in the area of Natural Language Processing. Secondly, Historical Corpus Linguistics, due to the overriding aim of the developed software being to aid corpus linguistic tools and methods in terms of accuracy when dealing with historical texts. The background presented in this chapter describes two specific areas from these broader subjects which are relevant to the research in question. In Section 2.1, the Early Modern period of the English language is introduced, with specific focus on the inherent spelling variation found in the period's language, the increasing number of corpora available from the period, and how spelling variation can affect historical corpus linguistic studies. The second branch of the background literature, presented in Section 2.2, focuses on methods used for both detecting and correcting spelling errors, such as those used in word processing software to deal with spelling and typing errors, but also research into dealing with problems caused by different types of spelling variation in other fields, such as Information Retrieval. Section 2.3 looks at the overlap between these two branches and the specific focus of this thesis; i.e. methods used to deal with historical spelling variation, both in English and in historical varieties of other languages. The chapter concludes with a summary of the presented background reading, establishing the characteristics of the two areas under investigation and providing motivation for the research undertaken.

2.1 Early Modern English

In texts describing the history of the English Language, four separate periods of the language are commonly described: Old English (500–1100 A.D.), Middle English (1100–1500), Early Modern English (1500–1700) and Modern English (1700 onwards)[1]. Many varying political and social factors have influenced English throughout its history, shaping the phonology, vocabulary, morphology, syntax and semantics of the language (see Singh, 2005). The Early Modern English (EModE) period is of significant importance for the study of the English language as it is the most influential in the formation of the standard modern English we use today. The period is also of particular interest in the field of corpus linguistics because it is the earliest period of the English language from which a reasonably large corpus can be constructed. This was largely due to a sharp increase in book production through the introduction of the printing press by William Caxton in 1476 and an increasingly literate public (Görlach, 1991: 6).

2.1.1 History

The actual dating of the EModE period is a topic of some contention; Görlach (1991: 9-11) dedicates a section to the subject. Barber (1997: 1) and Singh (2005) settle with a period of 1500 to 1700, however, there are reasons to consider adjusting these endpoints when compiling an EModE corpus. Various events in the 15th Century point towards the EModE period being dated pre-1500, including the invention of the printing press in 1476. There was also a significant reduction in regional differentiation between texts from around 1450 (Görlach, 1985: 1), this was accelerated by the printing press as 98% of all English books were printed in the London area (Görlach, 1991: 13).

Various authors (e.g. Fisher, 1977; 1984; 1992; Richardson, 1980; Samuels, 1963) discuss the importance of the 'Chancery Standard' in the development of written English. Between 1066 and 1417 all official correspondence in England was written in Latin or French (Fisher, 1984: 161), this despite the majority of the population actually speaking English. This is not actually an unusual situation, Fisher (1992: 1174) explains that similar situations have been exhibited as late as

[1]There is some disagreement on the precise dating of these historical periods. This shall be discussed in detail for the Early Modern English period in Section 2.1.1.

the 20th Century in Montreal, India and Norway where most of the population spoke the vernacular tongue yet official writings were largely in English, English and Danish respectively. Quite suddenly in August 1417, upon Henry V's second invasion of France, all of the King's correspondence were written in English rather than French or Latin, this 'commitment to the vernacular' was of significant importance, equivalent to Chaucer's commitment to English in the literary world in the 14th century (Richardson, 1980: 727). The official correspondence of Henry V was dealt with by the Chancery (a similar body to the English civil service of today, responsible for the King's administration), who, until the end of the fifteenth century, dealt with virtually all of the national bureaucracy of England (from the fifteenth century onwards the bureaucracy was departmentalised into various offices of the government (Fisher, 1977)). By the early 1430s the Chancery had developed its own 'standard' written English which more closely resembled today's modern Standard English than other texts from the period, such as personal letters. Due to the prestige and authority of any documents written by the Chancery and the need for a standardised form of English for official bodies, Chancery English became the most commonly accepted written standard and thus a forerunner to modern standard English (Richardson, 1980). A study by Fisher (1984) also shows that Chancery English greatly influenced Caxton and the written English produced by his printing press from 1476, and Shaklee (1980: 48) argues that Caxton "may have influenced the direction in which the language grew more than any other single man." Due to the King's commitment to written English and the importance of Chancery English, it could be argued that 1417 should mark the beginning of the EModE period.

The dating of the end of the EModE period is also a subject of debate. 1660 is a common date considered mainly for historical reasons, i.e. the end of the Civil War, but Görlach (1991: 11) states that by this time "Spelling has, more or less, become fixed in its modern form." Another date to be considered is the introduction of Samuel Johnson's dictionary in 1755, considered by many to be a milestone in the English language. Lass (1999: 1) considers 1776 to be significant as it was the year of the American Declaration of Independence; "the notional birth of the first (non-insular) extraterritorial English" (it is also conveniently 300 years after the introduction of the printing press by William Caxton). However, 1700 seems to be the preferred choice, Görlach (1991: 11,38-40) explains that by

this date Latin had all but been replaced with English in writing and speech, and that the language had achieved "considerable homogeneity," with regional (written) dialect differences no longer present and "the period of co-existing variants, so typical of all levels of EModE, being over by 1700."

An actual definition of the dates EModE represents is not necessarily required here as it would be foolish to rule out the study of a text simply due to its age or which period of the English Language it is considered to originate from. Besides, there is an inevitable overlap in the characteristics of adjacent periods, indeed Barber (1997) states, "All such divisions are arbitrary, for linguistic change is continuous." Clearly, maximising the period in which texts can be studied from is preferential; however, the lack of texts prior to the advent of the printing press makes large-scale corpus study difficult. Another problem which would be encountered in earlier texts (Middle English) would be characters outside the standard alphabet, this does not include foreign accents as are found in other European languages – these should be expected due to borrowings from other languages – but medieval characters such as ȝ (yogh) or þ (thorn) could potentially cause problems in terms of encoding and how to transform these to modern equivalents.

2.1.2 Spelling Variation

The English Language was under significant change throughout the EModE period, one reason being that Latin and French were rapidly being replaced by English as the preferred choice of language for print and speech for many institutions and individuals (see Singh, 2005: 140-147). For the language to obtain credibility the need for standardisation became apparent as the language "lacked obligatory rules of spelling, pronunciation, morphology and syntax" (Görlach, 1991: 36), the 'Chancery Standard' went some way to achieving this in some official texts of the period leading up to the introduction of the printing press in 1476, however, in the majority of texts from the EModE period spelling variation remained a prominent feature. Some common spelling variant examples are shown in Table 2.1.

Variant	Modern Equivalent	Notes
“goodnesse”	“goodness”	‘e’ often added to end of words.
“brush’d”	“brushed”	Apostrophes often used instead of ‘e’.
“encrease”	“increase”	Vowels commonly interchanged.
“spels”	“spells”	Consonants often doubled or singled.
“deliuering”	“delivering”	‘u’ and ‘v’ often interchangeable.
“conuay’d”	“conveyed”	Many combinations of the above.

Table 2.1: *Examples of spelling variants found in Early Modern English*

Spelling variation was not solely between different authors, scribes, editors and printing houses, as one may expect; it is common to find a word spelt several different ways in the same text or even on the same page. The reasons for so many variant spellings in EModE texts are numerous; Vallins & Scragg (1965: 71) state:

> This freedom of choice in earlier spelling, strange as it seems to us, was in fact perfectly natural. Although individual compositors normally held to a single spelling of a word, they occasionally used variant spellings to ease justification of the lines, or a spelling against their own usage might creep in from the copy they were using.

Furthermore, texts were often written by numerous scribes who would sometimes use their own spelling preferences resulting in totally different spelling conventions from one page to the next. Another reason is that spelling tended to be influenced by the local dialect and so could differ between regions, this was especially the case earlier in the EModE period, before the spread of London and Chancery English was complete. A further point to note is that the language, and particularly the spelling, of the EModE period cannot be viewed as a single entity with texts from the beginning of the EModE period being similar to texts from the end of EModE period; Nevalainen (2006: 4-6), for example, compared the language from three selected texts dated around 1500, 1600 and 1700, highlighting significant differences in grammar and spelling. The reasons for the large amount of spelling variation in EModE texts are wide and varied, what is clear is that the

task of normalising spelling cannot be equated to a simple translation problem with words appearing in a different fixed form; words could be spelt in a variety of different ways and one EModE text may have spelling variations which might not be found anywhere else.

The eventual 'complete' standardisation of spelling was a slow process, Vallins & Scragg (1965: 65-71) discuss how texts throughout the EModE period and earlier have frequent spelling variants, however, by the end of the EModE period variant spellings were becoming less frequent, and by the 18th century printers were using a single spelling for most words, and the modern spelling system slowly became fixed (Vallins & Scragg, 1965: 71), this was signified by the introduction of dictionaries, especially that of Samuel Johnson's in 1755. Reduction in spelling variation in the 18th century was also shown by Schneider (2001); unrecognised word types (i.e. likely spelling variants) by the ENGCG wordclass tagger (Voutilainen & Heikkilä, 1993) decreased by nearly 10% from the period 1670-1709 to 1770-1799. However, as stated in *RQ 1* (Section 1.2), a full quantitative analysis of the levels of spelling variation over the whole EModE period needs to be established.

It should be noted that spelling has still not stabilised completely in Present Day English, and variation still occurs. Vallins & Scragg (1965: 150-183) dedicate an entire chapter to the subject of continued spelling variation, entitled "Style of the House," in which they point out common discrepancies between printers, authors and even dictionaries, for example:

- *-ise* and *-ize* being interchangeable, e.g. *criticise / criticize.*
- Mute *e* before suffix being optional, e.g. *judgement / judgment.*
- *ct* and *x* being interchangeable, e.g. *inflection / inflexion.*

These and similar issues also occur with the overlap of British and American spelling (see e.g. Hofland & Johansson, 1982; Shaw, 2008; Swan, 2005: 39-44). The joining of words with (or without) hyphens can also cause considerable variation between texts, Fowler (1926, cited by Vallins & Scragg, 1965: 178) states, "The chaos prevailing among writers or printers or both regarding the use of hyphens is discreditable to English education." The use of hyphens is still debated today; for example, around 16,000 words recently lost their hyphens in the new edition of the Shorter OED (Rabinovitch, 2007).

In recent times, technology and other factors have brought about further variations on standard written English, Sebba (2007) discusses society's view and the social reasoning and implications of the varying orthographies found in modern language. The orthography of Computer-Mediated Communication has received particular attention in recent research, particularly in terms of normalising texts (this will be given more attention in Section 2.2). Studies have looked at the orthography of SMS (Crystal, 2004; 2008; Shortis, 2007; Tagg *et al.*, 2010; Thurlow, 2003), Instant Messaging (Varnhagen *et al.*, 2009), chat rooms (Al-Sa'di & Hamdan, 2005; Driscoll, 2002), web pages (Ringlstetter *et al.*, 2006), weblogs (Tavosanis, 2007) and social networking sites (Shaw, 2008). Spelling variation is also prevalent in other varieties of English, such as children's writing (Perera, 1986; Pooley *et al.*, 2008; Smith *et al.*, 1998; Sofkova Hashemi, 2003), non-native written texts (Granger, 1998; Pravec, 2002) and dialectal variations (Anderwald & Szmrecsanyi, 2009; Trudgill, 1999; Trudgill & Chambers, 1991; Wales, 2000). However, the focus of this thesis is on spelling variation in the EModE period and particularly dealing with its effect on corpus linguistics, to which our attention now turns.

2.1.3 Corpus Linguistics and Early Modern English

A corpus is "a collection of naturally occurring language text, chosen to characterize a state or variety of a language" (Sinclair, 1991: 171) and corpus linguistics is the study of language through corpus-based research (McEnery & Wilson, 2001), which has become synonymous with using a computer to analyse a large body of text collected to represent a certain subject. Analysis performed on corpora may include:

- Simple string searching.
- Word frequency lists.
- Concordances – a list of the occurrences of a word with their immediate contexts to the left and right.
- Collocations – words which co-occur more often than would be expected by chance.
- Keywords – looking at which words are significantly more frequent in one text (or collection of texts) compared to another text (or collection of texts).

- Annotation – attaching extra information to words, sentences or other points in the document, or to the whole document. Two of the more common forms of corpus annotation are:
  - Part-of-Speech (POS) tagging – Each word is given a grammatical tag (verb, noun, adjective, etc), further analysis can then be made to find when and how certain grammatical classes are used (e.g. Garside & Smith, 1997).
  - Semantic tagging – Each word is given a semantic category tag which relates to a particular topic or concept (e.g. climate and weather conditions). As with POS tagging, these tags can then be used for further analysis, such as finding key semantic categories (e.g. Rayson, 2008).

Various pieces of software have been created to help perform these tasks automatically, including: *Wordsmith Tools* (Scott, 2004), *BNCweb* (Hoffmann *et al.*, 2008), *CQPweb* (Hardie, forthcoming) and *Wmatrix* (Rayson, 2008). All of the software listed can perform one or more of the analysis functions described above, but they are all designed to work with modern English (and in some cases other modern languages), problems occur when these tools and methods are used to analyse historical varieties or dialects of English (and other languages), especially when large amounts of spelling variation occurs – as in EModE.

The construction of EModE and other historical corpora has become an important focus of research; Kytö *et al.* (1994) state:

> In recent years, interest in the compilation of corpora containing texts from the earlier periods of English has increased rapidly, together with the development of new methods and aids for tagging and parsing the texts in these corpora. The number of computer-assisted studies of the history of English has soared and there are important major research projects making effective use of databases of early English.

Many historical English corpora have been created containing texts from the EModE period, these include the *Helsinki*, *ARCHER (A Representative Corpus of Historical English Registers)*, *Lampeter* and *Zurich English Newspapers (ZEN)* corpora (detailed in Kytö *et al.*, 1994), the *Corpus of Early English*

Correspondence (CEEC) (Nevalainen, 1997), the *Corpus of English Dialogues (CED)* (Culpeper & Kytö, 1997), the *Early Modern English Medical Texts (EMEMT)* corpus (Taavitsainen & Pahta, 1997; 2010), the *Innsbruck Letters* corpus - part of the *Innsbruck Computer-Archive of Machine-Readable English Texts (ICAMET)* corpus (Markus, 1999) and also many different versions of Shakespeare's works, for example, the *First Folio* as printed in 1623, which can be sourced from the Oxford Text Archive[2]. Whilst these corpora are relatively modest in size, compared to large modern corpora such as the *British National Corpus (BNC)* at 100 million words (Burnard, 2007) and the *Corpus of Contemporary American English (COCA)* at 400 million words (Davies, 2008), large amounts of text from the period are being digitised through various ongoing initiatives. One such project undertaken by the Text Creation Partnership (TCP)[3] has transcribed 25,000 books from ProQuest's *Early English Books Online (EEBO)*[4]. The full EEBO collection contains digital facsimiles of virtually every English printed work between 1473 and 1700; nearly 125,000 works. Whilst these digital facsimiles are a very useful resource, the TCP's ASCII SGML transcriptions will allow researchers to search and perform corpus linguistic functions on a much larger dataset than previously available for the EModE period. Further textual data from the period are being digitised through other schemes including newspapers by the British Library[5], and books by the Open Content Alliance[6] and Google Book Search[7]. A summary of the main corpora available containing EModE texts is given in Table 2.2.

[2] `http://ota.ahds.ac.uk`
[3] `http://www.lib.umich.edu/tcp/eebo/description.html`
[4] `http://eebo.chadwyck.com/home`
[5] `http://www.bl.uk/reshelp/findhelprestype/news/earlyenglishnews/`
[6] `http://www.opencontentalliance.org`
[7] `http://books.google.com`

Corpus	Genre & Type	Period	Size (words)[a]
ARCHER	General/Mixed	1650–1990	1.7 Million
CED	Speech-related	1560–1760	1.2 Million
CEEC	Letters	1400–1800	5.1 Million
EEBO	General/Mixed	1473–1700	c. 1 Billion[b]
EMEMT	Medical texts	1500–1700	2 Million
Helsinki	General/Mixed	730–1710	1.6 Million
Innsbruck	Letters	1386–1688	170,000
Lampeter	Various tracts and pamphlets	1640-1740	1.2 Million
Shakespeare First Folio	Plays	c. 1590–1613	800,000
ZEN	Newspapers	1661–1791	1.6 Million

[a] The sizes given here are for the whole corpus, some of which may be outside the EModE period.

[b] An earlier version of EEBO containing roughly half (12,268) of the works was analysed and found to contain over 500 million words, the full version of EEBO is likely to contain around double this.

Table 2.2: *Summary of the main Early Modern English corpora*

Whilst the corpora described provide sources for a wide-range of research of the EModE period, the spelling variation prevalent within these texts (as described in Section 2.1.2) creates a barrier to accurate and meaningful corpus linguistic results; Culpeper (2007: 69) states, "Early Modern English spelling variation has been perhaps the major stumbling block for historical corpus linguistics." In some cases, it may be possible to avoid this problem by using modernised versions of the text; Culpeper (2002), for example, utilised a modern edition of *Romeo and Juliet* in his study of the Shakespeare play. However, modernised versions of texts are often unavailable and with the increasing size of EModE corpora being released (see Table 2.2), the proportion of texts with modernised equivalents is diminishing further. The effect of ignoring the issue of spelling variation in corpus linguistic research will now be outlined, starting with simple functions which produce a cumulative effect on more complicated corpus linguistic techniques.

Searching for a word is possibly the most basic corpus linguistic function, and the basis for the majority of more complicated techniques. However, searching in an EModE corpus can be problematic; using a simple search algorithm would only return the occurrences of the word when it is spelt exactly the same as the search query – spelling variants of a word would not be returned. One option is to search for both the word and its variants, however, it is often difficult to know all of the possible spelling variants for a word and the lists can be very long, substantially increasing processing time. To demonstrate this, a relatively simple word such as *would* could be spelt in a variety of forms, including: *would*, *wolde*, *woolde*, *wuld*, *wulde*, *wud*, *wald*, *vvould*, *vvold*, and so on. To return all occurrences of the lemma *would*, the user would first need to know all possible spelling variants of the word and then either search for these in turn or build a complicated search query. This problem can occur even when looking at small portions of text, or from one author; for example, Vallins & Scragg (1965: 70-71) showed that it was frequent within EModE texts for words to be spelt differently even in a short paragraph, exemplified in this short passage from the *Authorised Version of the Bible (1611)* (cited by Vallins & Scragg, 1965: 67):

> Though I speake with the tongues of men & of Angels, and haue not charity, I am become as sounding brasse or a tinkling cymbal. And though I haue the gift of prophesie, and vnderstand all mysteries and all knowledge: and though I haue all faith, so that I could remooue mountaines, and haue no charitie, I am nothing[...]

Linked to the above, creating a simple word frequency list can also become difficult as each of the different spellings of a word would be listed as a separate entry, with the actual frequency split between the entries. This may not be as much of an issue with words at the top of typical word frequency lists; two recent studies by Lieberman *et al.* (2007) and Pagel *et al.* (2007) have shown that more frequent words are less likely to change over time, and are in the majority of cases spelt the same now as they have been since Middle and even Old English. However, further down the frequency list, results would clearly be inaccurate as in the case of *would* and *charity* given above. Furthermore, building a list of concordances or collocations would be affected in similar ways; concordance lists would be incomplete if all spelling variants of a word were not searched for, whilst

collocations would be incomplete and frequencies inaccurate as the search term and/or its collocates could have multiple spelling forms.

Keyword lists can be used, for example, to analyse the 'overuse' or 'underuse' of words when comparing one set of texts to another (see e.g. Granger & Rayson, 1998). As keywords are found by comparing frequency lists (Tribble, 2000) the obvious inaccuracies in frequencies are likely to have a cumulative effect on keyword analysis. This problem is potentially intensified when evaluating key word-clusters (e.g. Mahlberg, 2007), as even very low frequency word-clusters could be considered key, but if any one of the words within a particular cluster are spelt in different ways throughout a text or corpus the frequency of that cluster will be reduced. The comparison of frequencies can be performed using a variety of statistical methods with varying degrees of complexity. Some common statistics used include the Yule Coefficient (Yule, 1944), the χ^2 (chi-squared) test (Pearson, 1904) and the log-likelihood ratio (Dunning, 1993). Due to the complexity of these statistics, it is difficult to estimate the level of impact inaccurate frequencies will have on keyword results without first resolving these inaccuracies and comparing the results before and after; i.e. for EModE, one could normalise spelling and compare keyword results from the original texts and the normalised texts.

Part-of-Speech (POS) tagging is perhaps the most common form of corpus annotation, many methods for which have been developed, including manual and automatic techniques. Automatic POS tagging of English text is possible by using well-defined rules of the language (for example, words ending *-ness* are likely to be nouns), amongst other techniques. However, these methods are based on modern English and problems are encountered when dealing with variations of the language, e.g. EModE. The CLAWS POS tagger (Garside, 1987; Garside & Smith, 1997), for example, uses a dictionary which includes words (or multi-word units) and suffixes with their possible parts of speech. This dictionary is based upon modern English and does not include the large amount of spelling variants (as previously discussed) and the archaic / obsolete words found in EModE texts. CLAWS also uses a probabilistic Hidden Markov Model (e.g. the likelihood that an adjective will be followed by a noun) to disambiguate words which could potentially be several different parts-of-speech. Similarly, these probabilities are based on modern English and may not apply to EModE; there are definite differences in the grammar of Present-day English and EModE, as

discussed by Kytö & Voutilainen (1995) in their application of the ENGCG Parser to the previously mentioned Helsinki Corpus.

Semantic annotation can also be assigned automatically, but like POS tagging, it is expected that accuracy suffers when dealing with the characteristics of EModE. One example of an automatic semantic tagger is the UCREL Semantic Analysis System (USAS) (Rayson *et al.*, 2004), again this has been developed for processing modern English. USAS uses CLAWS POS tagged text as its input, hence, with the characteristics of EModE likely to produce inaccuracies in POS tagging, it is likely that these inaccuracies will be passed down through USAS. Also, USAS relies much more heavily on a large dictionary than CLAWS, and cannot guess the semantic field of a word from its immediate neighbours or from other surface clues (e.g. the *-ness* indicates noun rule in CLAWS); this obviously will cause problems when words which are not in the dictionary are encountered – which, as previously shown, is common in EModE texts. Another point to consider is the possibility of a semantic shift in words from EModE to present-day English. Archer *et al.* (2003) discuss extending USAS for EModE texts, the paper reports on an evaluation performed on relatively contemporary texts from 1640, later texts were used to avoid the effect of semantic shift and the differences in grammar. Error rates were quite low at 2.9% for one text, and 4.0% for another (the authors indicate that error rates in older texts would be higher), however, dealing in part with spelling variation produced a reduction in error rates to 1.2% and 1.4% respectively. It is, therefore, clear that spelling variation is causing inaccuracies in the semantic annotation.

It has been shown here that spelling variation in EModE will clearly produce boundaries in the accuracy and meaningfulness of results from corpus based research, resulting, for instance, in a lack of annotated EModE corpora; Markus (2002) states "[f]or studies of the Middle and Early Modern English periods [...], the lack of tagged versions of the corpora concerned has caused considerable problems of retrieval." It is clear that the spelling variation can have an effect on even the most basic techniques of corpus linguistics, such as searching for words or building frequency lists. A subsequent effect on performance of more complicated corpus linguistic methods is expected as they rely upon the accuracy of these basic techniques. Archer *et al.* (2003) found this to be the case when attempting to semantically tag even relatively contemporary EModE texts. However, as stated

in *RQ 1* (Section 1.2), a quantification of the level of the effect of significant amounts of spelling variation on other corpus linguistic techniques would be desirable, although the exact extent of the problem cannot be quantified without first normalising an EModE corpus and comparing results before and after the normalisation. The focus of this background chapter now turns to a review of the techniques which may help to address the issue of spelling variation.

2.2 Spelling Error Detection and Correction

The use of a computer to assist in the detection and correction of spelling errors has been the subject of extensive research for over 40 years (e.g. Damerau, 1964; Jurafsky & Martin, 2000; Kukich, 1992; Mitton, 1996; 2010) give comprehensive surveys of the subject literature. Virtually all of the literature focuses upon dealing with modern (largely English) texts, a review of the techniques used will be given here. Kukich (1992) separates dealing with spelling errors into "three increasingly broader problems." These being: (1) Non-word error detection, (2) Isolated-word error correction, and (3) Context-dependent word correction. For this chapter's review of the surrounding literature, the same three problems, and their solutions, shall be analysed following an overview of the tasks and processes which are affected by spelling issues.

2.2.1 Spelling Error Tools and Applications

Spelling is a potential problem for a variety of different tasks, each with characteristics which require individual tailoring of spell checking and correction techniques. The most commonly known variety of spell checking and correction occurs within modern word processing software, such as Microsoft Word. Within these programs the text inputted by the user is analysed and any words which the systems deems to be invalid or misspelt are highlighted as such. Usually, the user has the option to view a list of potential replacements for a highlighted word, one of which is then chosen by the user to be the correct word and the system replaces the highlighted word with the correction. Recently, similar functionality has appeared on the web; for instance, the Mozilla Firefox browser[8] now has

[8] `http://www.mozilla.com/firefox`

in-line spell checking for web forms, and 'gadgets' such as Ask A Word[9] can be added to 'personalised start pages' such as iGoogle[10] and netvibes[11], these take inputted text, process it and highlight spelling errors and offer replacements in much the same way as Microsoft Word. Another well known spell checking and correction tool is Aspell[12] which claims to be superior at suggesting replacements for misspelt words; Aspell is aimed at eventually replacing the popular, although now dated, ispell[13] for UNIX.

Spelling problems are not limited to word processing and other similar user typing tasks, problems caused by spelling occur in a variety of fields. Information Retrieval (IR) is one such field; when searching a database the query string may be similar, but not identical, to desired matches, for such problems approximate string matching or 'fuzzy matching' is often used (see Zobel & Dart, 1995). This problem particularly occurs in name matching when a user may not know the 'correct' spelling of a particular name or the name might have many potential variant forms, also errors may have been made during data entry into the database. A similar problem exists in genealogy; for example, searching records for a family name where the name may have changed over time and/or been inputted into a record erroneously. Various studies (Cohen *et al.*, 2003; Järvelin *et al.*, 2007; Pfeifer *et al.*, 1996; Wu & Manber, 1992; Zobel & Dart, 1995; 1996) have focussed on dealing with this problem, most involve supplementing the query string with a list of variations in order to increase the recall from the database. A similar Information Retrieval problem is that of web queries, i.e. in search engines such as Google[14], Yahoo[15] and Microsoft's Bing Search[16]. In recent years all three of these search engines have added functionality to deal with misspelled query strings, e.g. if Google is used to search for the term *varient spelling*, a link is provided stating "did you mean *variant spelling*." Cucerzan & Brill (2004) discuss a novel method for this task which involves using the frequency of previous web query strings to determine if the current given query string is valid or has a misspelling. Also,

[9] `http://www.askaword.com`
[10] `http://www.google.com/ig`
[11] `http://www.netvibes.com`
[12] `http://aspell.net`
[13] `http://lasr.cs.ucla.edu/geoff/ispell.html`
[14] `http://www.google.co.uk`
[15] `http://www.yahoo.com`
[16] `http://www.bing.com`

Martins & Silva (2004) evaluate a system using common spell checking techniques to correct misspelled search queries in their Portuguese search engine. Finally, and of particular relevance to the research reported in this thesis, recent studies have centred around dealing with the spelling variation in historical textual databases, particularly in German (Hauser *et al.*, 2007; Pilz *et al.*, 2006; 2008), and also less recently with English (Robertson & Willett, 1993) – these will be discussed further in Section 2.3.

As well as adapting query strings for IR with noisy texts (i.e. containing spelling variation), many studies have looked at normalising the spelling within the documents themselves to aid IR and corpus linguistic techniques (as discussed in Section 2.1.3). Such studies include the post-correction of texts scanned through Optical Character Recognition (OCR) (e.g. Lopresti, 2008; Reynaert, 2008b; Ringlstetter *et al.*, 2007a; Strohmaier *et al.*, 2003a; Taghva & Stofsky, 2001) and handwriting recognition (e.g. Bhardwaj *et al.*, 2008; Pittman, 2007). Recent focus has fallen upon Computer-Mediated Communication (CMC) spelling variation; studies have looked into applying spelling correction techniques described in this section to normalise various forms of CMC, including: SMS (Acharyya *et al.*, 2009; Aw *et al.*, 2006; Choudhury *et al.*, 2007; Cook & Stevenson, 2009; Kobus *et al.*, 2008; Tagg *et al.*, 2010; Yvon, 2010), chat (Wong *et al.*, 2006; 2008), emails (Agarwal *et al.*, 2007; Sproat *et al.*, 2001) and newsgroups (Agarwal *et al.*, 2007; Clark, 2003; Zhu *et al.*, 2007).

The spelling problems which affect each of these tasks have different characteristics; for example, for word processing and information retrieval, spelling errors may be largely due to typographical errors (i.e. slips during typing resulting in the incorrect insertion of, for example, a *q* instead of a *w* due to their proximity on the keyboard) or due to common human misspellings (e.g. *wierd* instead of *weird*), OCR errors may be largely due to the graphical similarity of letters (e.g. *D* / *O* and *rn* / *m*), and name-matching errors may be due to phonetic similarities (e.g. *Bayer* / *Beyer* / *Baier*). The methods required to detect and correct these errors will need adjusting depending on these specific characteristics, this section now turns to discussing these methods.

2.2.2 Non-word Error Detection

Dictionary Lookup

The first stage of a modern spell checker is to detect spelling mistakes within the text; this is normally done by looking up each of the text's words in a dictionary (Mitton, 1996: 93-95). The actual construction of a dictionary for use in a spell checker can be difficult in itself. Firstly, the size of the dictionary is an issue; if the dictionary is too small then many words which one would deem valid would not be found in the dictionary and thus presumed by the system to be incorrect. This is more of an inconvenience to the user than a serious issue in traditional spell checking; the user normally has the option to disregard the word and mark it as correct, and in the majority of cases can add the word to the dictionary for future reference by the system. A more serious problem is when the system flags a word as correct when it is in fact a spelling error (Mitton, 1996: 95); this problem is intensified with larger dictionaries that contain less frequent words which are potential misspellings of other words, e.g. if *veery*, a type of bird, was included in the dictionary it is easy to see that *very* could be misspelt to produce the string *veery* which would be flagged as correct by the system. These problematic spellings, known as *real-word errors*, are discussed in more detail in Section 2.2.4.

The size of the dictionary is also an issue due to space and processing time constraints, too large a dictionary could make the list unmanageable and too time-consuming to search. Due to each word type in a text requiring lookup within the dictionary it is important to make this process as quick as possible. Various techniques have been developed for quick-searching of a dictionary, the most common of which is a hash table (Knuth, 1973). The main advantage of hash tables is the reduction in comparisons needed, however, it is necessary to "devise a clever hash function that avoids collisions without requiring a huge hash table," this problem has caused spelling mistakes to go undetected because they happen to have the same hash address of a valid word (Kukich, 1992). Other dictionary search techniques include Tries (Pittman, 2007) and frequency ordered binary search trees (Knuth, 1973), as well as finite-state automata (Aho & Corasick, 1975) and efforts have been made to reduce search times and storage space by partitioning the dictionary based on frequency levels, i.e. a table or tree of the most common words are searched first (Peterson, 1980) and also reducing the

size of a dictionary by only including the root form of words (Kukich, 1992). It should be noted that speed and space constraints have become less critical recently due to the increases in computational capability (Kukich, 1992), i.e. via Moore's Law (Moore, 1965). One interesting idea is to derive a specialised dictionary for the text being analysed based on its topic; Strohmaier *et al.* (2003b) introduce research which uses the web as a source for a "dynamic" dictionary, using targeted search queries.

A group of words which are relatively uncommon in dictionaries used for spell checking are proper nouns. Due to an infinite amount of potential proper nouns which could be found within a text it is not sensible to try and list them all (although adding more frequent proper nouns is a sensible first step). A common-sense approach to the problem would be to use the rule that proper nouns always begin with a capital letter in Modern English; this, however, does not work in all cases as a capital letter is also used to signify the start of a sentence. The problem is even worse in EModE; Osselton (1998) explains how between 1550 and 1750 there was a distinct increase in the use of a capital letter to begin nouns where one would not be present in Modern English. This particular problem with EModE texts rules out using the appearance of a capital letter with any reliability to distinguish proper nouns in an application for detecting EModE spelling variation.

N-Gram Analysis

Another less common method for discovering spelling errors is the analysis of the letter n-grams which make up the word. N-grams are subsequences or portions of a word of n length, n is usually 1, 2 or 3 (uni-, di-/bi- or tri- grams respectively). Analysis of the n-grams contained in words from a dictionary or large corpus can produce binary or frequency matrices; for a binary matrix, 1 is recorded for an n-gram found during the analysis at co-ordinates relative to the letters of the n-gram, for a frequency matrix the number of occurrences or the statistical probability is stored. More information can be captured by also including a dimension which indicates the position of the n-gram. Checking the validity of a given word involves checking each n-gram against the stored matrix to see if any n-grams are invalid or very infrequent, indicating a spelling error or likely source of one; for example, one would expect the bigram *QM* or the trigram *SHJ* to be invalid. Even unigrams

can be useful when positional information is included, for example a word ending in *Q* is unlikely in English.

N-gram analysis negates the necessity to include a dictionary, which, as previously discussed, can cause some difficulties. However, problems arise because a word may have valid n-grams but still be a spelling error, resulting in the error being missed. The process can still be useful, particularly with OCR, where errors usually result in invalid n-grams; Hanson *et al.* (1976) used positional binary trigram arrays on OCR output and found 98% of the errors present. However, Zamora *et al.* (1981) used trigram frequency statistics for general spell checking and found that "although trigram analysis was able to determine the error site within a misspelled word accurately, it did not distinguish effectively between valid words and misspellings" (Kukich, 1992).

2.2.3 Isolated Error Correction

In the previous section, methods were described for finding non-word errors within a text. Early spell checking programs, such as UNIX spell, stopped at this stage, simply presenting the strings which it considered to be spelling errors to the user. More recent spell checkers, as discussed in Section 2.2.1, go further by offering suggestions to the user of what the intended word might be. Various methods have been developed to search for these alternatives, usually including finding similar strings, by some metric, from a dictionary. The word 'isolated' distinguishes between looking at each word as an individual entity and the more complex task of also considering the surrounding context, as described in Section 2.2.4.

Phonetic Matching Algorithms

Phonetic matching has been used for decades to identify strings which have a similar sound when spoken, regardless of their spelling. They are used frequently when searching for a name in a database (see Pfeifer *et al.*, 1996; Zobel & Dart, 1996). The most familiar phonetic matching algorithm is *Soundex*, first patented in 1918 (Russell, 1918; 1922). The basic Soundex algorithm takes the following steps:

1. Replace all but the first letter with the appropriate digit listed below:

   (0) A, E, I, O, U, H, W, Y

   (1) B, F, P, V

   (2) C, G, J, K, Q, S, X, Z

   (3) D, T

   (4) L

   (5) M, N

   (6) R

2. Remove any pairs of digits that are the same and occur next to each other in the string.

3. Remove all occurrences of the digit 0.

4. The Soundex code is the first 4 letters of the remaining string (the string is padded with zeroes if the string length is less than 4).

The algorithm results in a code of length 4, which is the string's first letter, and three numbers representing the next three consonants (apart from H, W or Y). This can be very useful, as an example, the string *disapont* (a possible spelling variant) would have the Soundex code *D215*, the real word *disappoint* also has this code. One of the problems with Soundex is the number of false positives which occur; as well as *disappoint* many other real words have the code *D215*, including *dispense*, *deceiving* and *despond*, which are all obviously incorrect to a human reader. Soundex will also not always match two words which are similar sounding; for instance, *increase* has the code *I526*, but *encrease* has the code *E526*. This could be considered a slightly contrived example as it has been found that misspelling of the first letter is quite rare (Yannakoudakis & Fawthrop, 1983b). Whether this applies to EModE spelling variation has not been researched; hence, quantification of this feature and other EModE spelling variation characteristics is required to better understand how the methods described in the section could be applied to EModE texts. Regardless, it is clear that Soundex in isolation is not enough to find the correct replacements for misspellings without also considering a large number of false positives.

The Soundex algorithm is still used today in its basic form, although many extensions of the algorithm have been proposed and developed; yet, still virtually all phonetic spelling correction algorithms have the same basic structure with changes often only in the form of pre-processing, varying the code length or changing the letter-groupings. A well known Soundex variant is Phonix (Gadd, 1988; 1990; Pfeifer *et al.*, 1996), the same basic Soundex algorithm is present although slightly different letter groupings are used. The main difference in Phonix is the pre-processing of the string with 160 letter transformation rules such as *ecs* mapped to *x* and *gn*, *ghn* and *gne* mapped to *n*, these transformations are designed to increase the accuracy of the letter grouping stage. One issue with Phonix is the processing time required, this is due to each of the transformation rules being applied in turn (Zobel & Dart, 1995). Other Soundex variations include Phonetex (Hodge & Austin, 2001a) which is similar to Phonix but with different letter groupings and transformation rules, and Editex (Hodge & Austin, 2001a; Zobel & Dart, 1996) which combines Soundex and Phonix properties with Edit Distance measures – described later in this section.

Various papers have evaluated the effectiveness of phonetic matching algorithms, including Hodge & Austin (2001a); Pfeifer *et al.* (1996); Zobel & Dart (1995; 1996). Zobel & Dart (1995; 1996) found that Soundex and Phonix are inferior to other techniques such as Edit Distance (see Section 2.2.3) in name matching, which is surprising considering that name matching was the original aim of both algorithms. Hodge & Austin (2001a) tested Soundex, Editex and Phonetex on 360 phonetic misspellings, comparing them to Agrep (Wu & Manber, 1992), Microsoft Word 97 & 2000 and ispell. They found that recall was over 90% for all three algorithms, Phonetex scoring highest with 98%, then Soundex with 92% and Editex with 90%, for comparison Microsoft Word 97 scored 94%, Microsoft Word 2000 scored 96%, Agrep 87% and ispell only 69%. However, they also found that a large number of false positives were also present; Phonetex finding the correct replacement being 1 of 11.1 potential replacements, Soundex a 20.83:1 ratio and Editex a 1.43:1 ratio (Editex is not just using phonetic matching techniques however). This shows that high recall is at the cost of low precision; hence why phonetic matching algorithms are rarely used in isolation for spelling correction.

Rule Based Approaches

In certain spellchecking applications there exist groups of letters which are interchangeable, i.e. one letter (or sequence of letters) is commonly misplaced for another. One application in which this feature is abundant is OCR; for example, it is more likely that an *e* will be mistaken for an *o* than a *t* be mistaken for an *m* (Mitton, 1996: 105). A similar feature occurs in typed texts where the proximity of letters on the keyboard leads to common mistakes; for instance, *and* mistyped *anf* and *are* mistyped *arte* (Mitton, 1996: 77-92; Yannakoudakis & Fawthrop, 1983b). There are also some common misspellings by writers, e.g. *recieve* instead of *receive* or *definate* instead of *definite*; Mitton (1996: 54-76) gives a thorough analysis of these errors and Vallins & Scragg (1965: 150-183) describes some common discrepancies in spelling between different authors, printers and dictionaries.

The knowledge of these common letter errors can be exploited when correcting word-errors found in the text. A set of letter replacement rules can be derived which account for common misreading (for OCR) or mistyping (either typing error or the user is unsure of the correct spelling) of letters; for example, substitute *i* for *j* (OCR), substitute *f* for *d* (typing), or substitute *ie* for *ei* (human spelling misconception). These rules can then be applied to any non-word found in the text to produce a list of potential candidate replacement strings which can in turn be filtered to find potential dictionary word replacements. The rules could also have probabilities attached (see e.g. Mitton, 1996; Yannakoudakis & Fawthrop, 1983a) so that candidate replacements produced by more common letter errors are ranked higher for replacement when presented to the user. This concept is a major component of the much cited 'noisy channel' method of spelling correction (see Brill & Moore, 2000; Church & Gale, 1991; Kernighan *et al.*, 1990), where the letter replacement rule probabilities are used in the 'error' or 'channel model'.

Much of the above may also apply to EModE texts; many texts are OCR scanned so typical OCR errors are likely to be present, and the rest are manually transcribed so common typing errors may be present in these texts. However, the characteristics of EModE texts (see Section 2.1.2) dictate that some spelling errors are specific to the period. There has been very limited research into the frequency of specific EModE letter replacements, although some authors have listed common variations; Fisher (1977), for example, states some commonly interchangeable

pairs: *i / y*, *u / v* and *ou / ow*, as well as the inconsistency in the presence or absence of a final *e*.

String Similarity Measures

A further spelling correction technique involves computing the similarity between two strings, the most common forms of this measure are the minimum edit distance algorithms that have been around since Damerau (1964) and Levenshtein (1966) introduced their algorithms over 40 years ago. The algorithms compute the similarity of two strings, counting the minimum number of insertions, deletions and substitutions required to transform one string into the other. Many uses of these algorithms have been researched, particularly Levenshtein distance; applications range from DNA analysis to studies of bird song (Kruskal, 1983).

Searching for words with a minimum edit distance of 1 from a misspelling has been used as a method to find candidate spelling corrections in a number of studies (see e.g. Church & Gale, 1991; Mays *et al.*, 1991). This has been shown to be useful with early studies finding that in the majority of cases a misspelling is only 1 edit away from the intended correct word. The actual coverage of this rule varies from study to study: Damerau (1964) first calculated that approximately 80% of spelling errors in his study were 1 edit away from the original, however, in a study of scientific and scholarly texts Pollock & Zamora (1984) found this figure to be much higher at 94%, conversely, Mitton (1987) found that only 69% of the spelling mistakes in a corpus of spelling from a range of sources contained only one edit. For EModE spelling variation, it is difficult to know what the corresponding percentage would be; this should be investigated along with other characteristics of EModE spelling variation, as stated in *RQ 2* (Section 1.2).

Edit Distance algorithms are generally computationally expensive if used to check a word against a large dictionary; however, they can be useful for comparing a string to a subset of the dictionary found by another string analysis method (e.g. Mitton, 2008; Zobel & Dart, 1995). An application of the Levenshtein Distance algorithm on a subset of Soundex matches of the string *disapont* (see page 27) is shown in Table 2.3. This example shows that the correct replacement *disappoint* has the smallest distance, so clearly, in this example, the algorithm is useful for deciding which replacement is most likely to be correct.

Phonetic Match	Distance	Edits
disapont → *disappoint*	2	Insert *p* Insert *i*
disapont → *dispense*	4	Delete *a* Substitute *o* → *e* Substitute *t* → *s* Insert *e*
disapont → *deceiving*	7	Substitute *i* → *e* Substitute *s* → *c* Substitute *a* → *e* Insert *i* Substitute *p* → *v* Substitute *o* → *i* Substitute *t* → *g*
disapont → *despond*	3	Substitute *i* → *e* Delete *a* Substitute *t* → *d*

Table 2.3: *An example application of Levenshtein Distance*

As shown, Levenshtein's algorithm returns a count of the minimum edits needed to convert one string to another, many of the extensions of the algorithm return a similar figure. Some, such as the Needleman-Wunsch distance algorithm (Needleman & Wunsch, 1970), designed to search for similarities in amino acid sequences, assign different values to different edit operations; Levenshtein distance is equivalent to the Needleman-Wunsch algorithm with each edit operation having a value of 1. An efficient method to find all words from a dictionary within a small number of edits from a given string has been presented (Mihov & Schulz, 2004; Mihov *et al.*, 2007), and extended with "symbol dependent edit weights" (Ringlstetter *et al.*, 2007b), much like the rule-based approach discussed above.

Another technique for computing string similarity is n-gram analysis; this involves splitting the two strings being compared into arrays of substrings of length n and then, on a basic level, counting how many n-grams occur in both strings. An example of the use of this method is described in a proposed spell checking system by Hodge & Austin (2001b) where unigrams are used for spellings with less

than 4 characters, digrams for 4-6 characters and trigrams for spellings with more than 6 characters. Their hybrid approach (they also used phonetic matching) produced a recall rate of 93.9% (precision rates were not given). It is important to take into account the length of words; e.g. using a simple count of n-grams, *water* will match as many n-grams with *water* as it will with *waterline* (Zobel & Dart, 1995). Other spelling correction studies which use n-gram techniques include Järvelin *et al.* (2007) who used digrams of adjacent and non-adjacent letters (which they called s-grams) for information retrieval, and Robertson & Willett (1993) who used n-grams to find words in a dictionary similar to a word from a historical database (texts from 16th, 17th and 18th centuries) search query (see Section 2.3.1 for more details).

2.2.4 Context Sensitive Error Detection and Correction

The previous sections have described searching for non-word errors and then correcting them in isolation. On the surface this may seem sufficient to correct all of the spelling mistakes in a given text; however, problems occur when words which are misspelt happen to form strings which are also in the dictionary, thus deemed by the system to be correct. These errors are generally referred to as *real-word errors* and are much more difficult to detect than non-word errors due to dictionary lookup being of no use by definition, although changing the size of the dictionary will have an effect on the number of real-word errors present (Peterson, 1986). The extent of this problem is naturally difficult to approximate due to the difficulty in detection, although a handful of early studies have attempted to calculate the proportion of spelling errors that are real-word errors: Peterson (1986) estimated that the probability of an undetected typing error is between 2% and 16%, depending on the size of the word list used, the findings of Mitton (1987) are much more alarming, 40% of the spelling errors found in his study were real-word errors. A more recent study found that 31.4% of the spelling errors in a corpus of dyslexic writers were real-word errors (Pedler & Mitton, 2010: Table 2, 757). The extent of the problem real-word errors may cause when dealing with EModE spelling variation requires investigation and quantification.

Various methods have been proposed for dealing with real-word errors. A simple approach has already been touched upon in Section 2.2.2, whereby less

frequent words are omitted from the dictionary, hence words such as *veery* (a type of bird) will be marked as incorrect. Some spell checkers may mark certain words as infrequent, and indicate this to the user (Mays *et al.*, 1991). However, even for small dictionaries, real-word errors will still occur between frequent English words. For instance, the common misconception of *there*, *their* and *they're*, between *are* and *our*, and between *to* and *too* (these are often referred to as *confusion sets* (Mitton, 1987)) could all produce spelling errors which remain undetected due to all of the listed words being frequent in English language. Word-division errors can also lead to real-word errors; for example, *forgot* to *for got* and *inform* to *in form* are both possible errors which result in frequent words which would be found in any English dictionary (Mitton, 1987) – Grefenstette & Tapanainen (1994) give a good overview of such *tokenization* issues and potential solutions.

Due to many real-word errors resulting in words which would be in most English dictionaries it is apparent that further techniques are often needed to deal with the problem. As no further information can be gained from analysing the words individually, the context in which words appear needs to be taken into account. There have been many different methods presented in the literature, all of which attempt to search for words which look out of place in the surrounding context. In general, this is achieved through defining confusion sets which contain dictionary words likely to be mistakenly used in place of each other. On observation of a confusion set entry in a text, the local features of the word are calculated (i.e. its context) and also the features of each word in the confusion set if that word was placed in the same context as the word observed. If an entry in the observed word's confusion set is judged to be more likely, given the context, than the observed word then a real-word error is flagged and the most likely word is offered as a correction.

In earlier studies (e.g. Golding, 1995; Golding & Roth, 1999; Golding & Schabes, 1996; Jones & Martin, 1997; Mangu & Brill, 1997) a small finite set of common confusion sets was pre-defined, these included the likes of {*than*, *then*}, {*passed*, *past*} and {*cite*, *sight*, *site*}. The studies' evaluations tested the respective system's ability to predict the correct word from the small set of confusion sets when any word from the sets appeared in a test corpus. More recent studies (e.g. Fossati & Eugenio, 2007; 2008; Reffle *et al.*, 2009; Schaback

& Li, 2007) create confusion sets by computing the difference between all real-words found in a lexicon, this is based upon methods normally used for isolated error correction, such as edit distance and phonetic matching (see Section 2.2.3); for example, Fossati & Eugenio (2007) include all words within two edits via Levenshtein Distance (Levenshtein, 1966) as a confusion set.

Different features of a word's context are considered in different studies in order to detect (and correct) real-word errors. A popular method is the use of word-level n-grams (usually bigrams and/or trigrams) (see e.g. Asonov, 2010; Mays *et al.*, 1991; Reffle *et al.*, 2009; Verberne, 2002; Wilcox-O'Hearn *et al.*, 2008). This method requires probabilities for observed word n-grams calculated from a large corpus. When checking a text, the probability of an observed n-gram can be compared to the probabilities of the n-gram with one of the original words replaced by each word in its confusion set in turn. The n-gram with the highest probability is then the most likely correct sequence in the text – if this is different to the original sequence a real-word error is flagged and the most likely sequence used as a correction. The main problem with this method is the collection and storage of the n-gram model; the size of the corpus used has a direct impact on the coverage of the model and even for fairly large corpora *data-sparseness* is an issue due to perfectly valid n-grams just not occurring in the text. With very large datasets this becomes less of an issue; e.g. the Google n-grams from 1 trillion words of web text (Brants & Franz, 2006). However, processing time and how to store such a large matrix in memory becomes an issue, although good hashing functions will help (see e.g. Cohen, 1997).

Further information can be used 'above' the word-level to discern context. A system using Part-of-Speech (POS) tagging was proposed and partially developed by Atwell & Elliott (1987), this was based on the CLAWS POS tagger (Garside, 1987; Garside & Smith, 1997). CLAWS assigns POS tags to words based upon the probability of a word having a particular tag and the probability of POS tags co-occurring in sequence. The system proposed and developed uses these probabilities to find locations where an improbable POS tag occurs, this word is then marked as a potential real-word error. Suggestions for replacements can be found by selecting similar words which have a more probable POS tag for that location in the text. The main issue with this method is that the system will not

be able to detect real-word errors where the observed word's POS tag matches the POS tag of the intended word.

Other studies using contextual information with different and combined approaches include: Fossati & Eugenio (2007; 2008) who combined the trigrams method with the POS method, looking for unlikely sequences of mixed words and POS tags, Jones & Martin (1997) who attempted to apply *Latent Semantic Analysis* to the problem, Golding (1995) who applied a *Bayesian Hybrid* method, Golding & Schabes (1996) who combined a POS and Bayesian method, Bhardwaj *et al.* (2008) who utilised manually selected topic based language models for OCR of handwritten documents, and finally Mangu & Brill (1997), Golding & Roth (1999) and later Schaback & Li (2007) who used a variety of context features, such as co-occurrence and POS tags, in machine learning environments.

Despite the modest amount of research in the area of context sensitive spelling correction, the problem is far from being solved; Reffle *et al.* (2009) state:

> The detection and correction of false friends [real-word errors] is a notoriously difficult problem of natural language processing and despite several contributions [...], it is fair to say that the problem is unsolved from a practical point of view.

Some of the better results in the research for detection and correction of real-word errors are achieved on a small finite set of common confusion sets, the performance of the techniques discussed on less common real-word errors remains unknown. Furthermore, most studies are evaluated using artificially inserted errors as test data, i.e. words are replaced with alternatives from their respective confusion set. Whether this reflects naturally occurring real-word errors remains debatable. Of course, creating an appropriately sized test set of naturally occurring real-word errors is no small task, largely due to the inherent problem of detecting such errors. Recently, Pedler & Mitton (2010) have attempted to solve these two issues with the production of a large list of real-word confusion sets (c. 6,000 sets) and a real-word error corpus containing 833 real-word errors by dyslexic writers marked with XML tags in running text. They found that even with this larger list of confusion sets, there was an upper limit of 70% of the real-word errors in their corpus detectable with the confusion set approach. The Microsoft Office 2007 package included a context sensitive spelling corrector

(Fontenelle, 2006) which uses a trigram language model compressed with Golomb Coding (Church *et al.*, 2007). However, an "informal preliminary evaluation" by Wilcox-O'Hearn *et al.* (2008: 616) found that the system forgoes recall in order to maximise precision; their test found a precision of 100% but a recall of just 20%. This trade-off between recall and precision is common in the task (as with many other natural language processing tasks), high precision is usually preferred as low precision could potentially insert more errors into the text than are corrected, thus taking the text further away from a fully corrected version.

The use of contextual information does not need to be limited to dealing with real-word errors, the information could also be used to rank candidate replacements found for a non-word error; for instance, probabilities could be calculated based upon the frequency of the word within the text, the probability of word bigrams or trigrams constructed by using the replacement, or the most likely POS or semantic tag based on surrounding tags. In early studies using the "noisy channel" method of spelling correction (Church & Gale, 1991; Kernighan *et al.*, 1990), word frequencies are used as the "prior" for candidate scoring; more recent research by Ringlstetter *et al.* (2007a) used domain specific bigram models in much the same way. Alternative methodology by Schaback & Li (2007) exploited support vector machine learning in a "multi-level feature-based framework" to deal with both non-word and real-word errors. The contextual information used included bigrams and collocations (including POS tags).

Spelling variation in EModE is likely to include similar "real-word errors", where a variant form matches a dictionary word but another modern word is the more likely meaning. Such variants could only be normalised with context-sensitive methods, as described above. However, the extent of such variants in EModE corpora is not known and is difficult to establish due to the inherent problem of detecting such variants. Nevertheless, a quantification of the problem is required in order to better understand the characteristics of EModE spelling variation to aid the development of a normalisation tool.

2.2.5 Summary

Dealing with spelling errors is a well established and much studied area of research; an overview of the main considerations and solutions offered has been given

in this section. Whilst many studies have focused upon a specific method or application (or small subset of methods and applications), more recent studies have combined a variety of methods to gain increased performance. They have also introduced machine learning to allow the normalisation of different sources of spelling errors. Schaback & Li (2007), for instance, developed a tool which uses information at the phonetic, character, word, syntactic and semantic levels to find and rank candidate spelling corrections. Their system can deal with non-word and real-word errors and be trained via a support vector machine to deal with different spelling varieties. They boast high recall and precision figures of 90% for automatic correction[17], higher recall figures are achieved when taking into account an increased number of candidates, up to 97% for the five best. Other research by Mitton (2008) concentrates on non-word errors and uses a combination of well known techniques such as edit distance, letter replacement rules and phonetic matching to find and rank candidate corrections. This combination of techniques allows for quicker and coarser techniques (such as phonetic matching) to first find a list of candidates and then the slower but more accurate methods (such as edit distance) can be used to improve the ranking. Results are given for different collections of spelling errors. On one collection of "hard to correct" errors used for testing the Aspell spellchecker, Mitton's methodology placed the correct replacement in first position on 71.1% of occasions. This percentage again increases when more candidates are taken into account, up to 94.4% for the top ten candidates.

For word processing applications, having the correct word as the top candidate is perhaps not of key importance as the user can always choose a correction further down the list, if present; however, when automatic correction is desired, the top candidate must be correct in most cases to achieve high recall, and more importantly high precision. A system which erroneously 'corrects' words, whether originally a spelling error or not, on a regular basis will be of little use when pre-processing texts for applications such as IR or corpus linguistics and could actually make the situation worse. Reynaert (2008a) discusses the importance of taking into account recall and precision when evaluating spelling correction tools.

[17]It should be noted that Reynaert (2008a) disputes, to an extent, the figures presented, particularly precision as words in real text which would be replaced erroneously have not been taken into account.

Despite the amount of research in the field, there remains many research avenues to explore, both in terms of new methodology and improving established techniques. Particular issues remain in the detection of real-word errors, detection coverage and acquiring accurate corrections based on all relevant information in an interactive environment. Applying the techniques discussed here to different sources of spelling errors is also an area under increasing research, especially for CMC data and also historical spelling variation – the focus of this thesis and the more specific related work discussed in the next section.

2.3 Specific Research Dealing with Historical Spelling Variation

The previous section described some of the methods used to deal with various sources of spelling errors. The focus of this thesis, however, is historical spelling variation and its normalisation. In this section various studies are introduced which have attempted to address historical spelling variation for different purposes. Various methods have been used, including some introduced in the previous section. Studies dealing with specifically historical English spelling variation are discussed first, followed by research dealing with historical spelling variation in other languages.

2.3.1 Studies Concerning English

The VARiant Detector (VARD) (Rayson *et al.*, 2005) was developed with the goal of normalising Early Modern English spelling variation in order to improve the robustness of corpus analysis tools, particularly semantic annotation with the UCREL Semantic Analysis System (USAS) (Archer *et al.*, 2003). Their first step towards this goal was to compile a large "EModE regularisation list", this was built through manual inspection of words assigned the *Z99* tag by USAS. The *Z99* tag is assigned when USAS fails to assign a semantic tag to a word, this is likely to indicate a spelling variant due to USAS's reliance on a modern dictionary (see page 20). Various EModE corpora were used, including newspapers from 1653 and 1654, the *Nameless Shakespeare* and Chadwyk-Healey's *Eighteenth and Nineteenth Century Fiction* collection. They also used the *Oxford English*

Dictionary and "other historical sources" to verify their variants and also extend their list. In total, they collated 45,805 variant to modern equivalent mappings. The initial version of VARD, evaluated by Rayson *et al.* (2005), searched for the variant forms from the regularisation list and replaced them in the text with the modern equivalent, the original variant spelling being retained within an SGML tag. VARD was compared to Microsoft Word and Aspell (see Section 2.2.1) to evaluate its performance against commonly used spelling correction tools. Four texts (two for the Aspell evaluation) dated 1666–1679 from different categories of the *Lampeter* corpus were used in the evaluation. For the Microsoft Word evaluation, out of 551 variants detected by either program, VARD normalised 71.1% correctly compared to Word's 48.4%. In the Aspell evaluation 348 variants were detected by either program, of these VARD correctly normalised 59.8%, whilst Aspell normalised 35.4% correctly. It was also noted that both Word and Aspell marked many words as variants incorrectly, for example foreign words and proper nouns, however, VARD would only regularise words which were included as variants in its regularisation list. Clearly, VARD was much more useful than the two modern spell checkers which had been evaluated and the technique employed deals with a substantial amount of spelling variation. However, only the normalisation of a small selection of late 17th century texts has been evaluated, earlier texts from the EModE period are likely to provide increased amounts of spelling variation and texts from different sources are likely to introduce different challenges. One potential problem, due to the extensive variety in spelling variant forms (see Section 2.1.2), is that it is impossible to include all conceivable spelling variants in a pre-defined list, and even including a large proportion of variants would require an unreasonable amount of time and effort. Another problem is that the use of a pre-defined list in the way described is a binary operation, if a variant from the list is present in a text it will always be normalised with the mapped modern equivalent. The user may require more control than this as some mappings may be ambiguous or not appropriate for a given context (see Section 2.2.4). It is highly unlikely that one list of EModE spelling variants and their modern equivalents will be applicable to all EModE texts; for this initial version of VARD to be accurate as a 'generic' EModE normalisation tool, multiple lists will need to be managed for different EModE texts in terms of text type, genre, date and other factors.

Schneider (2001) investigated using a modern spell-checker to improve the accuracy of a wordclass tagger, ENGCG (Voutilainen & Heikkilä, 1993), on the ZEN corpus, a collection of late 17th and 18th century newspapers (Fries & Schneider, 2000). Microsoft Word's spell checker was briefly tested for appropriateness; it was concluded that whilst it found correct suggestions for some unrecognised items from ZEN it was "of limited use for a research project". Reasons given were that it always operates interactively – difficult when processing a million-word text, no customisation of the spell checker was possible except for adding to the dictionary of correct words, and finally the methods used by the system are "intentionally opaque". Schneider moved on to trying the open source application Aspell. This software allows for any dictionary to be used, manual editing of the phonetic rules used for spelling correction, and transparency on how the system came to produce a replacement. By providing a manually produced dictionary from the list of words in the ZEN corpus recognised by ENGCG and some customisation of the phonetic rules, some success was achieved. However, it was still apparent that Aspell quite often did not give the correct replacement as its first choice (no quantification of this was given) – as required for automatic processing of the text. It also dealt badly with hyphenated forms, and many problems still occurred where the correct replacement was not included in the dictionary. Schneider (2001: 209) concluded, "It is a long way from a decent interactive spelling checker to automatic spelling normalization." Although, it was pointed out that even dealing with a small amount of the spelling variation should considerably improve ENGCG's recognition rate of the ZEN corpus. It should also be noted that this study's concentration was mainly 18th century English, which contained less spelling variation than EModE texts from earlier centuries.

Markus (2000) introduced a system for "normalizing" Middle English texts, although Markus (2002) briefly shows its application with EModE texts. Middle English has its own features and challenges which are out of the scope of this thesis, however, this study is worthy of mention due to the unique presentation of the normalised forms. A set of basic rules were created which could transform some Middle English spelling into modern equivalents, these rules were then processed over the text and a (partially) normalised version of the text created. Interestingly though, each line of the original text is retained with the newly normalised form

of each line produced in parallel below. By using these parallel lines one can imagine that it is possible to instruct a corpus tagging program to process only the normalised lines, whilst maintaining links to the original word forms. Much greater precision should be achieved than would be possible by tagging the original text. This early work by Markus (2000; 2002) only achieved a small portion of the complete spelling normalisation automatically; he planned to improve the rules used which should have gone some way to addressing this problem. However, it is questionable how large a percentage of spelling variation can be normalised with a rule-based approach alone.

Craig & Whipp (2010) describe their methodology for creating more reliable statistics, particularly frequency lists, when analysing EModE plays and poems, with an overall aim to aid authorship attribution, including also type token ratio, collocations and concordances. To achieve their goal, they first normalised a fixed list of 200 common function words, they found that generally there was only a small range of variation in these words[18], and that they could use "find and replace" functions to achieve normalisation of these words within the text (the original spelling was retained as an XML attribute). Examples included *bee* to *be* and *al waies* to *always*. For the remainder of the words, mainly lexical, the problem was much more complicated, they state that "[s]pelling variation with the lexical words, by contrast, are multiple and unpredictable." To deal with this, they created lists of spelling variants which were linked to "an underlying common lexical item" and when creating statistics counted instances of this common "headword" instead of the spelling variant. Their list of spelling variants was sourced by three methods: harvesting variant spellings from a raw text version of the Oxford English Dictionary, aligning short passages from "old spelling" Shakespeare with modernised versions and picking out variants where present, and finally using a basic set of letter replacement rules (e.g. $u \rightarrow v$ and $i \rightarrow j$) on the 280,000 variant forms sourced by the previous two methods to find other possible variant forms. In some cases a variant was associated to more than one headword; for example, *weeke* could equate to *week*, *weak* or *wick*. Here, disambiguation using the word's context was employed. To achieve this, the prose fiction section of the British National Corpus was used as a reference corpus and statistics calculated

[18]This gives support to recent studies which have found that more frequent words are less likely to change over time (Lieberman *et al.*, 2007; Pagel *et al.*, 2007)

for each of the headword's collocates. The process worked by looking at which words appeared either side of the variant form in question (various window sizes were tested), a composite score was then calculated for each headword based on how often each of these collocate words appeared in the same position next to the headword in question within the BNC data. The headword with the highest composite score was chosen as the most likely headword for the variant due to the context of that headword best matching the context of the variant form. Their evaluation found that the immediate context of the variant form (i.e. the very next and previous words) was most useful in the disambiguation, although including a larger window of context improved results fairly substantially until -4 and $+4$ words. They also found that just choosing the most frequent headword produced similar recall figures, albeit lower precision. Their final results, combining all of their methods optimally, produced a reduction in word types from 88,075 to 61,471. Before disambiguation, 5,383 words are classed as ambiguous, the disambiguation only solves 350 of these (although a high precision threshold is used) and thus showing the "complexity of the disambiguation task".

Robertson & Willett (1992; 1993) describe their research into using modern spelling-correction methods to supplement queries on historical text databases (specifically from the 16th, 17th and 18th centuries). To this end, they developed a phonetic algorithm with rules customised to historical texts, they also evaluated other common spelling correction techniques applicability to the task, such as n-gram analysis and Minimum Edit Distance algorithms. Their evaluation found that their version of digram matching was the most appropriate method for their task with a recall rate of 90.7% - 96.7% (minimum and maximum recall in top 20 returned words in 4 different corpora tests), outperforming phonetic matching with a recall of 73.9% - 93.7%. The digram matching technique was slightly outperformed in terms of recall by an edit distance algorithm (Needleman & Wunsch, 1970): 92.7% - 98.4% and a similar Longest Common Sequence algorithm (see Wagner & Fischer, 1974): 95.4% - 98.3%, however, both of these techniques required a far greater amount of processing time and thus are too slow for interactive processing. Their study only evaluated the methods in terms of recall, the precision of each method does not seem to have been considered. Despite this and even though the study did not deal specifically with normalising spelling,

their study shows that modern spell-checking methods can be applied to spelling variation in historical texts with some success.

2.3.2 Studies Concerning Other Languages

There have also been various studies which deal with spelling variation in historical varieties of other languages. Whilst it should be taken into account that the methods described in these studies are generally tailored to these other languages, it is worth considering the possibility of adapting and applying methods used to Early Modern English.

Spanish

In their quest to apply automatic POS tagging to a diachronic corpus of Spanish, Sánchez-Marco *et al.* (2010) describe two methods for improving the accuracy of the FreeLing[19] POS tagger when used with their corpus. Their initial approach was to (partially) normalise spelling variation before running the texts through the POS tagger. This increased the tagging accuracy to 91.5%, up from 77.5% on the original texts. Despite these promising results, the researchers point to "shortcomings" of this approach in that the tagging accuracy is still lower than the expected 95% and above, that the original variant forms are lost in their normalisation procedure and that there are differences between old and modern Spanish other than on the orthographic level which will also affect POS tagging accuracy. Instead, they propose to modify the actual tagger rather than the texts being processed, adapting the FreeLing tagger to the "diachronic varieties of Spanish". It will be interesting to learn from their future work how this methodology performs compared to their initial approach. One reason to take preference in modifying the texts rather than the tagger is that the normalised texts will also be of use when increasing the accuracy of other corpus linguistic methods, as discussed in Section 2.1.3.

German

Bennett *et al.* (2009; 2010) describe their ongoing work annotating a corpus of Early Modern German. Due to the inaccuracies they identify for automatic

[19] `http://www.lsi.upc.edu/~nlp/freeling/`

annotation when dealing with texts containing high levels of spelling variation (as is the case in Early Modern German), they plan to create a "historical text processing pipeline" which will include the normalisation of historical German spelling variation to "improve the output of the POS tagger and lemmatiser and will thus reduce manual labour". The pipeline they advocate is similar to an aim of the research presented in this thesis, i.e. the development of a pre-processor for corpus linguistics on EModE texts which deals with spelling variation and thus improves the accuracy of corpus linguistic tools.

Pilz & Luther (2009); Pilz *et al.* (2006; 2008; 2009) present their work on producing a "fuzzy search engine" to increase recall when searching non-standardised pre-1901 German texts. They use phonetic matching, rule-based approaches and distance measure via an enhanced Levenshtein Distance algorithm. Interestingly, they also looked at the meta-data of the text, including the date the text was written and the geographical origin of the author, in order to decide how appropriate certain rules and methods are to the text. This research has a different focus to the research detailed in this thesis (other than the obvious language differences); the concentration in historical information retrieval is to supplement the query with possible spelling variants of the modern words found in the query string, that is, given a modern word, generate all possible spelling variants – recall is the priority.

Like Pilz *et al.*, Gotscharek *et al.* (2009) aim to improve Information Retrieval (IR) when searching historical German texts from the 16th through to the 19th Century. They evaluate the performance of *Matching Procedures*, which is essentially using letter replacement rules to find *fuzzy matches*, and note a significant drop in performance for texts from earlier centuries. They advocate the need for *Special Lexica*, which are a series of mappings between variant forms and modern equivalents that can be used to input extra search terms which are equivalent to the modern word form in the search query. They present a tool which can be used to build these letter replacement rules and the variant form mappings "in an interleaved way based on corpus analysis". The web-based tool introduced allows users to collaboratively decide on the modern form equivalent for variant forms which are presented from a document or corpus as being potential variants due to them not appearing in a modern lexicon. They go on to state that the

addition of mappings from their tool would be required to improve performance in the earlier texts where the letter replacement rules are less successful.

French

O'Rourke *et al.* (1997) reported on a study which evaluated modern spelling-correction methods on 13th century Old French spelling variation. This study followed on from similar research on Early Modern English texts by Robertson & Willett (1992; 1993) (see Section 2.3.1 for further details), evaluating the effectiveness of digrams, trigrams and Longest Common Sequence in terms of recall in historical database searching. They achieved recall rates of approximately 70%; this being some 15-25% lower than for English, attributed in part to the greater age of the French text. As with the English data (Robertson & Willett, 1992; 1993), the Longest Common Sequence (LCS) method achieved the highest recall score, although was much more demanding of computational time. They thus concluded that digram matching was the most appropriate method, with it outperforming trigram matching in terms of recall and outperforming LCS in terms of efficiency. As recall is the most important factor when supplementing search queries, precision does not seem to have been taken into account.

Dutch

As in other studies described in this section, Koolen *et al.* (2006) aimed to improve historical document retrieval, this time introducing what they state is a cross-language approach, although their case study only shows the results with 17th century Dutch documents. However, unlike other studies, the researchers performed "document translation" instead of "query translation". That is, they automatically normalised the historic texts being searched, rather than supplementing the modern search queries with spelling variant equivalents. The (partial) automatic normalisation was achieved through letter replacement rules, the rules were derived by finding the character differences between historical words and modern words where the historical and modern words were considered equivalent. The equivalent historical and modern words were found by three methods: phonetic matching, sequence (of vowels and consonants) matching and n-gram matching. It is likely that many of these matches were not actually

equivalents, however, the researchers counted only frequently occurring letter replacement rules in the hope that these would be sufficient for normalising historical variants into their correct modern forms. Evaluating the derived letter replacement rules on a test-set of 400 historical and modern equivalent word pairs, their best result was achieved by using all derived rules; 337 of the 400 historical words were re-written into some form by the rules. However, only 224 of these were perfectly re-written, i.e. into the correct modern form (see Koolen *et al.*, 2006: Table 1, pp. 413). Their evaluation of the normalisation's effectiveness on historical document retrieval showed some improvement through their partial normalisation, although it would seem further work is necessary to achieve results comparable to modern document retrieval.

Brazilian Portuguese

Giusti *et al.* (2007) describe their work on historical Brazilian Portuguese. In their efforts to produce "The Historical Dictionary of Brazilian Portuguese" from a corpus of Brazilian Portuguese texts from the 16th century through to the early 19th century, they used an approach heavily based upon transformation rules to "cluster distinct spelling variations around a common form". The form around which variants are clustered "is not always the orthographic (or modern) form" – the aim here was not to normalise the spelling of variants but to detect potential spelling variants and group equivalent variants together. This method of detecting spelling variants is interesting as it does not require a modern dictionary, the transformation rules find relationships between words and "it is expected that this relation shows spelling variation for any given word". The method was successful for the purpose of the task with high precision being achieved, however, recall was relatively low, although the study theorises that recall could be improved through the development of further transformation rules with little effect on precision. One may hypothesise that a similar technique could be used to cluster proper nouns in a corpus of EModE (or other historical language varieties) where variation in the spelling of names was frequent; it has been shown that Shakespeare was spelt in a multitude of forms, even by Shakespeare himself[20]. This study focuses on historical Brazilian Portuguese, so whether the techniques

[20]see `http://shakespeareauthorship.com/name1.html`

could be applied to EModE would require further investigation. It is worth noting that the production of a dictionary of EModE spelling variants, similar to that produced for historical Brazilian Portuguese, would be extremely useful in the correction of spelling variants; the data could be used to create an improved list of variant to modern form mappings, in turn this could be used as a "gold standard" for the evaluation of spelling correction techniques and also be used to find new letter replacement rules and statistics. The inclusion of historical spelling variants in the Oxford English Dictionary has been explored; however, this is largely a manual process and takes considerable time (see Simpson *et al.*, 2004).

2.4 Chapter Summary

This chapter has given an overview of the issues, potential solutions and related work relevant to the research detailed in this thesis. We began with an introduction to Early Modern English, with particular focus on its inherent spelling variation and the problems this causes for historical corpus linguistics. The characteristics of EModE spelling variation dictate that any effort to normalise EModE texts would not equate to a translation like exercise due to there being no fixed spelling forms and individual spellings potentially being unique to that author, scribe or piece of text. The actual size of the spelling variation problem is difficult to assess; many researchers have acknowledged that spelling variation is present in large quantities, particularly in earlier texts from the EModE period, but there is very little quantitative evidence of the ratio of spelling variants to modern forms and hence how much normalisation is required. This needs to be addressed with a quantitative analysis of the levels of spelling variation across different EModE corpora. A number of corpus linguistic techniques have been discussed, with particular focus on the possible effect that spelling variation will have on their accuracy. Whilst the effect on basic techniques, such as searching and frequency lists, are obvious, the effect on more complicated techniques may be more subtle and difficult to predict. The size of the impact that spelling variation has on these methods is unknown, with the exception of semantic annotation via a study by Archer *et al.* (2003). Therefore, analyses are required to establish the effect spelling variation has on more complicated corpus linguistic tools.

The second main strand of the background literature focused upon techniques used in research and software tools to deal with spelling errors and variants from a wide range of sources. Whether these methods can be applied to EModE spelling variation is one of the central research questions of this thesis (see *RQ 2* - Section 1.2). Whilst the specific related work (presented in Section 2.3) has shown some research using these techniques, none have established a robust enough solution to deal with the specific problem of EModE spelling variation and its effect on historical corpus linguistics. Throughout the literature presented in Section 2.2, the need for the analysis of various EModE spelling characteristics has been highlighted. These include features such as: the position of character level variation, particularly for the initial letter – it has been found that for modern spelling errors initial letter mistakes are relatively rare; whether EModE spelling variants contain patterns of specific letter replacements, such as those found for OCR and typing errors; the difference in terms of edits between spelling variants and their modern equivalents – it has been found that the majority of modern spelling errors are only one edit away from the intended form; and finally, how likely an EModE spelling variant is to match an 'unintended' dictionary term – as with real-word errors in modern spellchecking, which can only be detected through the analysis of contextual information. These spelling characteristics shall be those investigated in order to address *RQ 2* (Section 1.2).

Chapter 3 will build upon the background presented in this chapter by detailing efforts to provide the highlighted quantitative studies and analyses required to both reinforce the need for a solution to the problems caused by EModE spelling variation and to establish the specific characteristics of EModE spelling variation which will influence the application of modern spellchecking techniques.

Chapter 3

Analysis of Early Modern English Spelling Variation

The previous chapter has highlighted a number of areas which require additional research in order to better understand the problems caused by Early Modern English (EModE) spelling variation and to establish its precise characteristics. These will affect the application of modern spellchecking techniques to EModE spelling normalisation. The aim of this chapter is to bridge that gap by presenting the results of various quantitative studies of EModE spelling variation and its properties. The extent of the spelling variation problem shall be dealt with first. Section 3.1 will describe the process of quantifying the qualitative view of many scholars that a large amount of spelling variation existed in the EModE period and, furthermore, show that this level of spelling variation reduced progressively throughout the period. The problems this spelling variation has on corpus linguistic methodology shall be dealt with in Section 3.2, with two advanced automated corpus linguistic techniques evaluated with EModE texts before and after normalisation. The work presented in Sections 3.1 and 3.2 will address *RQ 1* (see Section 1.2). Section 3.3 will begin to address *RQ 2* with various EModE spelling variation characteristics, as specified in Section 2.4, investigated by comparing EModE spelling variant analysis with previous research and complementary analysis of spelling errors from modern sources. The potential implication of these characteristics on applying modern spellchecking techniques (as described in Section 2.2) to EModE spelling variation will also be discussed. The chapter will end with Section 3.4, which will summarise the findings presented

and link these to the development of a solution to the problem of EModE spelling variation, as will be presented in Chapter 4.

3.1 Levels of Spelling Variation in Early Modern English

In order to address the first part of *RQ 1* (Section 1.2), here we discover, quantitatively, the extent of spelling variation in the EModE period. This is important because many researchers comment on the large amount of spelling variation within the period without explicitly quantifying it (see, e.g. Görlach, 1991; Vallins & Scragg, 1965). One exception being Schneider (2001), who, in his attempts to develop a normalised version of the Zurich English Newspaper (ZEN) Corpus (1670-1799), produced an overview of the spelling variations contained within. Schneider found that 3.99% of the tokens and 38.02% of the types[1] within the ZEN corpus were unrecognised by the ENGCG tagger[2], and hence could be considered spelling variants. The corpus was also split into four time periods, 1670–1709, 1710–1739, 1740–1769 and 1770–1799. The percentage of unrecognised tokens and types reduced in each subsequent time period, from 4.66% tokens and 36.57% types in the 1670–1709 sub-corpus to 2.85% tokens and 26.06% types in the 1770–1799 sub-corpus.

The EModE period could be considered to be from as early as 1417 and to as late as 1776 (see Section 2.1.1 for a full discussion on the dating of the period), hence the ZEN corpus studied by Schneider only covers a small portion of the full EModE period. A more thorough quantitative study of the spelling variation in various corpora covering the whole EModE period is required. To this end, six different corpora were analysed: The ARCHER corpus, Early English Books Online, the Innsbruck Letter corpus, the Lampeter corpus, the EMEMT corpus, and a collection of Shakespeare's works. The ARCHER corpus (A Representative Corpus of Historical English Registers)[3] is a multi-purpose diachronic corpus covering texts from 1650 to the present day (only texts dated before 1800 were

[1]Types are distinct instances of a word, i.e. each word is counted once despite its frequency. Whereas the token count includes all instances of each word.

[2]See Voutilainen & Heikkilä (1993) for details.

[3]We used the ARCHER-3.x version of the corpus.

used in this study). It was built to facilitate the analysis of historical change in written and speech-based registers. Early English Books Online (EEBO)[4] is a collection of digital facsimiles of virtually every English printed work between 1473 and 1700; nearly 125,000 works. As digital images of texts are of no use in this study, advantage has been taken of access to 12,268 of the 25,000 works that have being transcribed into ASCII SGML texts as part of the EEBO Text Creation Partnership[5]. The Innsbruck Letter corpus, part of the Innsbruck Computer-Archive of Machine-Readable English Texts (ICAMET) corpus (Markus, 1999) is a collection of 469 complete letters dated between 1386 and 1688, a total of 182,000 words. The Lampeter corpus of Early Modern English Tracts (Schmied, 1994) is a collection of tracts and pamphlets published between 1640 and 1740. Each decade has two texts from each of the following six domains: religion, politics, economy and trade, science, law, and miscellaneous; resulting in a corpus of 120 texts and c. 1.1 millions words. The Early Modern English Medical Texts (EMEMT) corpus (Taavitsainen & Pahta, 1997; 2010) is a collection of specifically medical texts built to study the evolution of medical writing. The portion of the corpus used in this particular study covers 1525 to 1700. The collection of Shakespeare's works is a digitally-transcribed version of the first folio, which was printed in 1623. This can be sourced from the Oxford Text Archive[6]. Shakespeare's works were written between c. 1590 and c. 1613[7]. A summary of the corpora used in the analysis is shown in Table 3.1. The total coverage of the corpora used in the quantitative analysis dates from 1410 to 1799; thus representing the entire EModE period. The corpora are all very different, covering various genres and text types. It is important to note that the corpora are never combined in the analysis and are always treated as separate entities.

[4] `http://eebo.chadwyck.com/`

[5] `http://www.lib.umich.edu/tcp/eebo/`

[6] `http://ota.ahds.ac.uk/`

[7] It should be noted that the dates given for Shakespeare's plays are estimates as there is considerable debate with respect to precise dating.

Corpus	Genre/Type	Years[a]	Texts	Tokens
ARCHER	General/Mixed	1660–1799	364	632,639
EEBO	General/Mixed	1470–1709	12,265	535,910,150
EMEMT	Medical texts	1540–1699	51	491,384
Innsbruck	Letters	1410–1689	436	170,538
Lampeter	Various tracts and pamphlets	1640–1739	120	1,124,131
Shakespeare First Folio	Plays	1590–1613	36	821,123

[a] The full decade range was not used from all corpora due to texts dating too far from the EModE period or a lack of texts and/or words from certain decades.

Table 3.1: *Summary of corpora used in study of EModE spelling variation levels.*

The first stage of the investigation involved sampling each corpus at regular intervals in order to gain a fair representation of the corpus over time. A sample period of ten years was chosen, hence the texts were split into their relevant decade (e.g. 1410–1419). This level of sampling did mean a small number of decades were omitted in certain corpora due to a lack of texts and/or words. The smaller EMEMT corpus could not be sampled in this way due to many decades containing only one or two files, or a small number of words; therefore the decision was made to include everything from the EMEMT corpus with a minimum of two files per decade. All results were normalised to a percentage in order to compare corpora with different sample sizes. The sampling sizes for each corpus are shown in Table 3.2.

For the more general corpora (ARCHER, EEBO and Lampeter), a minimum of ten texts per decade were required to ensure that one text did not account for more than 10% of a decade's sample. Elsewhere, a smaller number of texts were sufficient due to the fact that the specialised form of the corpora resulted in less variety of text. Samples were chosen from randomly selected texts from each decade, with the sample from each text beginning at a randomly selected index (word count) within the text.

Corpus	**Decade Sample Size**	**Minimum Texts**	**Decades not included due to a lack of texts and/or words**
ARCHER	4,000	10	1740
EEBO	80,000	10	
EMEMT	Total possible	2	1620, 1640
Innsbruck	1,200	4	1420, 1430, 1490, 1590
Lampeter	40,000	10	
Shakespeare First Folio	60,000	4	

Table 3.2: *Corpus sample sizes for study of EModE spelling variant levels.*

In order to assess the level of spelling variation in each sample, variants needed to be detected and counted. To achieve this, each word in each sample was checked against a dictionary of modern words, if a word was not found in the dictionary then it was counted as a variant. The dictionary used was a list of modern words derived from the Spell Checking Oriented Word List (SCOWL)[8] and a list containing words with a frequency of at least 1 per million and a range of at least 50 (out of 100 sectors of the corpus) in the British National Corpus (BNC) (Leech *et al.*, 2001). This analysis provided figures per decade sample in each corpus for the percentage of both types and tokens which could be considered spelling variants. These percentages are comparable to those found by Schneider (2001), as described on page 50.The variant type percentages are plotted in Figure 3.1 and the variant token percentages are plotted in Figure 3.2. An average variant percentage over all the available corpora for each decade was also calculated; this is shown for both types and tokens in Figure 3.3. The general trend line is shown with a dotted line in all four graphs.

[8] `http://wordlist.sourceforge.net/scowl-readme`

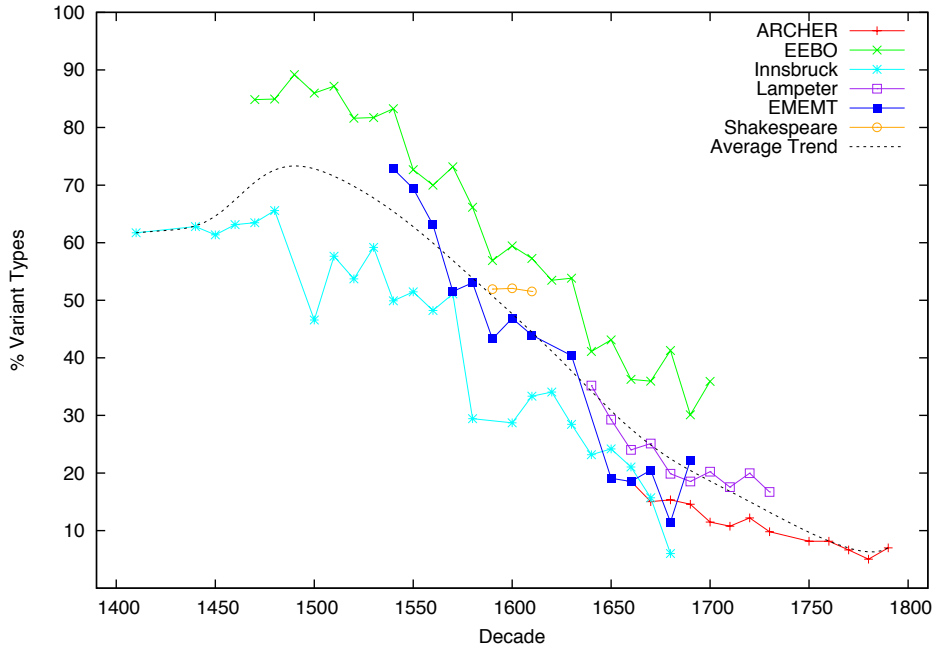


Figure 3.1: *Graph showing variant types % in all corpora over time.*

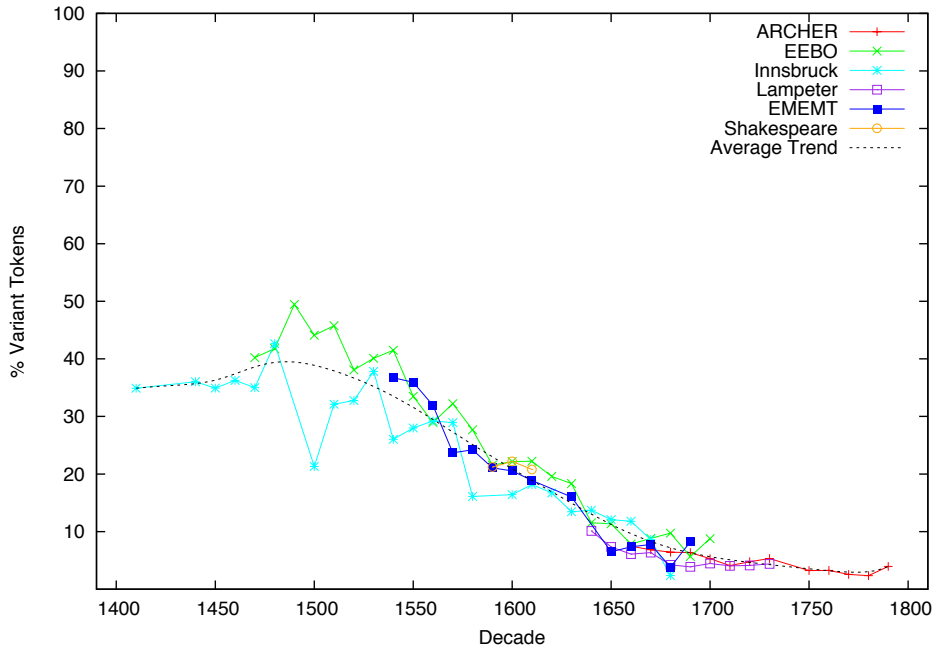
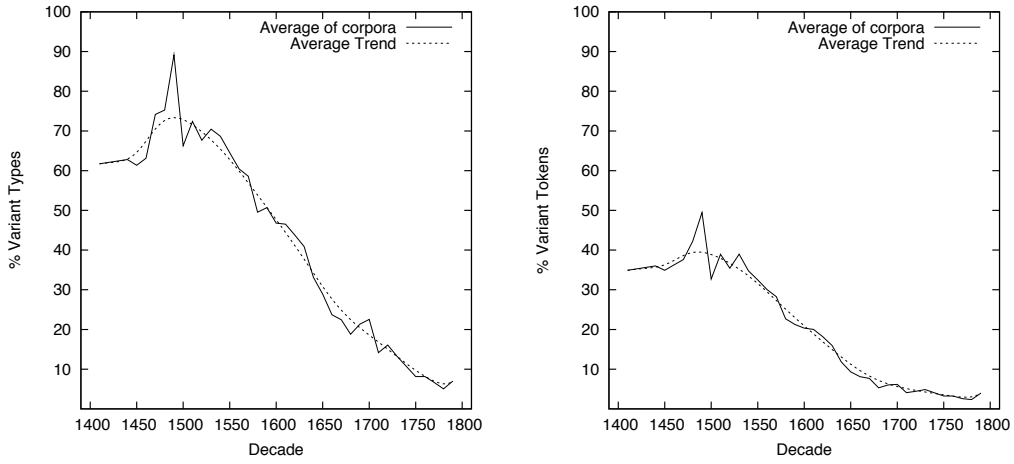


Figure 3.2: *Graph showing variant tokens % in all corpora over time.*



(a) Types %

(b) Tokens %

Figure 3.3: *Average variant percentage over corpora available for each decade.*

Figures 3.1–3.3 all show a definite downwards trend in respect of the amount of spelling variation occurring throughout the EModE period. This not only corroborates Schneider (2001)'s quantitative analysis of the ZEN Corpus for the latter part of the EModE period (1670–1799), but also quantifies the trend over the entire EModE period, verifying many scholars' claims that the language was under significant change throughout the period (see, e.g., Görlach (1991: 8–9), Lass (1999: 56) and Rissanen (1999: 187)). Another point to note is that the rate of reduction in variation slows from around 1700; this is particularly noticeable in the graphs representing tokens (Figures 3.2 and 3.3b). This corroborates Görlach (1991: 11)'s claim that by 1700 the language had achieved "considerable homogeneity", with regional (written) dialect differences no longer present and "the period of co-existing variants, so typical of all levels of EModE, being over".

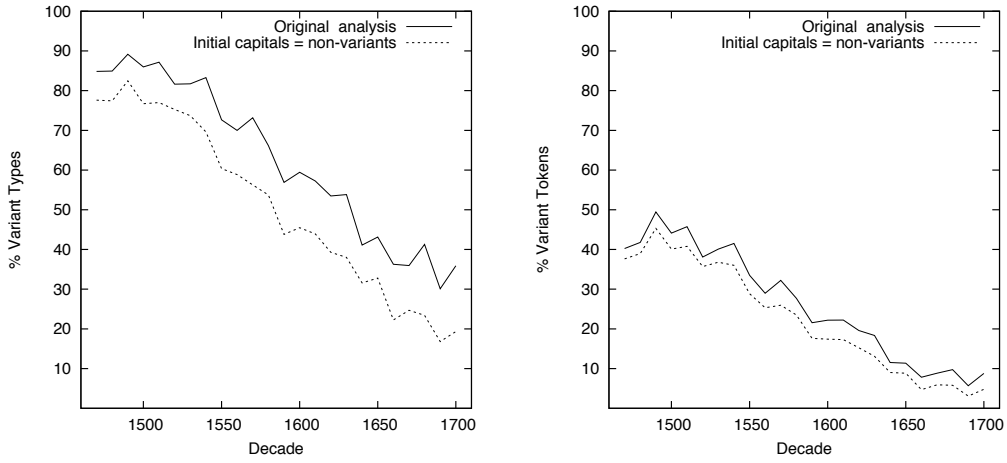
There is a noticeable difference between the levels of variant types and variant tokens in Figures 3.1–3.3. The differences are likely to be related to the underlying properties of language and the frequency of words, i.e. Zipf's Law (Zipf, 1932), but also due to the properties of the EModE spelling variation (as described in Section 2.1.2). In particular, words will be found spelt in a variety of forms (as discussed in terms of the effect on word frequency lists in Section 2.1.3, page 18). In the analysis presented, each spelling of a word will be counted as a separate

type, therefore the proportion of spelling variant types will increase at a greater rate than the proportion of spelling variant tokens when more spelling variation is present, i.e. in earlier periods of EModE. Furthermore, highly frequent words are less likely to have variation in their spelling, as found to be the case by Lieberman *et al.* (2007) and Pagel *et al.* (2007), and specifically for function words by Craig & Whipp (2010: 40). The result of this will be that higher frequency words are less likely to be counted as spelling variants, therefore the proportion of spelling variant tokens overall will be lower than the proportion of spelling variant types – by definition, frequency will not effect the variant type percentage.

It should be noted that the variant percentages shown in this section do not represent absolutely precise variant rates; they are all approximate values due to the automatic method used to detect variants. First, some variants may appear in the dictionary as other words, these are known as real-word errors in spellchecking exercises (see Section 2.2.4), this particular problem shall be evaluated fully in Section 3.3.1. Secondly, words may not be in the dictionary, and hence considered variants, but are perfectly valid forms. These may include proper nouns, encoded words (e.g. with Unicode entity values), words in languages other than English (e.g. Latin, French or Italian) and words which are simply not in the modern word list but are perfectly valid (e.g. archaic and obsolete words such as *betwixt* and *howbeit*). All of the problems listed occur in some of the corpora used in this study. Whilst a large amount of time was spent 'cleaning' the texts, it is impossible to remove all imperfections; EEBO, for example, contains many Unicode entities for which there is no obvious ASCII replacement, and any word containing one (or more) of these values will be counted as a variant by the detection method used. The Lampeter, ARCHER, Innsbruck and EEBO corpora are known to contain sections of Latin and, in some cases, French and Italian passages; some of these passages will no doubt have been passed into the corpora samples. Aside from the odd exception all words in these foreign passages will be counted as variants.

Proper nouns invariably cause problems when detecting spelling variants, whether in historical texts or in modern spellchecking. Due to the potentially large number of proper nouns which could be found within any text, it is not sensible to try and list them all (although adding more frequent proper nouns is a sensible first step). A common-sense approach to the problem would be to exploit the rule that proper nouns always begin with a capital letter in Modern

English; this, however, does not work in all cases as a capital letter is also used to signify the start of a sentence. The problem is even worse in EModE, particularly in later EModE texts; as previously stated (p. 25), Osselton (1998) describes how between 1550 and 1750 there was a distinct climb in the use of a capital letter to begin nouns where one would not be present in Modern English. The effect of this proper noun issue is evaluated in Figure 3.4 where the EEBO corpus samples are analysed as above and also by counting all words beginning with a capital as non-variants. As can be seen, variant counts are consistently lower if words with initial capitals are not considered as variants. However, the general downward trend remains the same with the lines following almost parallel paths. Marking all initial capital words as non-variants will no doubt lead to an increase in real-word errors due to 'abnormal' capitalisation of words which are also variants, sentence initial variants and inconsistently spelt proper nouns.



(a) Types %

(b) Tokens %

Figure 3.4: *Comparison of variant counts in EEBO corpus samples with (=original) and without initial capital words.*

It is clear that the levels of variation displayed in Figures 3.1–3.4 are approximations. However, it is reasonable to assume that the level of 'noise' leading to inaccuracies is relatively uniform throughout corpus samples and thus the general trend of spelling variation reducing over time throughout the Early

Modern English period is maintained. The variant detection method is discussed in more detail in Section 2.2.2 and shall be fully evaluated in Section 5.1.

3.2 Effect on Corpus Linguistics

Having established the quantity of spelling variation in various EModE corpora, we now move to addressing the remainder of *RQ 1* (Section 1.2): how large an effect does this spelling variation have on corpus linguistic methodology? Establishing this effect is key to justifying the research described in this thesis; if it is shown that spelling variation creates a significant barrier to robust and accurate historical corpus linguistics with EModE corpora, then a solution is required to deal with the spelling variation. In Section 2.1.3, various EModE corpora were introduced and the potential issues spelling variation is likely to have on a number of corpus linguistics techniques when used with these corpora were discussed. In this section, we evaluate the precise effect of EModE spelling variation on two automated advanced corpus linguistic techniques, examining the accuracy of both automated methods with and without the spelling variation present.

3.2.1 Part-of-Speech Annotation[9]

The first corpus linguistic method evaluated was Part-of-Speech (POS) annotation (or tagging). POS annotation involves assigning each word a grammatical tag (verb, noun, adjective, etc), these can then be used to perform further analysis, such as disambiguating the grammatical meaning of a word (e.g. "ship" could be used as a noun or a verb). The importance of grammatical tags in historical corpora has been highlighted as early as Kytö & Rissanen (1993: 1), who, when discussing the Helsinki corpus, states "the usefulness of our corpus is diminished by the absence of grammatical tagging. This means all searches must be based on words, or their parts or combinations."

Automatic POS tagging can be achieved with various tools, the CLAWS tagger (Garside & Smith, 1997) developed at Lancaster University has been used for this particular evaluation. Since CLAWS uses a hybrid rule-based and

[9]The research presented in this section was completed in collaboration with Paul Rayson, Dawn Archer, Jonathan Culpeper and Nicholas Smith (see Rayson *et al.*, 2007).

probabilistic approach, it is anticipated that similar results would be observed with other POS taggers. For modern written text (a BNC sample), CLAWS achieves 96–97% accuracy (Leech & Smith, 2000) in terms of correct annotation. Here, we wish to observe the accuracy CLAWS achieves with EModE texts, and how much of an effect spelling variation has on this accuracy. Archer *et al.* (2003) have previously evaluated the performance of another form of automatic tagging, semantic annotation, on two short, relatively contemporary, EModE texts. Accuracy rates were high, despite the nature of the texts, at 97.1% for one text and 96% for the other. Although, after partially dealing with spelling variation, accuracy increased to 98.8% and 98.6% respectively. Here, earlier and possibly more challenging texts will be used to establish the accuracy of automated POS tagging with EModE texts.

For this experiment two sources of EModE text were used: Shakespearean texts and texts from the Lampeter Corpus. Five Shakespeare plays were sourced from the First Folio (printed in 1623) provided by the Oxford Text Archive[10]. The plays chosen were limited to one genre, comedies, as that was the only genre which covered his entire writing career. The plays selected were *Taming of the Shrew*, *Love's Labour's Lost*, *Merry Wives of Windsor*, *Twelfth Night* and *Tempest*. These plays evenly spanned his writing career, although the precise dating of plays is a subject of much debate. A further three texts were taken from the Lampeter Corpus (Schmied, 1994) to provide a contrast to the Shakespeare analysis. Three domains are represented: economy and trade (*eca1641*), law (*lawa1643*) and science (*scia1644*). Due to the size of the texts and to provide equal balance, a 1,000-word sample was taken from each. This was selected from a random line position and a minimum of 1,000 running words selected including up until the end of the sentence or speaker change. Microsoft Word 2003 was used to perform this task. 5,011 words of Shakespeare text and 3,025 words from the Lampeter corpus remained for analysis.

In order to perform the evaluation, for each of the eight texts four versions were produced:

[10] `http://ota.ahds.ac.uk/`

1. The raw texts were automatically POS tagged with CLAWS in its standard setup.

2. The CLAWS-tagged texts were manually post-edited to correct any tagging errors in order to produce a gold standard for comparison.

3. The spelling variation in the raw texts were automatically normalised using an early version of the VARD 2 software (see Section 4.3)[11] and then each (partially) normalised text was POS-tagged with CLAWS.

4. For each text, the spelling variation was manually (and fully) normalised and then POS-tagged with CLAWS.

The POS tags attached to each word in the gold standard (version 2) were compared to the other three versions in order to calculate the number of differences, and hence errors as the gold standard represents 100% accuracy[12]. Accuracy was calculated as the percentage of tags which matched the gold standard tags. The results for the three versions compared to the gold standard are shown in Table 3.3.

	POS tagging accuracy	
	Shakespeare	**Lampeter**
Automatically POS tagged only	81.94%	88.46%
Variant spellings automatically normalised	84.81%	89.39%
Variant spellings manually normalised	88.88%	91.24%

Table 3.3: *Comparison of CLAWS POS tagging accuracy.*

The results show a significant drop in tagging accuracy compared to results achieved with modern written texts (96–97%), this highlights the increased difficulty when dealing with historical texts. When spelling is fully normalised, a 6.94% increase in tagging accuracy is observed for the Shakespeare texts and a 2.78% increase observed for the Lampeter texts. This indicates that spelling

[11] An early version, rather than the final version, of VARD 2 is sufficient here as a formative evaluation of the effect automatic normalisation has on POS tagging accuracy.

[12] Clearly, the manual post-editing may still contain errors due to human error, but this was considered likely to be negligible for the purpose of the evaluation.

variation is a significant barrier to POS tagging accuracy. However, despite this increase, the accuracy rates are still some way from those achieved with modern texts. This is perhaps understandable due to the added difference in style and genre from the modern written texts used to evaluate CLAWS; a more accurate comparison would be against CLAWS' accuracy in tagging modern plays – at least for Shakespeare. Furthermore, various researchers have indicated the added problem of grammatical change over time (Britto *et al.*, 1999; Kytö & Voutilainen, 1995), to which CLAWS will not be sensitive to. In many cases, full manual normalisation will not be possible with large corpora due to its time-consuming nature. Hence, the manually normalised results discussed here indicate an upper boundary to POS tagging accuracy without considering additional factors to spelling variation. The automatically normalised results show that some improvement can be gained without manual intervention, but clearly improved performance in automatic normalisation will lead to increased accuracy in the POS tagging. How close automatic normalisation can get to manual normalisation shall be discussed in Section 5.2.

3.2.2 Key Word Analysis

The second corpus linguistic method evaluated was key word analysis. The process of key word analysis involves looking for words which are significantly more frequent in one text (or corpus) compared to another text (or corpus). In corpus linguistics, the standard method is to use one of a number of statistical processes to compare frequencies and find words which are 'overused' or 'underused' in a text compared to a reference corpus, these words are then marked as key and hence potentially interesting and worthy of further study. One of the earliest large-scale studies using key word analysis was that of Hofland & Johansson (1982), who compared British and American English using the Brown family of corpora. Later studies have looked at the difference between native and non-native language in learner corpora (Granger & Rayson, 1998) and language change over (modern) time (Baker, 2009).

Surprisingly, there are relatively few studies of historical data utilising key word analysis; notable exceptions include studies of classic English literature (Archer *et al.*, 2009; Culpeper, 2002; Mahlberg, 2007), specific genres, such as

letters (Markus, 2002) and courtroom language (Archer, 2006) and specific topics such as swearing (McEnery, 2006). Interestingly, in his key words study, Culpeper (2002) chose to use a modern edition of Shakespeare's *Romeo and Juliet* (Craig, 1914) rather than use a text which would be closer to the original orthography, such as from the First Folio. This decision was made to avoid as much spelling variation as possible, not least because "spelling variation is perhaps the greatest obstacle in the statistical manipulation of historical texts" (Culpeper, 2002: 14). Many scholars (e.g. Archer *et al.*, 2009; Markus, 2002) have noted that the spelling variation found in historical texts has a detrimental effect on key word analysis (and other corpus linguistic techniques); here, the degree of this effect will be quantified.

In order to discover any effect caused by spelling variation, key word lists needed to be formulated before and after spelling variation is removed; thus, any change in the key word list rankings indicates an effect of spelling variation. As discussed throughout this thesis, producing versions of texts or corpora with spelling variation removed is no simple task; except for very small samples, manually normalising texts is an exceedingly time-consuming process. Fortunately, for this study a version of the Innsbruck Letters Corpus (the original version was also used in Section 3.1) has been made available which has been normalised and, importantly, manually checked. The process of initial normalisation is explained by Markus (2000) for Middle English (see also Section 2.3.1). The normalised corpus contains parallel line pairs; the first line in each pair contains the original text, the second line contains a normalised version of the first line with any spelling variants replaced with modern English equivalents, for example:

```
$I schepyng at thys day, but be the grace of God I am avysyd
$N shipping at this day, but by the grace of God I am advised
```

The corpus was split into two parts, one containing just the original text lines (`$I`), the other containing the normalised equivalent lines (`$N`). This resulted in two separate corpora on which a key word analysis could be completed, and the differences between the lists analysed. As both corpora were equivalent except for the spelling variation, any difference between key word lists can be attributed to the spelling variation alone.

For this study, log-likelihood (Dunning, 1993) was used to identify key words, this is calculated as follows. For each corpus (i)[13], the total number of words (tokens) is counted (N_i). Then, for a given word, the raw frequency in each corpus is observed (O_i). Expected values (E_i) are also required, which are calculated as shown in Equation 3.1.

$$E_i = \frac{N_i \sum_i O_i}{\sum_i N_i} \tag{3.1}$$

The log-likelihood (G^2) can then be calculated as shown in Equation 3.2.

$$G^2 = 2 \sum_i O_i \ln \left(\frac{O_i}{E_i} \right) \tag{3.2}$$

The higher the value of G^2, the more significant the difference between the observed frequencies (or the more key); at 3.84 the difference is significant at $p < 0.05$, at 6.63 the difference is significant at $p < 0.01$. Overuse or underuse of a key word can be determined by comparing the relative frequency of the word in each corpus (i.e. a relative frequency greater than that found in the reference corpus indicates overuse).

The BNC Written Sampler was used as a reference corpus[14], with both the original and normalised versions of the Innsbruck corpus being compared against the reference corpus in turn. WMatrix (Rayson, 2009) was used to produce the key word lists. Any word with a log-likelihood greater than or equal to 6.63 ($p < 0.01$) was considered key, both overused and underused words were considered. Any word with an observed frequency less than 5 in either the Innsbruck Letter Corpus (before or after normalisation) or the BNC Written Sampler was removed from the key word list as it was important that both lists contained the same set of words so that the comparison showed the effect on key word list ranks, not

[13] In the study described, one corpus was compared against a single reference corpus in turn, hence i will take the values 1 or 2.

[14] Although clearly not the best match as a comparable corpus since it is from a different time period and design to the historical corpora, this effect will be minimised since the same reference corpus is being used for both before and after normalisation corpus comparisons. For more details about the BNC Sampler, see `http://www.natcorp.ox.ac.uk/corpus/index.xml.ID=products#sampler`.

the number of extra variants appearing in the original list. After this filtering process, two key word lists remained; one representing the original corpus and the other representing the normalised corpus, each containing the same list of words along with their log-likelihood value representing each word's keyness in its parent (original or normalised) corpus. The hypothesis was that whilst there will be some similarity between the key word list rankings from the original corpus and the normalised corpus due to them originating from the same texts, a large deviation in the rankings was expected; therefore showing a degradation in accuracy due to spelling variation. The aim here was to prove this hypothesis and quantify the amount of deviation.

In order to calculate the difference between the two key word lists, rank correlation was used. Rank correlation measures the correspondence between two different rankings on the same set of items and returns a value between -1 and 1; -1 is returned if one ranking is the exact reverse of the other, 0 is returned if the rankings are completely independent and 1 is returned if the two rankings are exactly the same. For this study, two rank correlation statistics were used: Spearman's Rank Correlation Coefficient (Spearman, 1904) and Kendall's Tau Rank Correlation Coefficient (Kendall, 1938).

The first stage was to produce a set of log-likelihood observation pairs, these were created by performing a look-up of the log-likelihood values from both lists for each word. Both rank correlation statistics convert the log-likelihood values into ranks; that is every word will have a rank associated to it representing where the word appears in each list sorted descending by log-likelihood. For Spearman's Rank Correlation Coefficient the differences (d_i) between each word's ranks are calculated, then the coefficient (ρ) is given as shown in Equation 3.3, where n is the number of observations.

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{3.3}$$

Kendall's Tau Rank Correlation Coefficient works slightly differently in that it looks at the difference between each possible pairing in one list, if the sign of this difference (whether it is greater than, equal to, or less than 0) is equal to the sign of the difference between the same pair in the other list a concordant pair is

counted (n_c), otherwise a discordant pair is counted (n_d). The coefficient (τ) is then calculated as shown in Equation 3.4, with n again representing the number of observations.

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)} \tag{3.4}$$

Both rank correlation statistics were calculated on the paired log-likelihoods as described above using the R statistics package (Ihaka & Gentleman, 1996). The results are shown in Table 3.4. Both coefficients show that whilst there is some correlation between the two key word lists, there is a definite difference between the rankings of the normalised version's key word list and the original version's key word list. We can therefore confirm our original hypothesis (i.e. a deviation in the rankings of some key words) and conclude that spelling variation does have an effect on key word analysis of the Innsbruck Letter Corpus.

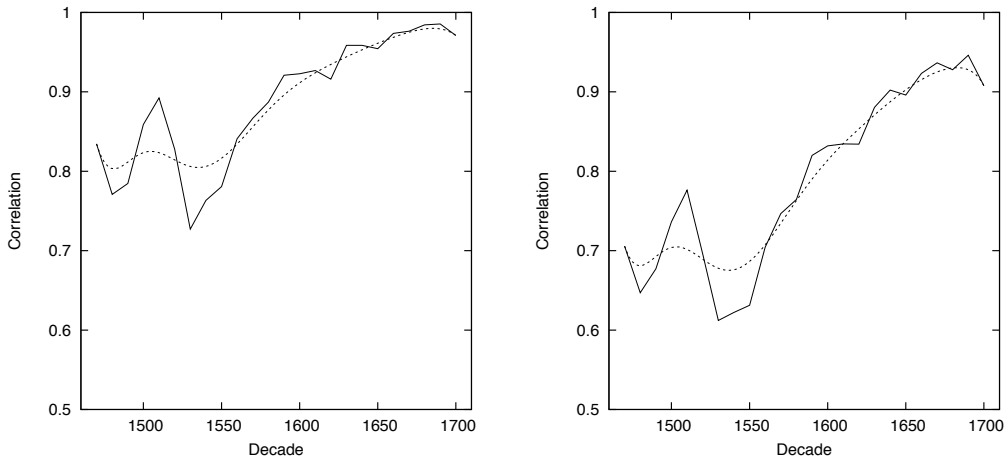
Rank Correlation Method	**Score**
Spearman's Rank Correlation Coefficient	0.7045437
Kendall's Tau Rank Correlation Coefficient	0.5304464

Table 3.4: *Rank correlation coefficients found when comparing the original and normalised versions of the Innsbruck Letter Corpus.*

In order to further show the effect of spelling variation on key word analysis, the study was extended to analyse key word lists before and after normalisation with samples from different EModE time periods. The hypothesis was that there would be more differentiation between the key word lists for samples that represent the earlier centuries of the EModE period, due to the greater levels of spelling variation evidenced at that time (as shown in Section 3.1). As with the key words analysis of the Innsbruck Letter Corpus, both original and normalised versions of a corpus were required, this time sampled at regular intervals throughout the EModE period. The EEBO corpus was chosen as it covers the EModE period and has enough texts available per decade to build a large sample - the same decade samples used in Section 3.1 were utilised. Unfortunately, unlike the Innsbruck Letters corpus, normalised versions of these samples were not available for study. However, in order to detect a trend, it was deemed that automatically (partially) normalised samples were sufficient to detect a trend over time. An

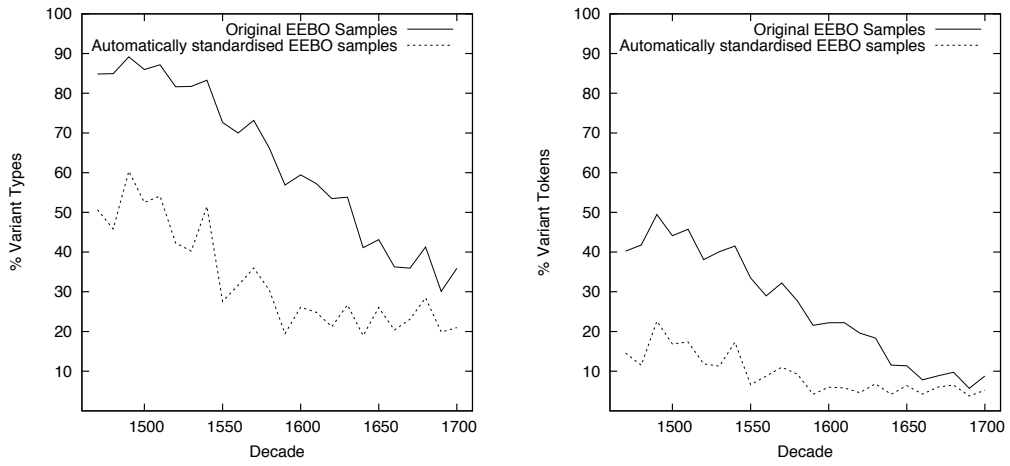
early version of the VARD 2 normalisation tool (Section 4.3) was used for this purpose, creating a version of each sample with modern equivalents automatically inserted for detected variants where possible. For each decade, a similar set of data to the Innsbruck Letters corpus data used above was now available, albeit with only partially normalised texts.

Key word lists for each decade sample were produced in the same way as above using WMatrix and filtered as before. R was again used to calculate Spearman's Rank Correlation Coefficient and Kendall's Tau Rank Correlation Coefficient for each decade sample. The two coefficients are plotted in Figure 3.5, with the dotted lines showing the average trend. The two graphs show erratic results for the earliest decade samples. This is mirrored, to some extent, in the variant rates shown in Figures 3.1–3.4. This can be explained by examining the samples, especially that for 1510–19, a local maximum in Figure 3.5. The sample for 1510–19 contains a large section of foreign translations, containing many different languages. It is not possible to normalise the majority of this section, and so the normalised version will be more similar to the original version. This is shown in Figure 3.6, where the amount of spelling variation remaining after automatic normalisation is both higher and more erratic for the earlier decade samples.



(a) Spearman's Rank Correlation (b) Kendall's Tau Rank Correlation

Figure 3.5: *Graphs showing the rank correlation coefficients comparing EEBO decade samples' key word lists before and after normalisation.*



(a) Types %

(b) Tokens %

Figure 3.6: *Graphs showing the frequency of spelling variants in the EEBO samples before and after automatic normalisation.*

Noise in corpora of this nature is unavoidable and will have an influence on the results, also the effect of spelling variation is underestimated due to spelling variation still remaining (shown in Figure 3.6). However, the general upwards trend can be clearly seen for both coefficients, indicating an increase in correlation between the two key word lists the later the decade of the sample. We can conclude that a reduction in spelling variation over time produces less effect on key word analysis, thus proving our hypothesis.

3.3 Spelling Variation Characteristics

In Section 2.2 various issues related to modern spellchecking were introduced, this was the first step in tackling *RQ 2* (Section 1.2), which asks what the characteristics of EModE spelling variants are and how these affect the application of modern spellchecking techniques. The next step in answering *RQ 2* is to examine how EModE spelling variants compare to spelling issues found elsewhere in terms of some of these characteristics, particularly those highlighted in Section 2.4. This facilitates the application of modern spellchecking methods to EModE spelling variants, which will be described in Chapter 4.

The issue of real-word spelling variants will be analysed first and foremost as this is a key issue in most spelling related problems. The focus will then turn to various specific character-level spelling features; namely, spelling variation position, character edit patterns and the levels of edit distance.

3.3.1 Real-Word Spelling Variants

In Section 2.2.4 the issues related to real-word errors in modern spellchecking were introduced, and whilst many potential solutions have been researched, it is fair to say that real-word errors still cause considerable barriers to many areas of natural language research and applications. As highlighted previously, whilst it is likely that a similar issue will present itself when dealing with EModE spelling variation, it is unclear how much of an issue it may be. To establish the extent of the problem, here the levels of *real-word spelling variants* in EModE texts shall be quantified; that is, the number of spelling variants which happen to match another modern dictionary word which clearly is not a modern equivalent to the intended word. By comparing the levels found to the levels of real-word errors found in studies of other forms of spelling errors and variation, the relative extent of the problem shall be known.

Real-word errors, or in this case real-word spelling variants, are notoriously difficult to locate automatically due to their very nature, especially when just looking at single words out of context. Fortunately, manually normalised EModE texts are available for study which contain marked-up normalisations, some of which may be normalisations of real-word spelling variants. Here we utilise three sources of manually normalised (or checked) texts already used: the Shakespeare and Lampeter samples manually normalised for the study detailed in Section 3.2.1 and the automatically normalised and manually checked Innsbruck corpus, as used in Section 3.2.2. In order to count real-word spelling variants, pairs containing the original form and the modern equivalent chosen for each normalisation made were required. For the Shakespeare and Lampeter samples this was fairly straightforward as the original form and normalised form were present in XML tags within the text, for example:

```
<replaced orig="companie">company</replaced>
```

Creating a list of the original variants simply involved searching for the "replaced" tag and collecting the contents of the "orig" attribute. For the Innsbruck corpus, collecting the original variants was a more complicated process due to the original and normalised versions of the text only being paired on a line-by-line basis. In order to access the individual normalisations, a series of scripts were written to semi-automatically combine the pairs of lines into single lines, similar to those found in the Shakespeare and Lampeter samples, with the original and normalised versions of each word appearing in single XML tags. Thus, the example:

```
$I schepyng at thys day, but be the grace of God I am avysyd
$N shipping at this day, but by the grace of God I am advised
```

becomes:

```
<replaced orig="schepyng">shipping</replaced> at <replaced
orig="thys">this</replaced> day, but <replaced orig="be">by
</replaced> the grace of God I am <replaced orig="avysyd">
advised</replaced>
```

A list of the original variants could now be extracted in the same way as with the manually normalised samples.

In order to determine the proportion of spelling variants which were real-word spelling variants, each spelling variant was compared to a modern word list in a similar process, and using the same word list, as in Section 3.1. Any spelling variants which appeared in the modern word list were counted as real-word spelling variants. In some cases, two (or more) words may have been normalised to a single word, for example:

```
<replaced orig="to morrow">tomorrow</replaced>
```

In these cases both words were checked against the modern word list and only if both strings (or all strings) were present was a real-word spelling variant counted. From this real-word spelling variant count a percentage of spelling variant tokens which are real-words can be calculated. A real-word spelling variant type percentage can also be calculated by only counting each instance of a spelling variant once when it appears multiple times in the text being analysed. The

	Spelling variants		% which are real-word spelling variants	
	Tokens	**Types**	**Tokens**	**Types**
Shakespeare	959	573	10.32	8.38
Lampeter	262	161	9.54	4.35
Innsbruck	43,740	13,520	11.58	5.95

Table 3.5: *Real-word spelling variant rates found in normalised EModE corpora.*

results are shown in Table 3.5 for all three sources. Examples of real-word spelling variants found include: *bee [be]*, *dye [die]* and *then [than]*.

Across all three normalised corpora, the real-word spelling variant ratio is fairly low. This reveals that, at least for the corpora analysed, when a spelling variant exists in an EModE text, only around 10% of the time will the variant be a modern dictionary form (limited to the modern word list used in the study). To put the results in context with other spelling-related issues, previous research, also discussed in Section 2.2.4, has shown real-word error rates for modern spelling errors to be higher, and in some cases much higher. Peterson (1986) found that between 2% and 16% of generated typing errors would be real-word errors depending on the size of the word-list used for detecting real-words. Mitton (1987) found much larger levels, with 40% of a large range of spelling errors being real-words. More recently, Pedler & Mitton (2010) found that 31.4% of spelling errors in a corpus of dyslexic writers were real-words.

In addition to the previous research, the above procedure used to detect real-word spelling variants in the EModE texts was also used on modern spelling errors. This allows for a direct comparison of results as the same method and word list was used in each case. Two manually normalised corpora were sourced. The first was a small corpus, approximately 47,000 words, of child language with spellings manually corrected (see Pooley *et al.*, 2008). The second was a corpus of essays, approximately 154,000 words, by learners of English from another mother-tongue background (French, German and Spanish), again with spelling errors manually corrected (see Lefer & Thewissen, 2007; Rayson & Baron, 2011). As the corrections in the two corpora were tagged in a similar format as with the EModE data, real-word errors could be counted as before. The results, shown in

Table 3.6, again show considerably higher rates of real-word errors compared to the EModE real-word spelling variants.

	Spelling errors		**% which are real-word errors**	
	Tokens	**Types**	**Tokens**	**Types**
Child language	3,802	2,791	26.41	22.10
Second language (French)	351	293	34.48	28.28
Second language (German)	432	350	43.52	43.30
Second language (Spanish)	982	764	29.23	21.85

Table 3.6: *Real-word error rates found in learner corpora.*

The relatively low levels of EModE real-word spelling variants observed is an important result as it reduces the necessity to take into account context in order to detect EModE spelling variants, as the large majority can be found through comparison with a modern word list alone. Whilst not being able to automatically identify real-word spelling variants will have a limiting factor in terms of how many normalisations can be successfully made, this must be balanced with the likely effort required to use context to detect such variants and the quite low success rates observed in previous research with modern real-word spelling error detection (as detailed in Section 2.2.4).

3.3.2 Character Level Variation

In this section, various properties of EModE spelling variants at the character level will be investigated. To achieve this, actual EModE spelling variant normalisations will be analysed with the normalised form compared to the original form and the specific character changes counted in terms of what characters have changed, what position in the variant changes have been made and how many character changes are required to normalise the original variant. A web-based tool, named DICER (Discovery and Investigation of Character Edit Rules), has been developed which can analyse pairs of spelling variants and their normalisations to produce quantities for the properties being investigated. Given a variant string and its normalised equivalent, DICER locates the differences between the two strings and counts how many differences there are, where the differences occur and

precisely which characters are changed. Given a number of these pairs, quantities are summed and stored in a database for analysis. DICER will be described in greater detail in Section 4.4.

Pairs of EModE spelling variants and their normalisations were taken from three sources. Firstly, the automatically normalised and manually checked Innsbruck corpus was used again, with variant–normalisation pairs taken from the processed texts as described previously (in Section 3.3.1). Secondly, a new source was utilised in the form of manually normalised samples from the Early Modern English Medical Texts (EMEMT) corpus (see Lehto *et al.*, 2010), variant normalisations were marked as with the Shakespeare and Lampeter samples used previously in this chapter[15]. Finally, the list of "known variants" used in the initial version of VARD (Rayson *et al.*, 2005) (introduced in Section 2.3.1) was parsed, this contained just pairs of variants and their modern equivalents[16]. It is worth noting that this is not running text like the other sources used here, hence, any statistics for the variants list are based on types and not tokens. In addition, the child language and second language spelling errors[17] (as used in Section 3.3.1) provided a basis for comparison to forms of modern spelling variation where appropriate (similar results from previous research will also be used for comparison). Each set of variant–normalisation pairs were analysed by DICER in turn and the resulting quantities examined. The data available for analysis is summarised in Table 3.7[18]. The separate analyses and findings are detailed in the following three subsections.

[15]The Shakespeare and Lampeter samples were not used in this section as they were considered to contain too few normalisations to detect the specific trends being analysed.

[16]The list has been cleaned up somewhat since its original use with numerous erroneous entries deleted or edited.

[17]The French, German and Spanish texts were combined for these analyses due to the small size of each language sample individually. Whilst this is not ideal, the focus here is not to investigate the influence of mother-tongue on spelling errors – Rayson & Baron (2011), for example, do investigate this.

[18]The quantities given in Table 3.7 differ slightly in some cases from those given elsewhere, this is due to different versions of texts being used and/or variants being detected slightly differently.

Source	Variant–normalisation pairs
Innsbruck	43,579
VARD variants list	44,422
EMEMT samples	5,409
Child language	3,504
Second language	1,458

Table 3.7: *Summary of data available for analysis of character level variation.*

Edit Distance

Edit distance is a measure of how many characters need to change to get from one string to another. In terms of spelling variants, we are interested in how many characters need to change to get from the original spelling to the normalised form. There are many methods to calculate the minimum edit distance between two strings; see Section 2.2.3 (String Similarity Measures) for a discussion. Here, Levenshtein Distance (Levenshtein, 1966) will be used to calculate the minimum edit distance between each pair. The algorithm determines the minimum number of deletions, insertions and substitutions needed to transform one string (the variant form) into another (the normalised form), an example application has been previously given in Table 2.3. For each pair in each dataset the edit distance was calculated and the frequency of pairs exhibiting each level of edit distance recorded. The results of this analysis are shown in Table 3.8.

	Edit Distance (% of pairs)					
	1	**2**	**3**	**4**	**5**	**6+**
Innsbruck	58.12	30.02	8.76	2.15	0.69	0.25
VARD variants list	60.47	26.89	9.82	2.16	0.41	0.25
EMEMT samples	61.86	25.11	9.87	2.26	0.48	0.42
Child language	65.35	23.40	7.62	2.31	0.91	0.40
Second language	74.97	14.20	4.05	1.99	0.75	4.05

Table 3.8: *Edit distances found for pairs of spelling variants and their normalisations.*

Knowledge of typical edit distance can be very useful when evaluating a spelling related problem as it gives an indication of how much effort is generally required for normalisation. Knowing the number of variants with an edit distance of 1 is particularly useful as searching for words which are exactly 1 edit away from a given variant is a common technique when searching for spelling correction candidates (e.g. Church & Gale, 1991). In analysis of the edit distance of EModE spelling variants (as shown in Table 3.8), only around 60% of variants are 1 edit away from their corresponding normalisation. This is a low rate, particularly when compared to rates found in early studies of modern spelling mistakes; 80% of spelling errors in one study (Damerau, 1964), 94% in another study of spelling errors in scientific and scholarly texts (Pollock & Zamora, 1984), although closer to the EModE figures at 69% in a study of spelling from a range of modern sources (Mitton, 1987). Higher rates are also found in the two corpora of learner spelling errors also analysed. Child language errors are only a little higher at 65%, but second language spelling errors are only 1 edit away 75% of the time.

Due to relatively more EModE variants being 2 or more edits away from their correct normalisation, only considering potential normalisations only 1 edit away from the original variant – as is often used in modern spellchecking – would not be a sensible option as this would leave around 40% of variants impossible to normalise automatically. The vast majority of EModE variants in the analysis are within 3 edits of their correct normalisation. Although considering all words which are up to 3 edits away (or even just up to 2 edits away) from a given variant would, on the vast majority of cases, give the correct normalisation as an option, the amount of processing time required and the size of the lists produced would likely be unmanageable. So, whilst low edit distance is a good indication of a potential normalisation being correct, it is not as good an indicator as in modern spellchecking and thus should not be solely relied upon.

Edit Rules

Many spelling error correction methods rely on a set of character edit rules which dictate which specific characters in a variant can be replaced with other specific characters to transform it into the correct normalised form. Patterns to help define these rules exist for many spelling related problems, such as OCR errors, typing errors and human spelling errors; see Section 2.2.3 (Rule Based Approaches) for

a full discussion. DICER can be used to automatically elicit and quantify these patterns, and hence define potential rules. A variant and normalisation pair is analysed by DICER and character differences detected. A rule is mapped by taking account of which characters appear in the variant form and not in the normalised form, and vice versa. The rule may contain any number of characters on each side, including zero. Deletion rules involve having one or more characters in the variant form removed and hence replaced with nothing in the normalised form. Insertion rules add one or more characters to the normalised form where no characters are present in the variant form. Finally, substitution rules replace one or more characters in the variant form with another set of one or more characters in the normalised form. The rules generated and their application on the variant and normalisation pairs (described in the analysis presented here as *rule instances*) are collated into a database which can be viewed through a series of webpages.

The first analysis presented of the rules generated for the five different corpora looks at the rule types present. Table 3.9 shows the number of rules generated and how these are distributed between deletion, insertion and substitution operations. As can be seen, the distribution of rules is similar for all five corpora with the vast majority of rules generated being substitution rules. Deletion rules are slightly more frequent than insertion rules in all cases except for the child language data. Another way of looking at the rule distribution is shown in Table 3.10, where the number of times each rule is generated for a variant–normalisation pair is taken into account. Here, whilst still the majority rule type, the proportion of substitution rules is much reduced – it will be shown in the analysis to follow that this is due to some insertion and deletion rules having high frequencies. Furthermore, the difference between deletion and insertion ratios is more pronounced; now, deletion rules outweigh insertion rules by a considerable amount in all EModE datasets, whilst insertion rules outweigh deletion rules in the learner language datasets. The results for the learner datasets correlate with results found by Mitton (2008: Table 3); out of 11,769 misspellings containing just one simple error, 48%[19] could be corrected with a substitution rule, 33% with an

[19]This also includes transpositions (6%) which are a restricted form of a substitution where two letters are swapped, e.g. *ei* → *ie*.

insertion rule[20] and 19% with a deletion rule. The differences shown in rule type distributions between EModE spelling variants and learner spelling errors begin to make apparent how modern spellchecking techniques cannot be used directly to deal with EModE spelling variation.

	Rules	**Rule Type (% of rules)**		
		Deletion	**Insertion**	**Substitution**
Innsbruck	2,027	6.96	5.77	87.27
VARD variants list	2,104	9.22	8.84	81.94
EMEMT samples	431	11.60	9.05	79.35
Child language	864	5.79	8.80	85.42
Second language	437	12.13	11.67	76.20

Table 3.9: *Rule types found for pairs of spelling variants and their normalisation.*

	Rule Type (% of rule instances)		
	Deletion	**Insertion**	**Substitution**
Innsbruck	27.00	14.50	58.50
VARD variants list	29.65	9.99	60.36
EMEMT samples	35.60	8.17	56.23
Child language	19.02	29.57	51.41
Second language	20.63	23.70	55.67

Table 3.10: *Rule type instances found for pairs of spelling variants and their normalisation.*

Attention now turns to the specific rules which are generated by DICER from the spelling variants and spelling errors found. Tables 3.11–3.15 show the top ten most frequently generated rules in each of the five datasets. Whilst a greater number of rules could be considered for each dataset, this level of detail is out of the scope of this thesis. Here, we consider only the most frequent rules in order to illustrate the differences in edit rules between the historical and modern datasets.

[20]The table given by Mitton shows "Omissions" and "Insertions", these are in terms of how the misspelling is changed from the correct word, i.e. an omission means a letter is missing from the misspelling. The results from DICER are in terms of how the misspelling (or variant) needs to change to successfully correct (or normalise) it.

Rule	**% of total rule instances**	**Top position (% of rule instances)**	**Example**
Delete *E*	17.94%	End (80.47%)	*longe* → *long*
Sub. *Y* → *I*	11.68%	Middle(51.73%)	*thyng* → *thing*
Insert *E*	5.14%	End (57.69%)	*statut* → *statute*
Sub. *U* → *V*	2.24%	Penultimate (48.48%)	*haue* → *have*
Sub. *LL* → *L*	2.04%	End (71.25%)	*perill* → *peril*
Sub. *TT* → *T*	1.74%	End (79.94%)	*thatt* → *that*
Sub. *W* → *U*	1.71%	End (41.87%)	*yow* → *you*
Insert *U*	1.64%	Middle (74.79%)	*noght* → *nought*
Insert *A*	1.62%	Middle (78.09%)	*disese* → *disease*
Sub. *EE* → *E*	1.62%	Middle (47.42%)	*kep* → *keep*

Table 3.11: *Top 10 rules found for pairs of spelling variants and their normalisations in the Innsbruck corpus.*

Rule	**% of total rule instances**	**Top position (% of rule instances)**	**Example**
Delete *E*	18.50%	End (70.10%)	*adde* → *add*
Sub. *Y* → *I*	8.05%	Middle(77.55%)	*chalyce* → *chalice*
Sub. *'* → *E*	6.11%	Penultimate (93.77%)	*startl'd* → *startled*
Sub. *U* → *V*	5.24%	Middle (75.38%)	*grieued* → *grieved*
Sub. *IE* → *Y*	3.37%	End (97.04%)	*privie* → *privy*
Insert *E*	3.26%	Middle (42.15%)	*rarly* → *rarely*
Sub. *LL* → *L*	2.74%	End (61.23%)	*equall* → *equal*
Sub. *V* → *U*	2.33%	Start (93.96%)	*vnpaid* → *unpaid*
Delete *U*	2.07%	Middle (85.55%)	*graund* → *grand*
Sub. *ETH* → *S*	1.63%	End (99.78%)	*eateth* → *eat*

Table 3.12: *Top 10 rules found for pairs of spelling variants and their normalisations in the VARD variants list.*

Rule	**% of total rule instances**	**Top position (% of rule instances)**	**Example**
Delete *E*	26.66%	End (74.48%)	*bodye → body*
Sub. *Y → I*	11.80%	Middle (66.14%)	*strayned → strained*
Sub. *U → V*	5.38%	Middle (54.30%)	*priuate → private*
Insert *E*	3.13%	Middle (55.09%)	*entred → entered*
Sub. *IE → Y*	2.91%	End (93.53%)	*anie → any*
Sub. *LL → L*	2.62%	End (86.19%)	*vitall → vital*
Sub. *TH → S*	2.40%	End (100%)	*cureth → cures*
Sub. *V → U*	2.36%	Start (97.55%)	*vnite → unite*
Sub. *ETH → S*	1.61%	End (100%)	*worketh → work*
Sub. *E → EE*	1.23%	Middle (49.41%)	*Grece → Greece*

Table 3.13: *Top 10 rules found for pairs of spelling variants and their normalisations in the EMEMT samples.*

Rule	**% of total rule instances**	**Top position (% of rule instances)**	**Example**
Delete *[space]*	7.51%	Middle (67.08%)	*my self → myself*
Insert *E*	4.99%	Middle (42.59%)	*safly → safely*
Insert *[space]*	4.92%	Middle (84.04%)	*weare → we are*
Delete *E*	2.70%	Middle (37.61%)	*useing → using*
Insert *A*	2.52%	Middle (78.90%)	*lerning → learning*
Insert ’	2.01%	Penultimate (73.56%)	*dont → don’t*
Insert *U*	1.80%	Middle (71.79%)	*forth → fourth*
Sub. *L → LL*	1.55%	Penultimate (70.15%)	*realy → really*
Insert *H*	1.48%	Second (73.44%)	*were → where*
Insert *I*	1.48%	Middle (87.50%)	*frend → friend*

Table 3.14: *Top 10 rules found for pairs of spelling errors and their corrections in the child language data.*

Rule	**% of total rule instances**	**Top position (% of rule instances)**	**Example**
Sub. - → *[space]*	5.27%	Middle (100%)	*baby-boy → baby boy*
Delete *[space]*	3.23%	Middle (88.89%)	*may be → maybe*
Delete *S*	3.23%	End (81.48%)	*youngs → young*
Insert *[space]*	2.81%	Middle (100%)	*inspite → in spite*
Insert *E*	2.69%	Middle (42.22%)	*lowrs → lowers*
Sub. *[space]* → -	2.33%	Middle (100%)	*full time → full-time*
Sub. *S* → *SS*	2.21%	Middle (83.78%)	*clases → classes*
Delete *E*	2.21%	End (43.24%)	*muche → much*
Sub. *E* → *A*	1.86%	Middle (80.65%)	*Kuweit → Kuwait*
Delete -	1.74%	Middle (100%)	*lay-out → layout*

Table 3.15: *Top 10 rules found for pairs of spelling errors and their corrections in the second language data.*

Previously, only qualitative observations have been given for EModE spelling variant patterns in terms of character differences. Fisher (1977), for example, states some typical patterns observed: the interchangeability of *i / y*, *u / v* and *ou / ow* and the inconsistency in the presence of a final *e*. The DICER analysis shows that these patterns are present in the three EModE datasets analysed (Tables 3.11–3.13). Firstly, the inconsistency in the presence of a final *e* is corroborated with "Delete *E*" being the top rule and also most frequent rule by a considerable amount in all three analyses, the rule is also mostly present at the end of words. This high occurrence shows that *E* was consistently appended to words; this is likely to be an intentional variation added by printers to ease line justification (Potter, 1969: 40). "Insert *E*" is also present (3rd, 6th and 4th respectively) in all three EModE analyses, although more commonly found in the middle of words for the VARD variants list and EMEMT data[21]. The interchanging of *I* and *Y* is also apparent in the three datasets with the rule "Substitute *Y* → *I*" appearing as the second most frequent rule in all three

[21] A high proportion of rule instances were also present at the end of words for both: 26.30% for the VARD variants list and 36.11% for the EMEMT data.

EModE datasets[22]. The interchanging of *u / v* is also represented in all three EModE datasets; the rule "Substitute $U \rightarrow V$" appears 4th in the Innsbruck and VARD variants list analyses and 3rd in the EMEMT analysis, whilst the converse "Substitute $V \rightarrow U$" appears in the VARD variants list and EMEMT analyses in 8th position[23], with the vast majority of rule instances occurring at the start of the variant. Finally, the interchanging of *ou / ow* is also represented, but to a lesser degree. In the Innsbruck analysis the rule "Substitute $W \rightarrow U$" appears in 7th position. With further examination using DICER, one can observe which characters most commonly precede and follow a given rule. Using this, it can be observed that *O* precedes the "Substitute $W \rightarrow U$" rule in 88.92% of cases, somewhat corroborating with Fisher's observation. In the VARD variants list analysis, the "Substitute $W \rightarrow U$" rule appears only 50th, with *O* preceding in 55.5% of cases. In the EMEMT analysis, the rule does not appear until position 75, and with only a third of cases being preceded with *O*. Evidence of *ou* in place of *ow* is also present in all three analyses, but with much less frequency. Why this particular pattern is not present in the DICER analysis to the same degree as others observed by Fisher is open to debate; one potential hypothesis may be that it is a feature of a particular text type (such as personal letters, as in the Innsbruck corpus) or of a particular time period.

The DICER analysis also gives rise to new patterns in EModE spelling variation which may not have been noted previously. Examples include: "Substitute $LL \rightarrow L$" being in the top 10 for all EModE analyses, as well as "Substitute $IE \rightarrow Y$" and "Substitute $ETH \rightarrow S$" being present in the top 10 of the VARD variants list and EMEMT analyses. Furthermore, looking further down the lists, past the top 10, it is apparent that the singling and doubling of many characters is a consistent theme – there are many more examples of such patterns occurring in two or more lists, such as the interchanging of vowels.

Comparing the analyses of the learner datasets (Table 3.14 and Table 3.15) to the EModE datasets as a whole, there is little overlap; only 4 rules from the two learner analyses appear in the EModE analyses. This further shows that the

[22]The converse rule ("Substitute $I \rightarrow Y$") is also present in all three datasets, but not in the top 10, appearing 74th in the Innsbruck analysis, 40th in the VARD variants list analysis and 16th in the EMEMT analysis.

[23]The rule also appears in the Innsbruck analysis, but only in 22nd position.

characteristics of EModE spelling variation are very different to those of modern spelling errors, to which the majority of spellchecking techniques are aimed.

Whilst the EModE analyses show some homogeneity, particularly in the most frequent (top 5) rules – of the 17 amalgamated top 10 rules (from all three EModE analyses), 5 appear in all three analyses, with a further 3 appearing in at least two – there are definite differences between individual analyses; for instance, the Innsbruck analysis features 5 rules in its top 10 which do not appear in either of the other EModE analysis' top 10. This may indicate that different text types (e.g. personal letters) and corpora contain specific rules which characterise the spelling variation within them. One conclusion from this may be that building a static solution to normalising EModE spelling variation as a whole would not be realistic and that a dynamic solution which can be adapted to different EModE corpora would be more desirable.

Edit Position

As shown for the edit rules, the position in a variant at which a rule is actioned is also recorded and quantified in the DICER analysis. Table 3.16 shows the distribution of rule instances (a variant–normalisation pair generating a rule) applied on the first characters (start), beginning at the second character, in the middle, beginning at the penultimate character and on the last character(s) (end) of variants.

	Rule position (% of rule instances)				
	Start	**Second**	**Middle**	**Penultimate**	**End**
Innsbruck	6.55	11.72	30.92	14.41	36.40
VARD variants list	6.51	6.88	37.06	17.77	31.79
EMEMT samples	6.70	8.42	30.51	14.67	39.70
Child language	7.74	13.37	46.65	17.46	14.78
Second language	7.18	8.44	56.67	8.56	19.15

Table 3.16: *Positions of rule instances found for pairs of spelling variants and their normalisation.*

The most common position for rule edits on EModE spelling variants is on the last character(s) of the variant, with the middle of variants the second most

commonly edited position (the VARD variants list has middle of the variant slightly ahead of the end). This is quite different to the learner spelling error datasets which have the middle position quite far ahead as the most frequent position; again, this shows the differences between EModE spelling variants and modern spelling errors. Conversely, the start of the word is consistently, in both the EModE data and learner data, the least frequent edit position. This result is particularly interesting as many modern spellchecking techniques rely upon the first letter being correct (Yannakoudakis & Fawthrop, 1983b), particularly phonetic matching techniques such as Soundex (see Section 2.2.3: Phonetic Matching Algorithms). As the proportion of rule instances in the EModE analyses needing to act on the first character is low (even lower than the modern spelling errors found in the learner datasets), it follows that, for EModE spelling variation, the first character, in the majority of cases, does not need to be edited for normalisation. Hence, it may be possible that the assumption that the first letter is correct, as commonly exploited in modern spellchecking, may carry over to EModE spelling variation.

3.4 Chapter Summary

The aim of this chapter was to gain a better understanding of the problem of EModE spelling variation. This has been achieved through several quantitative analyses, which have verified trends and features previously only reported as qualitative observations (at least on the scale shown here), and identified new patterns and observations to aid in the development of a normalisation solution.

The extent of the problem has been evaluated, with high levels of spelling variation observed in several EModE corpora, with increased variation found in earlier decades. The effect of this spelling variation on the application of corpus linguistic methods has also been evaluated for both Part-of-Speech tagging and key word analysis. It has been shown that EModE spelling variation has a considerable impact on the results of automated tools. These evaluations have addressed *RQ 1* (Section 1.2), highlighting the necessity of an EModE spelling normalisation solution.

In Chapter 4 the application of modern spellchecking techniques to EModE spelling variation will be explored. In order to inform the decisions made when

applying these techniques, various characteristics of EModE spelling variation have been investigated. The key area of real-word errors (or in this case, real-word spelling variants) has been explored, with it being shown that, in two manually normalised EModE corpus samples and one automatically normalised and manually checked full EModE corpus, the levels of real-word spelling variants are substantially lower than real-word error rates found in numerous modern spelling error sources. This is an important result, as dealing with real-word errors is a difficult task in modern spellchecking and it is likely that a solution for dealing with EModE real-word spelling variants would be equally challenging. Whilst any developed variant detection procedure must take into account real-word spelling variants, detection using contextual information (as described in Section 2.2.4) is less of a priority due to these findings.

The methods used for spelling normalisation can also be informed by the results given in this chapter. Two well-used 'rules-of-thumb' in modern spellchecking have been scrutinised in terms of EModE spelling variation. First, the assumption that the large majority of spelling errors will be one edit away from the intended word was shown not to apply to the same extent to EModE spelling variants; around 40% of variants were shown to be more than one edit away from their modern equivalent. This will obviously make finding candidate normalisations for spelling variants more difficult. Second, previous research showing that the majority of modern spelling errors are correct in their first character was shown to also be the case for EModE spelling variants, with less than 7% of required character edits applying at the start of a variant. This finding allows for methods utilising this assumption (generally phonetic matching algorithms) to be applied more easily to EModE spelling variation.

Rule-based approaches are common in solving many spelling related issues, such as OCR errors and typing errors. This chapter has shown that such rule-based approaches may also be applicable to EModE spelling variants, with specific character edit rules identified, differing to those found in modern spelling error sources. Whilst the analysis showed overlap between the rule sets found for different EModE spelling variant sources, it was also found that each dataset had its own features and applicable rules. This disparity between corpora highlights the need for an adaptable normalisation solution which can be customised and trained to the characteristics of the texts to be normalised.

The spelling variation characteristics analysed and quantified in this chapter have largely addressed *RQ 2*. Chapter 4 shall complete the fulfilment of *RQ 2* by detailing how the modern spellchecking techniques described in Section 2.2 are applied to EModE spelling variation.

Chapter 4

Normalising Early Modern English Spelling Variation

The problems caused by spelling variation when using corpus linguistic methods to analyse EModE texts have been detailed and analysed extensively in previous chapters. This chapter describes the adaptation of modern spellchecking techniques to EModE spelling variation, which can be used to create a spelling normalisation pre-processor which will insert modern equivalents alongside spelling variants to aid subsequent studies with automated corpus linguistic tools.

Various techniques for both detecting and correcting modern spelling errors have been introduced and several specific characteristics of EModE spelling variation which will influence the application of these techniques have been identified and studied. Here, discussion will focus on the methods chosen for the specific task of EModE spelling normalisation and how these methods will be used together to produce a ranked list of modern equivalent candidates for variants found, which may be used for automatic or manual normalisation. Throughout the chapter, it shall be shown that the detection and normalisation procedures developed are flexible, allowing for customisation and training to tune the normalisation procedure to the corpus being processed. The flexibility extends to allowing a balance to be struck between the recall and precision of normalisation, depending on the user's needs.

The first stage of normalisation will be to identify the spelling variants present in a given text, Section 4.1 will describe how this can be achieved efficiently and accurately, whilst at the same time highlighting likely pitfalls in detection. The

process of finding candidate modern equivalents for these variants and ranking them in terms of how likely they are to be the correct normalisation will be discussed in Section 4.2. Bringing these two processes together into a customisable tool for manual and automatic normalisation shall then be discussed in Section 4.3. A supplementary tool will be described in Section 4.4, which can be used to analyse character edits from previous variant normalisations (analyses from this tool have already been described in Section 3.3.2). The chapter will then be summarised in Section 4.5.

4.1 Spelling Variant Detection

As discussed in Section 2.2.2, the most common method for finding spelling errors in modern spellchecking is through dictionary lookup. For the purposes of finding EModE spelling variants, the same method can be used, as a high majority of spelling variants are not found in a modern word list (or dictionary). This was proved to be the case in Section 3.3.1, where the rates of EModE real-word spelling variants (those found in a modern word list) were shown to be much lower than the rates of real-word spelling errors found in modern data – around 10% compared to 22%–43%. Whilst detecting as many variants as possible is important, adding context-sensitive detection methods would be a very complex process (see Section 2.2.4) and probably not the most efficient use of development time for a maximum 10% increase in overall detection – reaching close to this maximum would also be highly unlikely as rates of real-word spelling error detection are still fairly modest, even in the latest research (Reffle *et al.*, 2009).

This section shall discuss the details of a dictionary lookup methodology which can be used for detecting EModE spelling variants, this will include detailing sources of modern word lists, describing a suitable data structure which can be used for fast lookup and looking at how words can be detected from running text.

4.1.1 Dictionary Source

Any dictionary lookup procedure will obviously rely upon a list of words for reference. As the normalisation procedure being developed shall aim to have all

words normalised to their modern form (as opposed to normalising to a consistent form, regardless of its presence in a modern dictionary), a modern word list is required for spelling variant detection – any word not in the modern word list would be considered a variant. There are various sources from which this list could be created, but it is important to consider the frequency of words chosen. Including too many low frequency words will increase the likelihood of real-word spelling variants, whilst not including high frequency words will increase the likelihood of detecting a variant erroneously. The size of dictionary is also an issue to consider in terms of processing speed and space constraints.

The first source considered for a modern word list was the British National Corpus (BNC), which is available as a set of frequency lists (Leech *et al.*, 2001). The corpus is a valid source due to being relatively modern (collected 1991–1994) and of British English – EModE is comparable to British English, as opposed to, for example, American English, due to the first extraterritorial English not being present until 1776 (American Declaration of Independence) (Lass, 1999: 1). Using the entire BNC word list would be ill-advised due to the problems highlighted above, instead words were chosen which appeared in the corpus at least once per million words and also appeared in 50 out of 100 sectors of the corpus, this range of use is important to avoid words which only appear frequently in a specialised context. This filtering process resulted in a list of 26,097 words being available.

Whilst the BNC word list ensured that the majority of variants were detected, early tests revealed that far too many valid words were also being erroneously marked as variants. Hence, a further source was considered which could supplement the BNC word list. The Spell Checking Orientated Word Lists (SCOWL)[1] created by Kevin Atkinson provides a collection of word lists from a variety of sources which are sorted into separate lists based on frequency and categories (e.g. language variety, upper/lower case, contractions). A user of SCOWL can choose the lists which are suitable for their needs. For our purposes, British words are more appropriate, and less frequent words should be avoided to reduce the likelihood of variants not being detected (due to being real-word spelling variants). The following lists were chosen from SCOWL, the number shown refers to how frequent the words in the list are, e.g. 10 equates to the 10% most frequent words. An example entry in each list is also given.

[1] `http://wordlist.sourceforge.net/scowl-readme`

- british-words.10 (e.g. *flavour*)
- british-words.20 (e.g. *cancelled*)
- british-words.35 (e.g. *enrol*)
- british-words.40 (e.g. *adaptors*)
- british-words.50 (e.g. *canonise*)
- british-words.55 (e.g. *behaviours*)
- english-contractions.10 (e.g. *don't*)
- english-contractions.35 (e.g. *o'clock*)
- english-contractions.40 (e.g. *you'll*)
- english-upper.10 (e.g. *Europe*)
- english-upper.35 (e.g. *April*)
- english-upper.40 (e.g. *Swiss*)
- english-words.10 (e.g. *do*)
- english-words.20 (e.g. *mint*)
- english-words.35 (e.g. *ravens*)
- english-words.40 (e.g. *vegan*)
- english-words.50 (e.g. *sternum*)
- special-roman-numerals.35 (e.g. *xxvii*)

With these lists added to the BNC word list already created, the total number of words present is now 82,573. Whilst this word list can be used by default, it would be desirable for a user to be able to add or remove lists (from SCOWL or any other source) to suit their needs. For example, a user may know that roman numerals will not be present in their corpus. The effect of using different sized dictionaries will be investigated in Section 5.1.

It is worth noting that as the word list is modern, there will be numerous words present which have only entered the English language after the EModE period (e.g. *Jacuzzi*). These words are of little use in variant detection and could possibly introduce real-word spelling variants. However, it is difficult to remove such words without first having etymological data, which could be used, for example, to remove any words which entered the language post-1800.

4.1.2 Dictionary Lookup

Having collected a word list, the next issue to consider is how it shall be stored and accessed. Every word in a text will need to be compared against the word list in order to establish which are spelling variants; therefore, lookup needs to be fast. A common data structure used for dictionaries is a trie (Fredkin, 1960; Pittman, 2007), due to having quick search, insert and deletion operations, and also not having problems with collisions, as with hash tables. A trie is a tree with levels representing successive characters of the (typically string) keys it contains. For a dictionary of words, the root links to a node for every first letter found in the dictionary, each of these nodes contains a subsequent link for every second letter found after the first letter, and so on. The height of the tree is dictated by the length of the longest word in the dictionary. An example with a small list of short words is shown in Figure 4.1, the following words have been added: *ant, any, an, ape, at, a, ban, bat, beep, beer, bee, bet, be, caret, care, cart, car, can, cape, cap, cat, cent.*

A node in the trie can be marked as representing the end of a word, these are shown as underlined in Figure 4.1. All leaf nodes will represent words, but intermediate nodes may also represent words (e.g. in Figure 4.1: *car*, *care* and the leaf node *caret*), whilst some intermediate nodes are only present as a prefix of a longer word (e.g. *cen*). Three operations are required to make the trie usable as a dictionary in a spellchecking type setting; lookup, insertion and removal of words. The algorithms for achieving these three operations shall now be discussed.

Looking up a word in the dictionary trie involves attempting to traverse the trie using each character in the word to choose the next node. If a node is not present for a given character, then the word is not in the dictionary. If the end of the word is reached and a full node path found, then whether the lookup word is

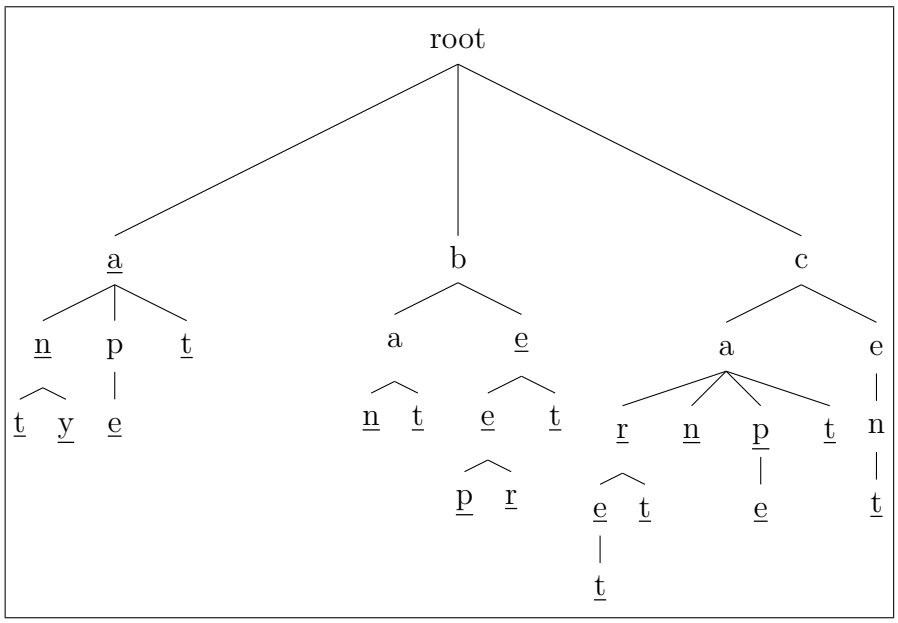


Figure 4.1: *Trie example.*

present depends on the last node representing a word. Pseudocode for a lookup method is shown in Algorithm 4.1.1. The algorithm simply returns a boolean indicating whether the word is present, it could be easily adapted to return more data about the word such as its frequency. As the lookup method depends on the number of characters in the lookup word (in the worst case), it is $O(m)$ where m is the length of the word.

Algorithm 4.1.1: TRIELOOKUP(*string*)

node ← *root*
for each *char* ∈ *string*
do { **if** *char* ∈ *node.next*
  then *node* ← *node.next*[*char*]
  else return (**false**)
return (*node.isword*)

Inserting a word into the dictionary trie involves ensuring a path for the word exists, including adding nodes when not already present for a given character. This is best explained through example. If one wished to add *cane* to the trie shown in Figure 4.1, the trie would first be traversed to the node {*root, c, a, n*}, *e* would then need to be added as child of this node, and finally the new node set to represent a word. Pseudocode for an insert method is shown in Algorithm 4.1.2. Here, true is returned if the word was added, false is returned if the word was already present. The insert method depends on the length of the word being added, hence is $O(m)$ with m being the length of the word.

Algorithm 4.1.2: TRIEINSERT(*string*)

```
node ← root
for each char ∈ string
  do { if char ∉ node.next
       then node.next[char] ← create node
     { node ← node.next[char]
if node.isword = true
  then return ( false )
  else { node.isword ← true
       { return ( true )
```

The final operation required is a removal function. This can take a similar form to the previous two operations, traversing the tree to find a path which represents the word to be removed. Once the path is located, the final node can be set to not representing a word. However, an additional step is required if the final node is a leaf node, as this will mean unnecessary nodes exist in the trie, which could slow future lookups. For example, in Figure 4.1, if *cent* was removed from the trie, the node {*root*, *c*, *e*, *n*, *t*} (a leaf node) would be set as not being a word. Any future lookup of a word with the prefix *cent* would now have to traverse this and intermediate nodes before finding that the word is not present. Instead, the node should be removed from the trie and any intermediate unnecessary nodes also removed; this would mean that a search for a word with the prefix *cent* would end when the *e* is not found as a node following *c*. Algorithm 4.1.3 shows a method for achieving this; as the tree is traversed a node path is stored in reverse order. Once the full path has been found (if present) then each node of the path is looked

back at, starting with the final character of the word (the node which has been set to not represent a word). If the node is found to be a leaf node (i.e. has no child nodes), then the node is removed from the trie. The next node back is then considered in the same way and if the removal of the last node results in this node being a leaf node, this node is also deleted. This is repeated until a node is not made into a leaf node. The algorithm returns true if the word was removed from the trie, it returns false if the word was already not present in the trie. The algorithm is dependent on the length of the word being deleted, and hence, is again $O(m)$ with m representing the number of characters in the word.

Algorithm 4.1.3: TRIEREMOVE(*string*)

```
node ← root
nodepath ← ∅
for each char ∈ string
  do { if char ∈ node.next
       then { nodepath ← node + nodepath
              node ← node.next[char]
       else return ( false )
if node.isword = false
  then return ( false )
  else node.isword ← false
comment: nodepath is reverse path back to root
last ← node
if last.next = ∅
  then { for each n ∈ nodepath
         do { n.next[last.key] ← ∅
              if n.isword = false and n.next = ∅
                then last ← n
                else return ( true )
```

These three operations can be used to insert all of the words from the lists described in Section 4.1.1 into the trie, establish whether words found in a text should be considered spelling variants by looking up the words in the dictionary and marking those not found as variants, and for removing words and adding new words to the dictionary based on user decisions. It is also straight-forward to save the trie to a file in the form of an alphabetical list by performing a preorder depth-

first traversal. Loading the trie from the file is a simple process of performing the insertion operation (Algorithm 4.1.2) with each word in the list.

4.1.3 Tokenization

One further subject to consider, when developing methods for spelling variant detection, is locating the actual words to lookup in running text. For modern corpora, such as the British National Corpus (BNC) XML edition (Burnard, 2007), words are already marked individually (e.g. with `<w>` tag). However, this is unlikely to be the case for the majority of EModE corpora, and most texts will be in *plain text* form; meaning words will need to be searched for within the running text. This is an issue of *tokenization*, which has received much attention, particularly with it being the first stage of many Natural Language Processing applications (see e.g. Grefenstette & Tapanainen, 1994). Efficient methods exist for searching text for strings which match a specific format, regular expressions (Thompson, 1968) for example. However, it is important to consider what should constitute a word. Some possible issues to take into account include:

- Apostrophes may indicate possession or a contraction (e.g. *John's*, *don't* or *'tis*), but may also indicate a quotation.
- Should two words split by a hyphen be considered separately?
- How should digits be dealt with? For example, should *1st* be considered a word?
- Diacritics may be present and should be detected as letters, e.g. *façade*.
- In historical texts especially, alternative characters may be used when a letter is not clear or to indicate formatting such as superscript.
- Letters may be encoded, e.g. þ (thorn) may be represented with a unicode representations such as `&#00FE;`.

A regular expression can be created for searching for words, taking into account any decisions made for the above potential issues. One example pattern could be:

```
([\p{L}\'\-\^~=]|(&[#]?[a-zA-Z0-9]+;))+
```

This regular expression finds words containing one or more characters which are considered letters of a word. The meaning of each part is as follows:

- `[...]` – list of alternative characters.
- `p{L}` – any character in the category 'letter', including diacritics.
- `\'` – apostrophe.
- `\-` – hyphen.
- `\^` – caret.
- `~` – tilde.
- `=` – equals sign.
- `&[#]?[a-zA-Z0-9]+;` – detects unicode entries.
- `(...)` – grouping to separate unicode entries.
- `+` – one or more characters.

Through the use of a regular expression similar to the one above, a text can be searched and all detected words returned. These detected words can then be looked up in the dictionary trie and any words not found in the dictionary marked as potential spelling variants. The next stage is to find candidate modern equivalents for each of these spelling variants, to which our attention now turns.

4.2 Spelling Variant Normalisation

Section 2.2.3 described the main methods used in modern spelling correction, here we shall discuss using these methods to normalise EModE spelling variants. The first stage is to produce a list of candidate normalisations, Section 4.2.1 shall detail this procedure. Section 4.2.2 will introduce several steps which can be used to attach confidence scores, these can then be used to rank the list of candidates. Finally, Section 4.2.3 will show how previous manual normalisations can be used as training data to improve the scoring and ranking of candidates further. The full normalisation procedure shall be summarised in Section 4.2.4.

4.2.1 Producing a List of Candidates

The first stage of finding the correct normalisation for a variant is to create a list of candidates from which the appropriate normalisation can be chosen. As the aim of the normalisation procedure is for all words to be in a single modern form, in order to assist subsequent corpus linguistic techniques (see Section 3.2), candidates should be modern words; hence, the list of candidates can be a subset of the dictionary as defined for detecting variants (Section 4.1). Section 2.2.3 introduced the main methods used in modern spelling correction, here we shall discuss using these methods to select this dictionary subset. The methods chosen and how they are implemented has been influenced by the analysis of EModE spelling variants presented in Section 3.3. As well as a method utilising a list of spelling variants mapped to their modern equivalents, a phonetic matching technique and a rule based approach shall also be described. A further technique, string similarity measures, was described in the background literature, this shall not be used to directly find candidates for two reasons. Firstly, Section 3.3.2: Table 3.8 showed that around 40% of EModE spelling variants were more than one edit away from their appropriate modern equivalent. This makes the technique of finding all modern words which are just one edit away from the variant much less effective than when it is used for finding modern spelling error correction candidates, where the likelihood of the correction being just one edit away is much higher. Secondly, calculating edit distance is generally computationally expensive; hence, comparing a variant to every word in the modern word list to find those which are 1, 2, 3 or 4 edits away (Section 3.3.2: Table 3.8 shows that the edit distance between variants and their normalisations is less than 5 in the vast majority of cases) would substantially increase processing time. However, as low edit distance has been shown to be a useful indication of a normalisation being correct it should still be considered; to this end, a methodology for calculating an edit distance which can influence the ranking of candidates is given in Section 4.2.2. The three methods which are used for collating a list of normalisation candidates for a variant shall now be described in detail.

Known Variants List

The first method for finding candidates is based on the list used for finding and replacing variants in the original VARD tool (Rayson *et al.*, 2005) (see also Section 2.3.1). The original list contained 45,805 variant to modern equivalent mappings, although numerous subsequent corrections and edits of the list has reduced the number of entries to 44,423. To use the list to find candidate normalisations for a given variant involves searching the entries for the given variant and returning the modern equivalent to which the variant is mapped. Examples of entries include: *compriz'd* mapped to *comprised*, *preuente* mapped to *prevent* and *vanish't* mapped to *vanished*. In a small number of cases two (or more) modern equivalents are mapped from the same variant, for example *vvold* is mapped to *wold* and *would*, in these cases all words can be returned as potential candidates. Searching the list of variants is very much like searching for a word in a dictionary, hence, we can use the same method as described in Section 4.1.2. A trie can be built in exactly the same way as for dictionary words, but instead containing spelling variants. For each variant entry added to the trie, a list (usually containing only one entry) of modern equivalents can be stored alongside and returned when a variant is looked up.

Providing the variants list is largely accurate, one would expect usage of the list to achieve high precision, as it is, in most cases, going to offer only one candidate normalisation (or none at all if the variant is not in the list), thus reducing the number of false positives. However, due to the diversity of possible variants in EModE texts (as discussed in Section 2.1.2), it is likely that many normalisations will be missing from the list, hence one would expect recall to be modest at best.

Phonetic Matching

Numerous phonetic matching algorithms exist and they have been used extensively in modern spellchecking and other applications (see Section 2.2.3), the vast majority are based on the original Soundex algorithm developed by Russell (1918; 1922). Phonetic algorithms, particularly Soundex based algorithms, are renowned for having low precision due to many words having the same phonetic code, hence producing a high number of false positives. However, high recall is also generally observed in evaluations due to many spellings of the same word having the same

phonetic make-up. This property makes phonetic matching a desirable method to utilise due to its contrast to other methods, particularly the known variants list, which is likely to produce high precision, but lower recall.

The basic Soundex algorithm will be utilised, the steps for which are given on page 27, with the addition of a small number of simple pre-processing transformation rules. These take into account silent letters and typical letter sequences that can have their basic sound simplified. The transformation rules chosen are those most commonly used in extensions of Soundex, such as those developed by Gadd (1990) and Hodge & Austin (2001b). Whilst a larger list could be used, the list was kept to a minimum to avoid extensive processing times.

- Replace all *DG* with *G* (e.g. *ridge* → *rige*)
- Replace all *GH* with *H* (e.g. *dough* → *douh*)
- Replace all *GN* with *N* (e.g. *reign* → *rein*)
- Replace all *KN* with *N* (e.g. *knife* → *nife*)
- Replace all *PH* with *F* (e.g. *phone* → *fone*)
- Replace all *MB* with *M* (e.g. *dumb* → *dum*)
- Replace all *TCH* with *CH* (e.g. *clutch* → *cluch*)
- Replace initial *PS* with *S* (e.g. *psalm* → *salm*)
- Remove initial *H* (e.g. *hour* → *our*)
- Replace initial *E,I,O,U* with *A* (e.g. *increase* → *ancrease*)
- Remove any characters which are not in the range *A–Z* (e.g. *'tis* → *tis*)

The standard Soundex algorithm can then be applied to produce a code of a single letter (generally the first, unless edited by the transformation rules) and the three digits representing the letter groups. We are able to reasonably rely on the first letter being correct, as the original Soundex algorithm did, due to analysis presented in Section 3.3.2: Table 3.16, which showed that normalisation was only

required at the start of words in less than 7% of cases. This percentage is lower than those observed for other forms of spelling problems.

The algorithm can be used to produce a phonetic code for a given variant. However, this is of little use without having something to compare the code to. The next step is to produce a list of modern words which have the same phonetic code produced for the variant. To achieve this, the same lookup procedure used to search for dictionary words can be employed, namely a trie (see Section 4.1.2). For every word in the modern word list (see Section 4.1.1), a phonetic code can be pre-calculated, these codes can then be inserted into the trie along with a mapping to the original dictionary word. Inevitably, numerous words will have the same phonetic code, hence, for a leaf-node, a list of words matching the trie path's phonetic code can be stored and returned when that code is searched for. An extra step can also be added utilising the known variants list (see above). A phonetic code is also pre-calculated for each variant in the list and inserted into a separate trie along with a reference to the modern equivalent the variant is mapped to. Again, multiple occurrences of the same phonetic code will result in a leaf-node containing a list of modern equivalents, which is returned when that code is searched for.

The full process for producing a list of phonetic matching normalisation candidates for a variant involves the following steps:

1. Calculate the phonetic code for the variant.
2. Search for this code in the dictionary phonetic code trie and store the set of words found (if any).
3. Search for the code in the known variants phonetic code trie and store the set of modern equivalents found (if any).
4. Combine the two sets and return this as the list of candidates.

An example application of this process on the spelling variant *nummed* (modern equivalent: *numbed*) would produce the following results:

1. Phonetic code calculated as *N530*.

2. The following subset of dictionary words which also have this phonetic code are returned: *named, nannied, neonate, ninety, ninth, nomad, nonwhite, noonday, nooned, numbed*.

3. The following modern equivalents mapped from the variants with this phonetic code are returned[2]: *'noint* → *anoint, nam'd* → *named, namde* → *named, nameth* → *names, nineth* → *ninth, ninetie* → *ninety, ninthe* → *ninth, numb'd* → *numbed, numbd* → *numbed, nummed* → *numbed*.

4. These are then combined into the final list of candidates to return: *anoint, named, names, nannied, neonate, ninety, ninth, nomad, nonwhite, noonday, nooned, numbed*.

The final list returned contains twelve candidates, including the appropriate normalisation.

Character Edit Rules

As discussed in Section 2.2.3, rule based approaches are often used to deal with spelling related problems – common character edit rules exist for correcting OCR, typing and human spelling errors. In Section 3.3.2, analysis indicated that character edit rules exist which could be applied to EModE spelling normalisation. Here, we describe the methodology for applying these rules to spelling variants in order to produce normalisation candidates. The algorithms described will be flexible to allow for the use of any set of character edit rules which can have different levels of specificity when applied to variants.

A character edit rule can be defined by three basic properties. Firstly, a search string must be included, which is the set of characters in the variant form which are to be replaced in order for the correct normalisation to be made. The string may be empty, which implies an insertion rule (e.g. *add an e to the end of a variant*). Secondly, a replacement string must also be present, this is the set of characters with which the search string will be replaced to make the normalisation. Like the search string, the replacement string may also be empty, implying a deletion

[2]The variant form is also included here for illustration, but is not strictly necessary.

rule (e.g. *remove e from the end of a variant*). If neither the search string or the replacement string are empty, then a substitution rule is implied (e.g. *replace v with u*). Finally, a position for where in a variant the rule should be applied should also be given. This can be very specific: *start*, *second*, *penultimate* or *end*, or less so with: *middle* (i.e. between *second* and *penultimate* (exclusive)[3]) or *anywhere*. With just these three properties, an extremely large range of character edit rules can be constructed; for example, extra context can be added to a rule by including an additional character in both the search and replacement strings – a specific example would be to restrict the context of the rule *substitute U for W* to be only applicable when the *U* follows an *O*, this would be achieved by adjusting the rule to *substitute OU for OW*.

One important factor to consider when applying a rule based approach is whether more than one character edit rule can be used on the same variant. Many applications using rule based approaches only allow for one rule to be activated on a string at one time; so, for example, a rule could not be used to edit the start of a string and then another rule (or indeed the same rule) used to edit the end of the string. Processing in this manner is much simpler than the alternative of having multiple rules applied on the same string as there is no need to consider rules overlapping in their use or how to find all permutations of applying rules in different combinations. Modern spellcheckers can afford to overlook multiple rule applications, to an extent, due to the relatively high percentage of errors which are only one edit away from the correct spelling. However, the findings presented in Section 3.3.2: Edit Distance revealed that around 40% of EModE spelling variants were more than one edit away from their modern equivalent; therefore, the need for the application of multiple rules on a single spelling variant becomes apparent[4].

The pseudocode for a number of linked algorithms which can be used to find candidate replacements through the application of character edit rules is shown

[3]A variant's middle position may also be considered to be inclusive of *second* and *penultimate* in the specific case of the search string's length being exactly 2 characters shorter than the variant string. For example, the rule *substitute EE with E* in the variant *beeg [beg]*.

[4]Some rules will produce an edit of more than one, e.g. *substitute ie for y*, thus the actual number of normalisations requiring more than one rule application is likely to be lower than 40%. However, informal observations revealed that the majority of cases where the edit distance was more than one also contained more than one rule application.

in Algorithms 4.2.1–4.2.9. As shall be seen, the majority of these methods are required to allow the application of more than one rule on the same variant.

The central method, shown in Algorithm 4.2.1, given a variant string and a set of character edit rules, attempts to apply each rule, in turn, on the variant string. If a rule is successfully applied, the resulting candidate string is looked up in the modern word dictionary using the same method as described in Section 4.1. If found to be a dictionary word, it is added to the list of candidate normalisations to return. As an extra step, the candidate is also looked up in the known variants list, if found then the modern equivalent the variant is mapped to is added to the candidate list. Recursion is then used to attempt to apply the rules to the candidate normalisation again (regardless of it being found in the dictionary), with the candidate string replacing the variant string in re-calling the method. Recursion will continue until each candidate produced has had a further attempted application of each rule on it.

Algorithm 4.2.1: FINDRULECANDIDATES(*string*, *rules*)

```
candidates ← ∅
for each rule ∈ rules
do { candidate ← APPLYRULE(string, rule.original, rule.replacement,
     rule.position)
   { if candidate ≠ ∅
     then { candidates ← candidates
            +FINDRULECANDIDATES(candidate, rules)
          { if TRIELOOKUP(candidate)
            then candidates ← candidates + candidate
            else { variantmapping ← VARIANTTRIELOOKUP(candidate)
                 { if variantmapping ≠ ∅
                   then candidates ← candidates
                    +variantmapping.modern
return (candidates)
```

It is important to avoid the danger of applying a rule to an area of a candidate string which has already been edited from the variant string, as this would mean that a rule is applied (from the original characters to the second replacement characters) that may not exist in the list of defined rules, but more importantly, from a processing point of view, this could also lead to an infinite loop in which a string is replaced and then re-replaced with the original characters and hence replaced again, and so on indefinitely. For example, two rules may exist: *substitute E with EE* and *substitute EE with E*, obviously if these two rules were repeated until they no longer could be applied, then an infinite loop would occur. This problem is averted by keeping track of where rules have been applied in candidates and marking these characters as *used*, this is managed by the methods described by Algorithm 4.2.2 and Algorithm 4.2.3.

Algorithm 4.2.2: IsUsed($string, start, end$)

```
comment: string is checked inclusive of start but exclusive of end
if start = end
  then { if start = string.length
         then return (string.used[start − 1])
         else return (string.used[start])
  else { for i ← start to (end − 1)
         do { if string.used[i]
              then return ( true )
return ( false )
```

Algorithm 4.2.3: MarkUsed($string, start, end$)

```
comment: string is marked inclusive of start but exclusive of end
if start = end
  then { if start = string.length
         then { string.used[start − 1] ← true
                comment: e.g. deletion at end of word
         else string.used[start] ← true
  else { for i ← start to (end − 1)
         do string.used[i] ← true
```

Whether a rule can be applied to a variant (i.e. whether the search string appears in the designated position) is delegated to Algorithm 4.2.4 and Algorithm 4.2.5. If the rule's position is set as *start*, *second*, *penultimate* or *end*, there is, by definition, only one place that the rule can be applied. Algorithm 4.2.4 deals with these cases, first using the IndexOf method (Algorithm 4.2.6[5]) to determine whether the search term appears in the variant/candidate string in the designated position, and that the location has not already been replaced (using Algorithm 4.2.2). If the rule can be applied, the variant string has the search term removed from it and the replacement term added in its place; this is performed by the ReplaceString method, described in Algorithm 4.2.7, which also marks the new candidate string as *used* in the location that has been replaced (using Algorithm 4.2.3). Rules set to be applied in the *middle* of variants or *anywhere* are dealt with by Algorithm 4.2.5. Because the location where these rules can be applied is less restrained, care needs to be taken to ensure that all possible locations for the rule application are considered. To this end, Algorithm 4.2.8 is used to make the replacement (using Algorithm 4.2.7) and then recursively re-applies the rule to the candidate; because the replacement is marked as *used*, if the rule can be applied later in the variant, the rule will be applied there also. Additionally, because the use of the rule on a later position should be considered separately, the *MarkUsed* method (Algorithm 4.2.3) is applied to the variant string without the replacement made, marking the search term's occurrence as *used*. This string is then passed (recursively) to Algorithm 4.2.5 to find other possible applications of the rule. A full list of candidates created by this process is returned for consideration as potential normalisation candidates.

[5]The IndexOf method uses the SearchString method, which is not described in detail as it is similar to a common method supplied in most programming languages. It takes a string to search in, a search term and an index to start searching from and returns the first index in the string, after the supplied start index, where the search term occurs. If the search term does not occur in the string, then -1 is returned.

Algorithm 4.2.4: ApplyRule($string, search, replace, position$)

```
if string.length < 2
  then return (∅)
if position = "start"
  then { if IndexOf(string, search, 0) = 0
         then { candidate ← ReplaceString(string, search, replace, 0)
                return (candidate)

  else if position = "end"
  then { if IndexOf(string, search, (string.length − search.length)) =
            (string.length − search.length)
         then { candidate ← ReplaceString(string, search, replace,
                   (string.length − search.length))
                return (candidate)

  else if position = "second"
  then { if IndexOf(string, search, 1) = 1
         then { candidate ← ReplaceString(string, search, replace, 1)
                return (candidate)

  else if position = "penultimate"
  then { if IndexOf(string, search, (string.length − search.length − 1)) =
            (string.length − search.length − 1)
         then { candidate ← ReplaceString(string, search, replace,
                   (string.length − search.length − 1))
                return (candidate)

  else return (ApplyRuleMultiple(string, search, replace, position))
```

Algorithm 4.2.5: ApplyRuleMultiple($string, search, replace, position$)

```
if position = "middle"
  then { if search.length = (string.length − 2)
           then { if search = SubString(string, 1, (string.length − 1))
                    then { candidate ← ApplyRule(string, search,
                           replace, "second")
                           return (candidate)
         else { index ← IndexOf(string, search, 2)
                if index < (string.length − 2)
                  then { candidates ← FindMultiCandidates(string,
                         search, replace, "middle", index)
                         return (candidates)
  else if position = "anywhere"
  then { index ← IndexOf(string, search, 0)
         if index > −1
           then { candidates ← FindMultiCandidates(string,
                  search, replace, "anywhere", index)
                  return (candidates)
  else return (∅)
```

Algorithm 4.2.6: IndexOf($string, search, from$)

```
start ← from
end ← start + search.length
while (end ≤ string.length)
  do { index ← SearchString(string, search, start)
       if index = −1
         then return (−1)
         else { start ← index
                end ← start + search.length
                if IsUsed(string, start, end)
                  then { start ← start + 1
                         end ← end + 1
                  else return (start)
return (−1)
```

Algorithm 4.2.7: REPLACESTRING($string, search, replace, index$)

$before \leftarrow$ SUBSTRING($string, 0, index$)
$after \leftarrow$ SUBSTRING($string, (index + search.length), string.length$)
$replaced \leftarrow before + replace + after$
MARKUSED($replaced, index, (index + replace.length)$)
return ($replaced$)

Algorithm 4.2.8: FINDMULTICANDIDATES($string, search, replace, position, index$)

$candidates \leftarrow \emptyset$
$candidate \leftarrow$ REPLACESTRING($string, search, replace, index$)
$candidates \leftarrow candidates + candidate$
$candidateagain \leftarrow$ APPLYRULEMULTIPLE($candidate, search, replace, position$)
$candidates \leftarrow candidates + candidateagain$
$markedstring \leftarrow$ MARKUSED($string, index, (index + search.length)$)
$stringagain \leftarrow$ APPLYRULEMULTIPLE($markedstring, search, replace, position$)
$candidates \leftarrow canididates + stringagain$
return ($candidates$)

Algorithm 4.2.9: SUBSTRING($string, start, end$)

comment: string returned is inclusive of *start* but exclusive of *end*

$substring \leftarrow \emptyset$
$substring.used \leftarrow \emptyset$
for $i \leftarrow start$ **to** end
do $\begin{cases} substring \leftarrow substring + string[i] \\ substring.used \leftarrow substring.used + string.used[i] \end{cases}$
return ($substring$)

The methodology described will ensure that each character edit rule is applied to each variant where it is possible for it to do so, and also that if more than one rule is applicable to a variant, all possible combinations of applying these rules are considered. The candidates list is filtered by whether they appear in a modern dictionary, or if a candidate appears in the known variants list, the modern equivalent the variant maps to is added. The performance of this rule based approach, in terms of precision and recall, will be determined by the rule

list used. If many rules are present which are likely to be applicable on many variants, then precision will be lower as an increased number of false positives will be produced. However, conversely, if the rules produce the correct normalisation candidate regularly, then recall will be high. Vice versa, if a small set of specific rules are used, then precision is likely to be improved as the number of candidates suggested, and hence also false positives, will be reduced; but, recall will be lower as the specific rules are likely to be less widely usable in finding the correct replacement.

Pilz *et al.* (2008) mention a list of "manually crafted letter replacement heuristics" for EModE spelling normalisation. This list was created by Dawn Archer at the University of Central Lancashire for use in improving the initial version of VARD (Rayson *et al.*, 2005) and has been made available for use here as the default list. 52 'letter replacement heuristics' existed in the original list, which can all be transformed into character edit rules for use with the methodology described here. Examples of entries in the list are shown in Table 4.1, along with how they are converted into rules to be used with Algorithms 4.2.1–4.2.9 and an example of a variant–normalisation pair to which the rule could be applied. Six further rules have been added to the default list to reflect casual observations made on normalisations the rule list was missing. These are given in Table 4.2.

Heuristic	**Rule Type**	**Search String**	**Replacement String**	**Position**	**Example**
replace initial ‘ with e	Substitution	‘	E	Start	*‘scaped* → *escaped*
add final e	Insertion		E	End	*ther* → *there*
replace final t with ed	Substitution	T	ED	End	*curst* → *cursed*
remove final e	Deletion	E		End	*loude* → *loud*
replace a with e	Substitution	A	E	Anywhere	*clark* → *clerk*

Table 4.1: *Example rules from default list.*

Heuristic	**Rule Type**	**Search String**	**Replacement String**	**Position**	**Example**
replace final ck with c	Substitution	C	CK	End	*publick* → *public*
replace final k with c	Substitution	K	C	End	*Arabik* → *Arabic*
replace final ey with y	Substitution	EY	Y	End	*spicey* → *spicy*
replace final es with s	Substitution	ES	S	End	*beates* → *beats*
replace final th with s	Substitution	TH	S	End	*orderth* → *orders*
remove - from middle	Deletion	-		Middle	*to-day* → *today*

Table 4.2: *Rules added to default list.*

To demonstrate the use of the character edit rules method, an example result is shown below for the variant *manie*, for which the appropriate normalisation should be *many*. The rules applied to the variant are given, as well as details of the extra step from the variants list (noted as *KVL*), where used.

- *Substitute A → O (Anywhere)*: monie, *KVL*: money
- *Substitute E → A (Anywhere)*: mania
- *Substitute IE → Y (End)*: many
- *Substitute A → I (Anywhere)*: minie, *Delete E (End)*: mini
- *Substitute I → E (Anywhere)*: manee, *Delete E (End)*: mane

In this example, five candidates are returned by the method, including the appropriate normalisation.

The candidates returned from the three methods described in this section will be combined into one list of candidates and offered as potential normalisations.

The aim of using the three methods in combination is that they will complement each other and produce a list of likely candidate normalisations whatever the properties of, and reasons behind, the variant are; the phonetic matching technique should find the correct candidate if the spelling of the variant is phonetical, the rule based approach should find the correct candidate if a common letter change is made due to printing decisions or other reasons and, for some sources, potentially allow for OCR errors, and the known variants list will account for unique cases which occur because of other reasons. Of course, there will also be overlap between the methods; some rules are likely to account for phonetic similarity (e.g. the interchanging of vowels) and the known variants list will contain entries that can be found with phonetic matching or the rule based approach. A final example is given in Table 4.3 of a full candidate list returned for the variant *clapd*, which should be normalised to *clapped*. For each candidate listed, the methods which returned it are marked with a tick (✓).

In this example, the phonetic matching algorithm returns many more results (and hence, false positives) than the other methods. All methods return the correct normalisation. Our attention now turns to ranking the combined candidate list in order to ensure that the candidate deemed most likely to be correct is suggested first. Which methods and how many methods found a candidate shall strongly influence its ranking.

4.2.2 Ranking Candidates

For automatic normalisation to take place, a long list of candidate normalisations is of little use without being ranked in some way, as the most likely candidate needs to be chosen. Even for manual normalisation, it would be useful for the candidates to be ranked so that the correct normalisation is more likely to be at the top of the offered list. Here, several steps are discussed with the aim of attaching a useful *confidence score* to each candidate, which will, in turn, allow the list to be ranked. The overall aim of the ranking procedure is to have the correct candidate ranked first with a high confidence score and incorrect candidates given considerably lower confidence scores.

Candidate	Known Variants List	Phonetic Matching	Character Edit Rules	No.
caliphate		✓		1
calved		✓		1
celibate		✓		1
childbed		✓		1
clapped	✓	✓	✓	3
cleaved		✓		1
cleaves		✓		1
cleft		✓		1
clepes		✓		1
clip		✓		1
clipped		✓	✓	2
clopped		✓		1
clubbed		✓		1
clubfeet		✓		1
clubfoot		✓		1
slapped			✓	1
Total:	1	15	3	

Table 4.3: *Candidate list for* clapd *returned from full set of methods.*

Edit Distance

One method introduced in Section 2.2.3, string similarity measures, were not used to find candidates, but can be used to provide an additional method for distinguishing between candidates and aid in ranking. Minimum edit distance is the most commonly used method for calculating string similarity, whereby the minimum number of operations (insertions, deletions and substitutions) required to transform one spelling to another is calculated. Unfortunately, the processing time required to compare a variant spelling to every word in a dictionary makes using an edit distance method to find candidate normalisations unworkable. However, calculating the minimum edit distance between a variant and a small subset of the dictionary is achievable. Such a subset is returned in the form of

a candidate list from the methods described in Section 4.2.1. A minimum edit distance shall be calculated for each candidate in this list, which will be the first stage in calculating confidence scores.

Levenshtein Distance (Levenshtein, 1966) is the most popular method of calculating the minimum edit distance between two words. Pseudocode for calculating the distance is given in Algorithm 4.2.10. The algorithm works by creating a matrix, an example of which is given in Table 4.4 from using Levenshtein Distance to compare *clapd* and *clipped*. In every cell of the matrix the minimum edit distance between the prefixes of the two words up to that point is present. The minimum edit distance between the two full words is therefore present in the bottom-right corner of the matrix; therefore, in the example in Table 4.4, the minimum edit distance is 3.

Algorithm 4.2.10: LEVENSHTEINDISTANCE($s1, s2$)

```
for i ← 0 to s1.length
  do matrix[i][0] ← i
for i ← 1 to s2.length
  do matrix[0][i] ← i
for i ← 1 to s1.length
  do { for j ← 1 to s2.length
       do { if s1[i − 1] = s2[j − 1]
              then cost ← 0
              else cost ← 1
            a ← matrix[i − 1][j] + 1          comment: Deletion
            b ← matrix[i][j − 1] + 1          comment: Insertion
            c ← matrix[i − 1][j − 1] + cost   comment: Substitution
            matrix[i][j] ← MINIMUM(a, b, c) }
return (matrix[s1.length][s2.length])
```

		C	**L**	**I**	**P**	**P**	**E**	**D**
	0	1	2	3	4	5	6	7
C	1	0	1	2	3	4	5	6
L	2	1	0	1	2	3	4	5
A	3	2	1	1	2	3	4	5
P	4	3	2	2	1	2	3	4
D	5	4	3	3	2	2	3	3

Table 4.4: *Matrix created when using Levenshtein Distance to compare* clapd *and* clipped.

One problem with the Levenshtein Distance is that the length of the strings being compared is not taken into account; common sense dictates that an edit distance cost of 2 between two strings of length 4 has more impact than a cost of 2 on two strings of length 10. To solve this problem, the distance can be simply normalised by the length of the two strings being compared. The formula used, shown in Equation 4.1, was considered by (Sampson & Babarczy, 2003) in their study of parsing accuracy. S is the similarity between the two strings being compared: x and y.

$$S = 1 - \frac{distance(x, y)}{length(x) + length(y)} \tag{4.1}$$

This returns a similarity score between 0 and 1; 0 meaning the two strings bear no similarity and 1 meaning the strings are exactly the same. For the example given in Table 4.4, the similarity score (S) between *clapd* (x) and *clipped* (y) would be $1 - \frac{3}{5+7} = 0.75$.

Table 4.5 shows the candidate list from Table 4.3 ranked by the similarity score calculated against *clapd*, the raw Levenshtein Distance (LD) is also given. The correct normalisation is ranked first, so, at least in this example, the similarity score is useful towards reaching a confidence measure to rank on. However, several candidates are still equally ranked and some clearly unlikely candidates have fairly high scores; further evidence is required to contribute to a useful confidence score for ranking.

#	Candidate	LD	S
1	*clapped*	2	0.833
2	*clip*	2	0.778
3	*cleaved*	3	0.750
3=	*clipped*	3	0.750
3=	*clopped*	3	0.750
3=	*slapped*	3	0.750
4	*calved*	3	0.727
4=	*clepes*	3	0.727
5	*cleft*	3	0.700
6	*cleaves*	4	0.667
6=	*clubbed*	4	0.667
7	*celibate*	5	0.615
7=	*childbed*	5	0.615
8	*caliphate*	6	0.571
9	*clubfeet*	6	0.538
9=	*clubfoot*	6	0.538

Table 4.5: *Standardisation candidates for* clapd *ranked by similarity score (S).*

Method Penalties

In Section 4.2.1 it was shown that both the phonetic matching and character edit rule methods use the known variants lists as an extra step for finding some candidate replacements. However, in the final list of candidates produced, whether this extra step was taken is not considered and all candidates are treated equally. Here, we rectify this by imposing a penalty on any candidate found by a two-stage process. This is implemented by attaching a score linking a candidate normalisation to each of the methods used to find candidates. If a method does not find a given candidate, the score is 0; if a method does find the candidate using only one step, the score is 1; and if a method finds the candidate but needs to use two-steps (e.g. phonetic matching via the known variants list), the score is 0.8[6].

[6] 0.2 is chosen as the default penalty to impose, although any other figure (between 0 and 1) could be used in a final implementation.

In the specific case of the character edit rules method, another penalty can also be applied. As described, candidates can be found by applying multiple character edit rules to a given variant. However, all candidates returned are treated equally and how many rules were needed to find the candidate is not taken into consideration. This is rectified by imposing a 0.1[7] penalty for every rule required after the first rule application.

The known variants list method only ever requires one step, so here a score of 1 will be given if the correct variant–normalisation pair is present, or 0 otherwise.

For each candidate, three scores will now be available, these can be used along with the similarity score (edit distance) described above and combined into one score using an average[8] (mean[9]), which could be considered a confidence measure. Table 4.6 shows this process for the list of candidates given for the variant *clapd* in Table 4.3, the entries are ranked by the average score. For this example, the correct candidate is ranked highest and has a score substantially higher than the second placed candidate, there are also fewer equally ranked candidates (compared to Table 4.5). The methods are given as KVL: Known Variants List, PM: Phonetic Matching, CER: Character Edit Rules and ED: Edit Distance (similarity measure). These are the four base methods used throughout the remainder of the research presented in this thesis to find and score a list of candidates[10].

[7] Again, a default penalty to impose is chosen, but any other figure (between 0 and 1) could be used in a final implementation.

[8] An average is used here because no knowledge is available (at this point) to judge if certain methods are likely to be more reliable, and hence given more weight. It shall be shown later that knowledge of a method's performance in previous normalisations can be used to alter the balance between the four scores.

[9] For the four scores (S): $mean = \frac{\sum_i^4 S_i}{4}$.

[10] Whilst edit distance is not used directly to find candidates, it can be considered equivalent to the other three methods because a score is given to each candidate which could be deemed equivalent to the edit distance method's confidence that the candidate is the correct normalisation.

#	Candidate	KVL	PM	CER	ED	Avg.
1	*clapped*	1.0	1.0	0.8	0.833	0.908
2	*clipped*	0.0	1.0	0.7	0.750	0.613
3	*cleaved*	0.0	1.0	0.0	0.750	0.438
3=	*clopped*	0.0	1.0	0.0	0.750	0.438
4	*calved*	0.0	1.0	0.0	0.727	0.432
5	*cleft*	0.0	1.0	0.0	0.700	0.425
6	*clubbed*	0.0	1.0	0.0	0.667	0.417
7	*celibate*	0.0	1.0	0.0	0.615	0.404
8	*clip*	0.0	0.8	0.0	0.778	0.395
9	*caliphate*	0.0	1.0	0.0	0.571	0.393
10	*clubfeet*	0.0	1.0	0.0	0.538	0.384
10=	*clubfoot*	0.0	1.0	0.0	0.538	0.384
11	*clepes*	0.0	0.8	0.0	0.727	0.382
12	*cleaves*	0.0	0.8	0.0	0.667	0.367
13	*slapped*	0.0	0.0	0.7	0.750	0.363
14	*childbed*	0.0	0.8	0.0	0.615	0.354

Table 4.6: *Standardisation candidates for* clapd *ranked by average of method scores.*

Considering Precision and Recall

Up until this point, calculating a confidence score has been solely based upon the individual candidate's score for each method. These scores can be defined as the method's *predicted recall* for the candidate being offered, in that they are estimated probabilities that the candidate is correct for the variant being normalised. The definition can be justified as follows. In classification problems, recall is calculated as $\frac{tp}{tp+fn}$, where tp is *true positives* (how many of the items of a particular class have been classified as belonging to that class) and fn is *false negatives* (how many of the items of a particular class have not been classified as belonging to that class). For a single variant normalisation, $tp + fn$ will always equate to 1 because there is only ever one correct normalisation available for a

given variant.[11] Hence, recall is equivalent to tp alone ($\frac{tp}{1} = tp$). For a predicted recall, whether the candidate is a *true positive* can not be known until the correct normalisation is chosen. However, for each candidate offered, four scores between 0 and 1 link a candidate to each method, these scores could be considered the method's predicted probability that this candidate is correct, or a predicted *true positive* (tp). As recall here is equal to tp, then for each method the *predicted recall* for a candidate is its score.

As well as recall, precision is also important in terms of automatic normalisation, in fact in many cases more so. High precision is key in order to ensure that any normalisations made are, in the vast majority of cases, correct. Otherwise, additional noise would be added to the text, making the problems that spelling variation causes worse. It was shown in Section 4.2.1 that the different methods offer varying numbers of candidates; from the known variants list method which generally returns only one candidate if the variant is present, to the phonetic matching algorithm which often returns high numbers of dictionary words with the same phonetic code as the variant. As only one candidate can be correct, any other candidates can be considered *false positives*[12] (fp). This should be taken into account because if a method is offering just one or two candidates to which it attaches high confidence scores, then this is more useful than a method which offers many candidates with high scores with a preferred candidate hard to distinguish. Precision is calculated as $\frac{tp}{tp+fp}$ in classification problems, this can be used to calculate a *predicted precision* for a method's candidate suggestion. As with recall, the tp can be predicted using the candidate method score. A predicted fp is also needed because whether a candidate is a *false positive* is not known until one of the candidates is chosen as correct. If a tp is predicted by the candidate's method score, it follows that the fp can be predicted by summing the method score for every other candidate offered; as these are the predicted tps for each candidate. $tp + fp$ is the full set of candidate scores offered by the method, hence the *predicted precision* (P) for each candidate (i) can now be calculated as shown

[11] It could be argued that in some cases the appropriate normalisation is ambiguous or would cause disagreement. However, any automatic normalisation procedure can only aim to achieve, as an upper limit, the performance of manual normalisation.

[12] In classification problems the *false positives* rate is how many of the items classified as belonging to a particular class do not belong to that class.

in Equation 4.2, where S is the method score and n is the number of candidates suggested.

$$P_i = \frac{S_i}{\sum_{j=1}^{n} S_j} \tag{4.2}$$

The predicted precisions can be calculated for each of the candidates found for the *clapd* example. These are shown in Table 4.7 along with the scores from Table 4.6 which are now predicted recalls.

#	Candidate	KVL		PM		CER		ED	
		R	**P**	**R**	**P**	**R**	**P**	**R**	**P**
1	*clapped*	1.0	1.0	1.0	0.070	0.8	0.364	0.833	0.076
2	*clipped*	0.0	0.0	1.0	0.070	0.7	0.318	0.750	0.068
3	*cleaved*	0.0	0.0	1.0	0.070	0.0	0.000	0.750	0.068
3=	*clopped*	0.0	0.0	1.0	0.070	0.0	0.000	0.750	0.068
4	*calved*	0.0	0.0	1.0	0.070	0.0	0.000	0.727	0.066
5	*cleft*	0.0	0.0	1.0	0.070	0.0	0.000	0.700	0.064
6	*clubbed*	0.0	0.0	1.0	0.070	0.0	0.000	0.667	0.061
7	*celibate*	0.0	0.0	1.0	0.070	0.0	0.000	0.615	0.056
8	*clip*	0.0	0.0	0.8	0.056	0.0	0.000	0.778	0.071
9	*caliphate*	0.0	0.0	1.0	0.070	0.0	0.000	0.571	0.052
10	*clubfeet*	0.0	0.0	1.0	0.070	0.0	0.000	0.538	0.049
10=	*clubfoot*	0.0	0.0	1.0	0.070	0.0	0.000	0.538	0.049
11	*clepes*	0.0	0.0	0.8	0.056	0.0	0.000	0.727	0.066
12	*cleaves*	0.0	0.0	0.8	0.056	0.0	0.000	0.667	0.061
13	*slapped*	0.0	0.0	0.0	0.000	0.7	0.318	0.750	0.068
14	*childbed*	0.0	0.0	0.8	0.056	0.0	0.000	0.615	0.056

Table 4.7: *Standardisation candidates for* clapd *with predicted recall (R) and predicted precision (P) for each method.*

The next stage is to combine the precision and recall scores to produce one confidence score for each candidate. In Table 4.6, it was shown how a simple average (mean) of the method scores can be used for ranking the candidates. Using the definitions above, this can now be used as an *average predicted recall*.

However, it would not be sensible to use the same method to combine methods' predicted precisions. Doing so would mean that one method's high precision would be negated by another method's low precision. Logically, a candidate's predicted precision should be based on the most specific method which found it. We therefore take the maximum precision score from the four methods as our overall predicted precision for the candidate. We now have an overall predicted precision and predicted recall for each candidate. If all methods find a given candidate, then high recall will be predicted. If only one method finds the candidate then the predicted recall will be low. For predicted precision, this will be high if a method returns the candidate with few other suggestions. If all methods returning the candidate also suggest many other candidates, predicted precision will be low.

In order to rank the candidates, a single combined score is required. An F-Score is commonly used to combine precision and recall scores, it can be calculated as shown in Equation 4.3, with β being a non-negative value which dictates the balance between precision (P) and recall (R).

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R} \tag{4.3}$$

For example, β could be set to 2 and recall would be weighted twice as much as precision, or if $\beta = \frac{1}{2}$, then precision would be weighted twice as much as recall. Setting β to 1 balances recall and precision equally. The simplified equation for F_1 is shown in Equation 4.4, this is equivalent to the *harmonic mean* of recall and precision.

$$F = 2 \cdot \frac{P \cdot R}{P + R} \tag{4.4}$$

In Table 4.8, we demonstrate the use of the F_1 measure, although different values for β could be used in an end system depending on the user's desire to prioritise either precision or recall. For the example shown, the results are a further improvement on those just using the method scores (Table 4.6). The top, and correct, candidate now has a score more than double that of the second placed candidate, other candidates further down the rankings have been given

considerably lower scores[13]. One main difference is the promotion of *slapped* to third ranked due to the character edit rules method having a higher predicted precision than the phonetic matching method.

#	Candidate	Avg. Recall	Max. Precision	F-Score
1	*clapped*	0.908	1.000	0.952
2	*clipped*	0.613	0.318	0.419
3	*slapped*	0.363	0.318	0.339
4	*cleaved*	0.438	0.070	0.121
4=	*clopped*	0.438	0.070	0.121
5	*calved*	0.432	0.070	0.121
6	*cleft*	0.425	0.070	0.121
7	*clubbed*	0.417	0.070	0.120
8	*clip*	0.395	0.071	0.120
9	*celibate*	0.404	0.070	0.120
10	*caliphate*	0.393	0.070	0.119
11	*clubfeet*	0.384	0.070	0.119
11=	*clubfoot*	0.384	0.070	0.119
12	*clepes*	0.382	0.066	0.113
13	*cleaves*	0.367	0.061	0.104
14	*childbed*	0.354	0.056	0.097

Table 4.8: *Standardisation candidates for* clapd *ranked by* F_1*-Score.*

Through the addition of the edit distance similarity measure, penalising methods' scores if they employ extra steps to find a candidate and by predicting and combining recall and precision scores, it has been shown how candidates found through the methods described in Section 4.2.1 can be ranked to allow for the most likely candidate to be offered first and with a high confidence score. This is based on just the method scores for the candidate and the scores for alternative candidates also returned. No prior knowledge, in terms of how successful methods have been for normalisation previously, is taken into account. Another problem

[13]Although for some candidates the scores are equal at the degree of precision used here, there are differences at a higher degree. The candidates which have exactly the same score are marked (=).

is that the rankings rely heavily on how specific a method is in determining its candidates. If a method returns just 1 or 2 candidates with high scores, it is likely that these candidates will be ranked highest. What is not taken into account is how successful a method is likely to be at predicting the correct candidates. It may be the case that a method is particularly bad at predicting the correct candidate (low recall), if the method also only produced a small number of candidates (and with high confidence scores) it is likely that these candidates will be ranked highly. We now move on to adding knowledge of previous normalisations to improve results through training.

4.2.3 Improvement Through Training

The need for a non-static normalisation tool has been highlighted previously through the analysis of EModE spelling variation characteristics in Section 3.3. It was shown that whilst EModE corpora contain many similarities in terms of the spelling variants present, subtle differences between individual corpora were found. A generic EModE spelling normalisation tool would not be able to take these nuances into account. In order to produce a more dynamic normalisation procedure which can be trained to deal with the specific spelling variation trends in a particular corpus, the success of each method in previous normalisations needs to be taken into account when ranking candidates.

Improving the Known Variants List Method

The first training technique described is aimed at specifically improving the performance of the known variants list method. Despite the list used having several iterations of corrections, there will inevitably still be mistakes in the list. These could be problematic if the variants list as a whole becomes a key indicator of the correct normalisation due to it performing well in general. Additionally, the list will be by no means complete; as discussed previously (Section 2.1.2), creating a full list of variants and their modern equivalents would be nearly impossible due to the wide variety of spellings present in EModE corpora. Therefore, being able to add to the variants list through training would be desirable. To account for these problems, the variants list procedure can be extended to include data from previous normalisation made. This can be achieved in a similar manner to that

used to keep track of how successful each full method has been, but instead keep track of each individual entry in the variants list. New variant–normalisation pairs can also be added to the list when encountered.

As discussed in Section 4.2.1 (Known Variant List), the list of variants are stored in a trie (see Section 4.1.2). For each variant in the trie, a list of candidate normalisations is present. For each variant–candidate pair, the candidate can have a usage number incremented (the usage number begins at 1 as the variants list is counted as 1 instance of training). If the normalisation has not been used before for the current variant, a new candidate can be added to the variant's list, with a usage number of 1. If the variant itself is not present in the list, the variant can be added to the trie, with the normalisation as its only candidate (again with usage 1). From a variant's list of candidates and their usage, a total usage for that variant can be gained by summing the list's usage numbers. The candidate's score for the KVL method can now be calculated by simply dividing the candidate usage by the variant usage. As the vast majority of variants in the list only have one candidate present, this score will normally be 1. However, for where there is ambiguity through multiple candidates, the candidate which has been used on more occasions previously will take preference through a higher score. If the candidates have a similar usage number, then their score will be reduced in terms of precision (as before). This will result in erroneous entries in the variants list having less of an impact on normalisation accuracy as the more appropriate normalisation will gain a higher score through training.

As an example, a variant, *clapd*, in the list contains two candidates, *clapped* and *clappd*, with *clappd* being an erroneous entry, which is indicated through *clapped* being present as a normalisation 8 times during training, with *clappd* never seen. Previously, each candidate would receive a precision score of 0.5 for the KVL method, despite the training. The KVL method as a whole may perform well for other variant normalisations, hence the KVL method as a whole could receive a high precision score. With the extension described here, after training the *clapped* candidate would have a usage number of 9, with *clappd* remaining at 1. Thus, the KVL candidate score for *clapped* would be 0.9 $(9/(9+1))$, and for *clappd* it would be 0.1 $(1/(9+1))$. Clearly, this is preferable as the appropriate normalisation, *clapped*, would be ranked much higher than *clappd*.

As the phonetic matching and character edit rules methods also use the variants list to find candidates, the scores produced by the KVL training procedure also need to be taken into account when candidates are found using a combination of methods. This is achieved by simply multiplying the KVL candidate score by the previously used score (e.g. 0.8 for PM, or 0.7 for CER – see Table 4.6). This will ensure that an erroneous entry (as in the example above) will not be offered via two methods with a high score (after training). At the same time, the scores of appropriate normalisations will receive the same scores as before, or very slightly reduced if affected by other entries (as with *clapped* in the example above).

Using Training to Improve Ranking

To determine how successful a method is likely to be at predicting the correct candidate for future normalisations in a particular corpus, the previous normalisations made in that corpus need to be taken into account. During training, each normalisation made can be analysed and the performance of each method assessed in terms of: whether the correct candidate was suggested by the method, and if so, at what confidence score was it suggested; and how many other candidates were also suggested by the method, and at what confidence scores. By keeping track of these figures, a cumulative recall and cumulative precision of each method can be calculated.

The procedure for calculating the cumulative recall and precision for methods is as follows. For each method (KVL, PM, CER and ED), the previously used true positives, false positives and false negatives are incremented for each normalisation in the training data to create cumulative scores. The Cumulative True Positives rate (CTP) equates to how often the correct candidate is suggested by a method. Instead of adding 1 to the figure (as is normally the case in similar methodology in classification problems), we use the confidence score at which the correct candidate was offered by the method. Thus, if a method is consistently predicting the correct candidate with high confidence, CTP will increase quickly, if a method is offering the correct candidate but only with low confidence, CTP will still increase but at a slower rate, if a method never offers the correct candidate, CTP will not increase at all. CTP can be calculated as shown in Equation 4.5, where n is the

number of previous normalisations and S_c is the correct candidate's method score for each normalisation.

$$CTP = \sum_{i=1}^{n} S_c \tag{4.5}$$

The Cumulative False Positives rate (CFP) measures the number of extra candidates offered by a method; for every alternative candidate offered, CFP will be incremented. As with CTP, the confidence score given to each alternative candidate can be added to CFP rather than adding 1 for each. Thus, if a method offers many alternatives with high confidence scores, CFP will increase quickly, if just a few candidates are offered at high scores or many candidates are offered, but at low scores, then CFP will rise less quickly, and if a method offers no alternatives, CFP will not increase at all. A method's FP value for each previous candidate (i) can be calculated as shown in Equation 4.6, where m is the number of candidates offered by the method, S_j is each candidate's confidence score and S_c is the correct candidate's score. CFP can then be calculated as shown in Equation 4.7.

$$FP_i = \left(\sum_{j=1}^{m} S_j \right) - S_c \tag{4.6}$$

$$CFP = \sum_{i=1}^{n} FP_i \tag{4.7}$$

The Cumulative False Negatives rate (CFN) is essentially the opposite of CTP, i.e. how often the correct candidate is not offered by the method. As there is only one correct candidate available for each normalisation, TP_i and FN_i will sum to 1 for each normalisation (this is discussed on page 116), hence $1 - TP_i$ can be added to represent FN_i for each candidate. Thus, CFN is calculated as shown in Equation 4.8.

$$CFN = \sum_{i=1}^{n} (1 - S_c) \tag{4.8}$$

From these three figures, the cumulative recall (CR) and cumulative precision

(CP) of each method can be calculated as shown in Equation 4.9 and Equation 4.10 respectively.

$$CR = \frac{CTP}{CTP + CFN} \tag{4.9}$$

$$CP = \frac{CTP}{CTP + CFP} \tag{4.10}$$

To exemplify the use of cumulative recall and precision, a manually normalised sample of Shakespeare text, as introduced in Section 3.2.1, was parsed for normalisations, with 959 variant–normalisation pairs found. Training was performed by using each variant from these pairs and searching for candidates as described in Section 4.2.1, the edit distance method was also used to attach a similarity score to each candidate found. Using the correct normalisation from each of the 959 pairs, CTP, CFP and CFN were calculated as described, which enabled the calculation of each method's CR and CP. The results of this training procedure are shown in Table 4.9.[14]

Method	CTP	CFP	CFN	CR	CP
KV	816.003	22.997	143.997	0.850	0.973
PM	840.967	46329.646	119.033	0.876	0.018
LR	691.167	1148.783	268.833	0.720	0.376
ED	784.342	30152.169	175.658	0.817	0.025

Table 4.9: *Cumulative recall and precision of methods after training with 959 manual normalisations of Shakespeare text.*

The results show that for the Shakespeare text normalisation, the phonetic matching (PM) method was most useful in terms of recall and the known variants list method was the most useful in terms of precision. All methods had reasonably high recall, but phonetic matching and edit distance (ED) had particularly bad scores for precision[15].

[14]Each of CTP, CFP and CFN begin at 0.5 in order to give CR and CP figures of 0.5. Hence, $CTP + CFN = 959 + 0.5 + 0.5 = 960$.

[15]This is to be expected for phonetic matching with it being notoriously low precision (see Section 2.2.3). This will have a knock on effect with edit distance as all candidates found with phonetic matching will have an edit distance score calculated.

These cumulative recall and precision figures are good indications of how successful the methods will be in finding candidates for other variants in the Shakespeare texts. Hence, the results need to influence the ranking of future suggested candidates. To achieve this, the cumulative recall and cumulative precision scores are combined with the predicted recall and predicted precision scores (as calculated in Section 4.2.2) when calculating the F-Score used to rank candidates. The combined recall (R_Ψ) is calculated as shown in Equation 4.11, with the cumulative recall (CR) and predicted recall (PR) for each method (m). Calculating the combined recall in this manner allows the cumulative recall scores to act as weights; i.e. if a method has a higher cumulative recall, whether a candidate is found by that method and with what score has a larger impact on the candidate's combined recall.

$$R_\Psi = \frac{\sum_{m}^{4}(CR_m \cdot PR_m)}{\sum_{m}^{4} CR_m} \tag{4.11}$$

With the predicted precision, the maximum method precision was taken (see page 118). Here, we use the product of the predicted precision (PP) and cumulative precision (CP) to get a combined precision for each method. We then, as before, take the maximum combined method precision as our overall combined precision (P_Ψ), as shown in Equation 4.12.

$$P_\Psi = \max_m(CP_m \cdot PP_m) \tag{4.12}$$

As before, the combined recall and combined precision can be merged into one score with an F-Score (Equation 4.3), which can be used to rank candidates. Table 4.10 shows the normalisation candidates from the variant *clapd* (the same example used in the previous section) ranked by the F_1-Score, which is the harmonic mean between the combined recall and combined precision. As can be seen, the top, and correct, candidate is now even further away from the other candidates in terms of its confidence score. The majority of the lower-ranked entries now have negligible scores.

With the training technique shown here, the likely precision and recall for the four methods can be set for a particular dataset through training with

#	Candidate	Combined Recall	Combined Precision	F-Score
1	clapped	0.914	0.973	0.942
2	clipped	0.611	0.120	0.120
3	slapped	0.342	0.120	0.177
4	clip	0.410	0.002	0.004
5	clopped	0.456	0.002	0.003
5=	cleaved	0.456	0.002	0.003
6	calved	0.451	0.002	0.003
7	clepes	0.397	0.002	0.003
8	cleft	0.444	0.002	0.003
9	clubbed	0.435	0.002	0.003
10	cleaves	0.382	0.002	0.003
11	celibate	0.423	0.001	0.003
12	childbed	0.369	0.001	0.003
13	caliphate	0.412	0.001	0.003
14	clubfeet	0.403	0.001	0.003
14=	clubfoot	0.403	0.001	0.003

Table 4.10: *Standardisation candidates for* clapd *ranked by combined F_1-Score after training with manual Shakespeare normalisations.*

manually normalised variants. The individual methods are likely to have varying performance levels depending on, for example, text type, genre or date of publication. This training ability allows for the normalisation procedure to be tuned for a particular corpus, increasing the likelihood of the correct candidate being ranked highest and also having a high confidence score, this will in turn lead to more accurate normalisation. How well the training procedures described perform will be evaluated in Section 5.2.1.

4.2.4 Summary of Normalisation Procedure

This section has introduced methods for finding and ranking candidate normalisations for a spelling variant, ranking is based on the current candidate list and how each candidate was found, but also previous normalisations are taken into account to improve the likelihood that previously successful candidates are scored

and ranked higher. The following steps represent a summary of those taken in the methodology described to produce a list of candidate normalisations for a given spelling variant, ranked by a confidence score:

1. Create a list of candidate normalisations from three methods:
   - Known variants list
   - Phonetic matching
   - Character edit rules
2. The predicted recall of the three methods for each candidate is calculated as follows :
   (a) For the known variants list: divide the number of times the candidate has been previously chosen as the appropriate candidate by the number of times any candidate has been offered for the current variant by the known variants list method.
   (b) For other methods: give a score of 1 if the method returns the candidate, 0 otherwise.
   (c) Penalise the character edit rules method score by 0.1 for each rule used after an initial rule edit.
   (d) If the known variants list is used as an additional step: penalise the method score by 0.2 and multiply the resulting score by the candidate's known variants list method score.
3. Produce an edit distance similarity score for each candidate and use this as the predicted recall for the edit distance method.
4. Calculate method's predicted precision (including edit distance's) for each of their candidates by dividing the candidate's method score by the sum of all of the method's suggested candidates' method scores.
5. Calculate the cumulative recall for each method (and edit distance similarity) by dividing the sum of all previous correct candidate method scores (or similarity scores) by the number of previous normalisations made.

6. Calculate the cumulative precision for each method (and edit distance similarity) by dividing the sum of all previous correct candidate method scores (or similarity scores) by the sum of all previously suggested candidate method scores (or similarity scores).

7. Combine the predicted recall scores for each method with an average weighted by the cumulative method recall scores.

8. For each method, calculate the product of the cumulative and predicted precisions and take the maximum product found as the combined precision for the candidate.

9. Merge the combined recall and precision scores into an F-Score for each candidate and use this as a confidence score to rank the candidates list.

With the ranked list produced, normalisation could be performed either automatically or manually. For automatic normalisation, the candidate with the highest confidence score could be chosen as the normalisation. This procedure can be enhanced by using a threshold which the top candidate's confidence score must reach for the normalisation to take place. This should improve precision as normalisations will only be made when there is high enough confidence of the candidate being appropriate – this shall be evaluated in Section 5.2.2. For manual normalisation, the ranked list of candidates could be presented to the user for consideration, the user could then choose the appropriate candidate from the list (if the appropriate normalisation is not present, the user should have the option to provide their own word). In the next section, a piece of software will be introduced which implements the methodology described here (and in previous sections) for detecting and both automatically and manually normalising spelling variation in single texts and entire corpora.

4.3 VARD 2

In this section a spelling normalisation tool will be described which implements the methods described in Sections 4.1 and 4.2 to provide the ability to detect and normalise spelling variants in EModE texts. The tool, named VARD 2 (VARiant Detector), builds upon the first iteration of VARD (Rayson *et al.*, 2005)

(see Section 2.3.1), adding much increased functionality and performance. The VARD 2 software was developed in Java and several iterations have been made available for use in academic research[16], with a user guide also available[17].

There are three main components of VARD 2: the interactive mode, which allows a user to manually normalise spelling variants, choosing from the list of candidates produced for each variant; an automatic (batch) mode, which allows multiple texts to be normalised automatically using the highest ranked candidates found; and finally, VARD 2 can be trained and customised in various ways to allow a user to tune the tool to their corpus. Here, these three components will be described, showing how the methods detailed in the previous sections can be used in a practical sense.

4.3.1 Interactive Processing

The interactive mode, a screenshot of which is given in Figure 4.2, allows a user to manually normalise a single text. The text is tokenised as described in Section 4.1.3, although the regular expression defining how words are detected can be customised by the user (see Section 4.3.3). Each word found is then looked up in the dictionary using the trie lookup method described in Section 4.1.2, with words found in the modern word list marked as "Not variants" and all others marked as "Variants". The list of variants can be displayed to the user and highlighted in the text. For each variant, a ranked list of candidates can be displayed, this is created using the methodology described in Section 4.2. A user can choose from this list of candidates and normalise the variant instance, or all instances of that variant in the text, to the candidate chosen. The user also has the option to provide their own word to normalise the variant to, if the required candidate is not present. For each candidate, information for how the candidate was found and ranked is also given. This includes its overall confidence score (F-Score as calculated in Section 4.2.3) and the predicted F-Score, precision and recall for each candidate finding method (see Section 4.2.2). The current cumulative F-Score, precision and recall (see Table 4.9) are also shown for each method at the bottom of the screen[18].

[16] `http://www.comp.lancs.ac.uk/~barona/vard2/availability.php`

[17] `http://www.comp.lancs.ac.uk/~barona/vard2/userguide.php`

[18] All figures given are displayed as a percentage rather than a decimal between 0 and 1 (as used in the previous sections). This simplifies the display for the user.

Once a candidate is chosen to normalise a variant, the normalisation is used as training data to adjust the future ranking of candidates (see Section 4.2.3). The normalisation is also added to the "Normalised" list which can be reviewed by the user, and normalisations reversed if necessary. Variants can also be marked as not needing normalisation by the user and subsequently added to the dictionary so future instances are not marked as variants. Furthermore, the "Not variants" list can be viewed and the user can mark words as being variants where necessary (this may include real-word variants – see Section 3.3.1). These words can then be removed from the dictionary so that future texts will have instances of the word marked as variants for normalisation. Other options available to the user include: joining two or more words separated by white space into a single word for processing (e.g. *to morow*); adjusting the F-Score recall and precision balance (β – see Equation 4.3), which will in turn affect the ranking of candidates based on whether recall or precision is the priority; and automatically normalising the entire text using the same procedure described in Section 4.3.2.

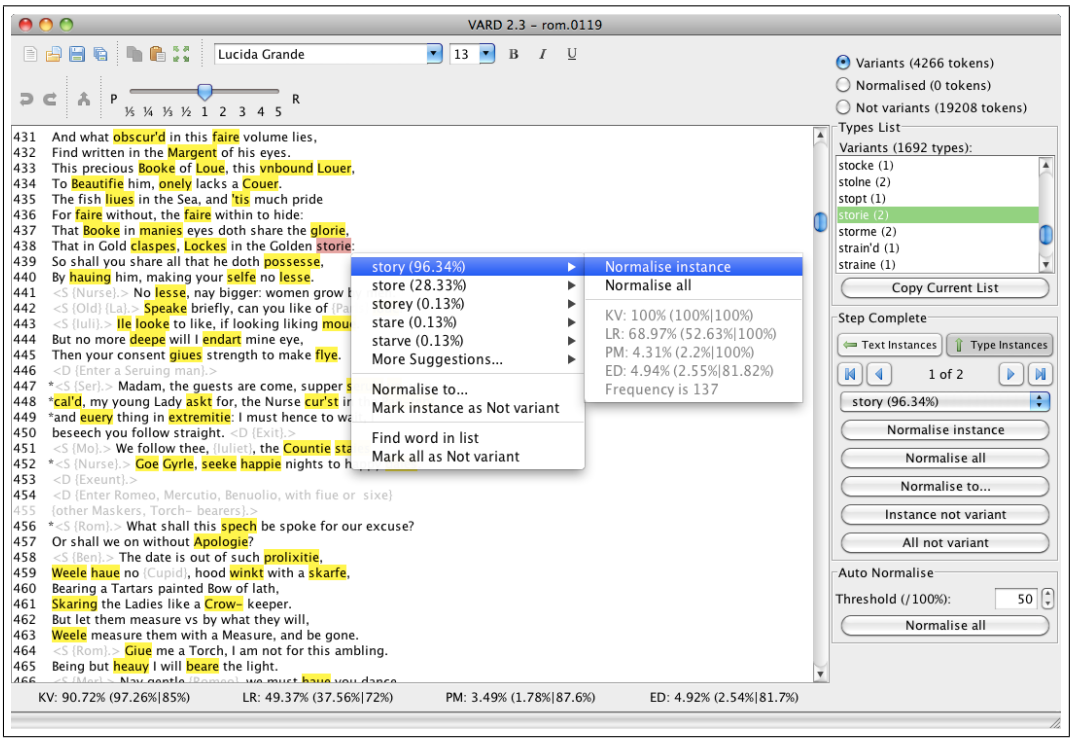


Figure 4.2: *Screenshot of interactive processing mode of VARD 2.*

The options provided by the interactive mode allow a user to manually fully process a single EModE text, a small corpus or training samples from a larger corpus. The text can then be outputted in the form of a normalised text with changes made marked with the following XML tags:

- `<normalised orig="[variant form]" auto="[true/false]"> [normalised form]</normalised>`
  Any normalisation made by the user or automatically.
- `<variant>[word]</variant>`
  Any word originally found in the dictionary but marked as variant by user.
- `<notvariant>[word]</notvariant>`
  Any word originally marked as a variant, but marked as being not a variant by the user.
- `<join orig="[old string]">[new string]</join>`
  Two or more words joined into a single word.

With the text in this form, the vast majority of corpus linguistics tools will treat the XML tags as meta-data, or can be customised to do so, and so the normalised text will be processed instead of the original text containing spelling variation. If the XML output is opened in VARD 2 again, the XML tags will be recognised and words placed in categories accordingly. VARD 2 can also be re-trained with this XML output, as described in Section 4.3.3. Also, DICER can examine the XML output to find character edit rules, this process will be described in Section 4.4.

4.3.2 Automatic Processing

Manually normalising each text in a large corpus is likely to be too time-consuming in most cases, particularly if the very large EModE datasets such as the 25,000 EEBO transcriptions (see Section 2.1.3) are considered. Therefore, an automatic normalisation mode of VARD 2 has also been developed. This mode allows a user to batch process any collection of text files, with an automatically normalised version of each file outputted (with XML tags marking normalisations as described above). Variants are detected in the same manner as in the interactive

mode and each variant word (*type*) found is looked at in turn. The same candidate normalisation lists are produced, but only the highest rank candidate is considered, with all other candidates discarded. The confidence score attached to the top candidate is compared to a user-defined threshold, if higher than the threshold then the variant is normalised with the candidate, otherwise the variant is left in its original state. This ensures that, with a high enough threshold set, normalisations are only made when the system is 'confident' of the candidate being appropriate. Setting a higher threshold will thus increase the precision of automatic normalisation, conversely setting a lower threshold will increase recall. The effect of the normalisation threshold shall be evaluated in Section 5.2.2. As in the interactive mode, the user also has the option to control the ranking of candidates by adjusting the F-Score precision and recall balance. The batch processing can be completed using the interface shown in Figure 4.3 or with a command-line version.

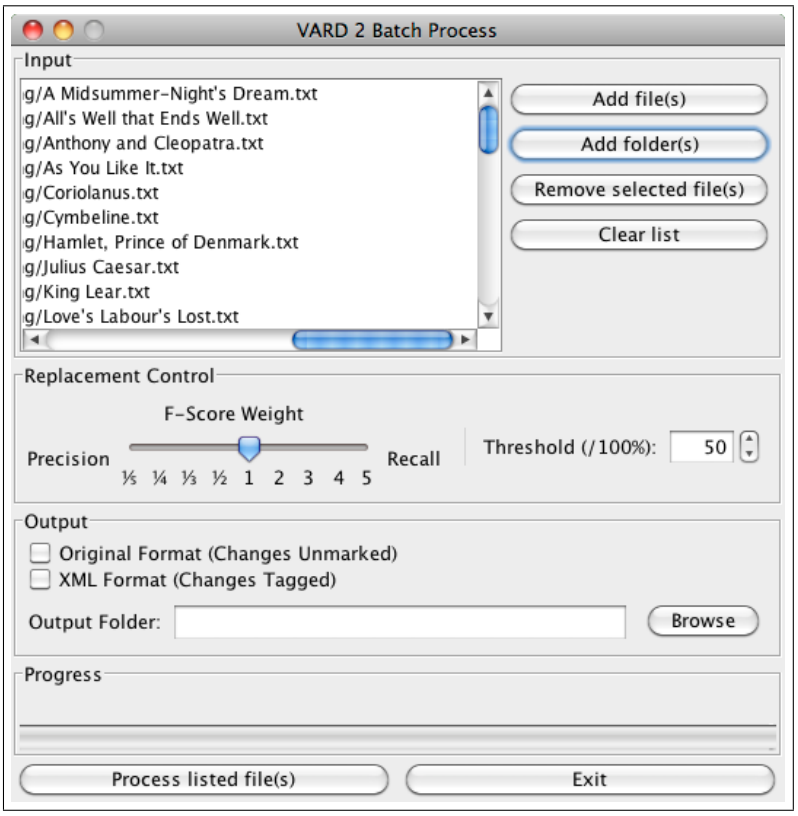


Figure 4.3: *Screenshot of batch processing mode of VARD 2.*

4.3.3 Training and Customisation

In order to make VARD 2 usable for the wide range of EModE corpora available (see Section 2.1.3) and to allow it to be able to handle the subtle differences in spelling variation characteristics between different sources (see Section 3.3), it is important that the tool is both customisable and trainable. Training can be achieved by examining normalisations made in the interactive mode, using the methodology described in Section 4.2.3 to influence the ranking of future normalisation candidates. A further method for training is available in VARD 2's training mode, a screenshot for which is shown in Figure 4.4. This allows a set of normalised text files containing VARD 2 XML tags to be used as training data. Every `<normalised>` tag is examined with the variant and normalised forms extracted and used to train VARD 2 as if selected in the interactive mode[19]. This is useful to train an instance of VARD 2 on some previously normalised training data and can also be used with texts normalised by other means, such as the Innsbruck Letters corpus utilised in Chapter 3, providing the normalisations are first converted into VARD 2 XML tags.

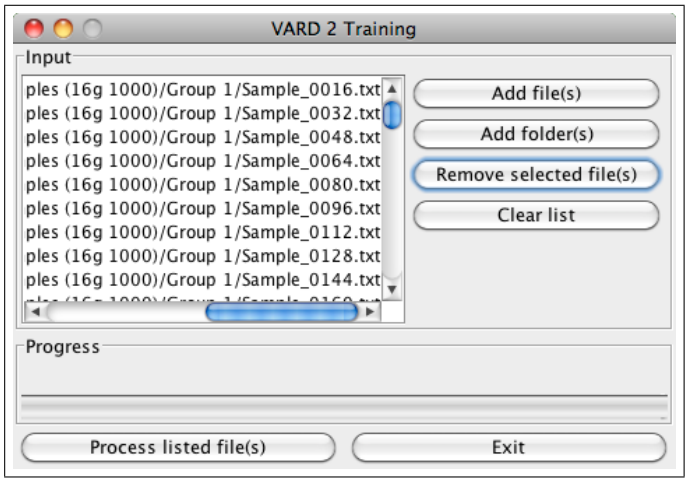


Figure 4.4: *Screenshot of training mode of VARD 2.*

There are various ways in which VARD 2 can be customised to deal with a particular corpus. For reading text files, a user can set the text encoding to be used (e.g. US-ASCII, UTF-8, etc) and indicate how meta-data is marked by

[19]Only normalisations marked as not being made automatically (through the *auto* attribute) are considered.

stating what structures VARD 2 should 'ignore', e.g. between XML tags (<. . .>). How words are detected can be set through a regular expression indicating which characters should be considered letters (see Section 4.1.3). The user can also manually change the known variants list, modern word list or rules list should external data be available; for example, rules can be taken from the DICER tool described in Section 4.4, or a user may wish to include different word groups from SCOWL (see Section 4.1.1).

With the VARD 2 tool, academic researchers are able to normalise their EModE corpora both manually and automatically to alleviate the issues spelling variation causes to corpus linguistic research; as described in Section 3.2. How well the methodology employed in VARD 2 performs will be evaluated in Chapter 5, with a case study detailing the use of the tool in the release of an EModE corpus given in Section 5.3. Our attention now turns to a supplementary development for finding and analysing character edit rules, which can be used in VARD 2.

4.4 DICER

This section shall describe DICER (Discovery and Investigation of Character Edit Rules), which was briefly introduced in Section 3.3.2 due to its use in analysing spelling variant characteristics. DICER was first developed to find spelling variation character edit rules, with the aim of improving EModE spelling normalisation. However, the tool created was also found to be valuable in the investigation of spelling characteristics, as exemplified with the analysis described in Section 3.3.2.

DICER finds character edit rules by examining variant and normalisation pairs. Such pairs can be sourced from the XML output of VARD (see Section 4.3.1), although any similar source can also be used. Given a variant and normalisation pair, DICER will produce a list of rules which can convert the variant form to the normalisation. This is achieved by utilising the matrix produced by the Levenshtein Algorithm (4.2.10), an example of which is given in Table 4.4. Algorithm 4.4.1 shows the pseudo-code for this process, which works by tracing back through the matrix to find the specific *deletion*, *insertion* and *substitution* edits that contribute to the minimum edit distance. This technique is similar to that presented by Kruskal (1983: 224-225). For each edit found, a new rule is

created using Algorithm 4.4.2. The actual occurrence details of each character edit rule are also examined and recorded. The rule's position (*start*, *second*, *middle*, *penultimate* or *end*) (determined by Algorithm 4.4.5), its index within the variant string, and the characters before and after the rule occurrence are determined by Algorithm 4.4.3. If two rules occur side-by-side then Algorithm 4.4.4 is used to merge these into a single rule. For example, if a variant *anie* has a normalisation *any*, then two rules would be created, *Substitute* $E \rightarrow Y$ and *Delete I*. The merging procedure would reduce these rules to to *Substitute* $IE \rightarrow Y$, which is much more specific and potentially more useful. Once all rule occurrences in a variant–normalisation pair are found, an additional step is taken to find any doubling or singling of letters. For example, if a rule *Delete* T occurs immediately after another T in the variant string, then the rule is changed to *Substitute* $TT \rightarrow T$. If a rule has been previously merged with another rule and part of it forms a singling or doubling rule, then the singling and doubling rule will be extracted and another rule created with the remainder of the edit. It was deemed important to distinguish doubling and singling rules because such rules are more specific and hence potentially more useful when deciding upon character edit rules for normalisation. Furthermore, the manually created rule list provided by Dawn Archer (see p. 107) contains several singling and doubling rules, indicating a likely strong presence in EModE spelling variation[20]. The rules produced for the *clapd* → *clipped* example shown in Table 4.4 are given in Table 4.11.

Rule	**Position**	**Index**	**Before**	**After**
Insert E	Penultimate	4	P	D
Sub. P → PP	Penultimate	3	A	D
Sub. A → I	Middle	2	L	P

Table 4.11: *Rule occurrences outputted from DICER when comparing* clapd *and* clipped.

[20]Tables 3.11, 3.12 and 3.13 all contain singling and doubling rules, adding further evidence to their usefulness.

Algorithm 4.4.1: FINDRULES($s1, s2, matrix$)

```
lastruleocc ← ∅
i ← s1.length
j ← s2.length
ed ← matrix[s1.length][s2.length]
while ed > 0
do { temprule ← ∅
     if i ≠ 0 and j ≠ 0 and matrix[i − 1][j − 1] = ed − 1
       then { ed ← ed − 1
              i ← i − 1
              j ← j − 1
              temprule ← CREATERULE(s1[i], s2[j])
     else if j ≠ 0 and matrix[i][j − 1] = ed − 1
       then { ed ← ed − 1
              j ← j − 1
              temprule ← CREATERULE(∅, s2[j])
     else if i ≠ 0 and matrix[i − 1][j] = ed − 1
       then { ed ← ed − 1
              i ← i − 1
              temprule ← CREATERULE(s1[i], ∅)
     else if i ≠ 0 and j ≠ 0 and matrix[i − 1][j − 1] = ed
       then { i ← i − 1
              j ← j − 1
     else if i ≠ 0 and matrix[i − 1][j] = ed
     then i ← i − 1
     else if j ≠ 0 and matrix[i][j − 1] = ed
     then j ← j − 1
     if temprule ≠ ∅
       then { ruleocc ← CREATERULEOCC(temprule, s1, s2, i)
              if lastruleocc ≠ ∅
                then ruleocc ← MERGERULEOCCS(ruleocc, lastruleocc)
              lastruleocc ← ruleocc
       else { ruleoccslist ← ruleoccslist + lastruleocc
              lastruleocc ← ∅
ruleoccslist ← ruleoccslist + lastruleocc
return (ruleoccslist)
```

Algorithm 4.4.2: CREATERULE($search, replacement$)

```
rule.search ← search
rule.replacement ← replacement
if search = ∅
  then rule.type ← "insertion"
  else if merged.rule.replacement = ∅
  then rule.type ← "deletion"
  else rule.type ← "substitution"
return (rule)
```

Algorithm 4.4.3: CREATERULEOCC($rule, s1, s2, index$)

```
ruleocc.cbindex ← index − 1
ruleocc.caindex ← index + 1
if index > 0
  then ruleocc.charbefore ← s1[ruleocc.cbindex]
if index < s1.length − 1
  then ruleocc.charafter ← s1[ruleocc.caindex]
if rule.type = "insertion" and index < s1.length
  then { ruleocc.caindex ← index
       { ruleocc.charafter ← s1[index]
ruleocc.pos ← GETPOSITION(ruleocc.cbindex, ruleocc.caindex, s1.length)
ruleocc.variant ← s1
ruleocc.normalisation ← s2
ruleocc.rule ← rule
ruleocc.index ← index
return (ruleocc)
```

Algorithm 4.4.4: MERGERULEOCCS($current, last$)

```
if current.pos = "end"
   or current.index = last.index
   or current.index = lastocc.index − 1
then { merged ← current
       merged.charafter ← last.charafter
       merged.caindex ← last.caindex
       merged.pos ← GETPOSITION(merged.cbindex, merged.caindex,
          merged.variant.length)
       search ← current.rule.search + last.rule.search
       replacement ← current.rule.replacement + last.rule.replacement
       merged.rule ← CREATERULE(search, replacement)
       return (merged)
else return (current)
```

Algorithm 4.4.5: GETPOSITION($cbindex, caindex, length$)

```
if cbindex < 0
  then return ("start")

  else if caindex ≥ length
  then return ("end")

  else if cbindex = 0 and caindex = length − 1
  then return ("middle")

  else if cbindex = 0
  then return ("second")

  else if caindex = length − 1
  then return ("penultimate")

  else return ("middle")
```

Using the methodology described, a series of variant and normalisation pairs can be analysed in turn to produce a list of rules and rule occurrences. The lists created are inserted into a MySQL database, which can be accessed and

analysed through a series of PHP webpages[21]. DICER's main summary tables for the Innsbruck Letters corpus analysis (as used in Section 3.3.2) are shown in Figure 4.5. The analysis can be used to examine the most common rules present, where rules occur and other trends such as edit distance and rule types. A user can also select a specific character edit rule and look at which characters occur before and after the rule. This is shown in Figure 4.6 for the rule *Substitute* $W \rightarrow U$ (in the Innsbruck analysis). Any frequency given in the DICER tables can be selected to view a list of variant–normalisation pairs producing that frequency. For example, a list of all variant normalisations using the rule *Substitute* $T \rightarrow TT$ at the *end* of the variant could be produced.

Using the analysis produced by DICER, a user can devise a list of character edit rules which represent the spelling variation found in the corpus analysed. This list can then be used to improve the performance of VARD 2 in normalisation. The user can choose how many of the most common rules to use and may also wish to make rules more specific by examining where in a variant rule occurrences most commonly occur (e.g. *Delete E* only at *end* of word). Further context for a rule can also be taken from surrounding characters if a rule is commonly found before or after a certain character (e.g. *Substitute OW* $\rightarrow$ *OU*).

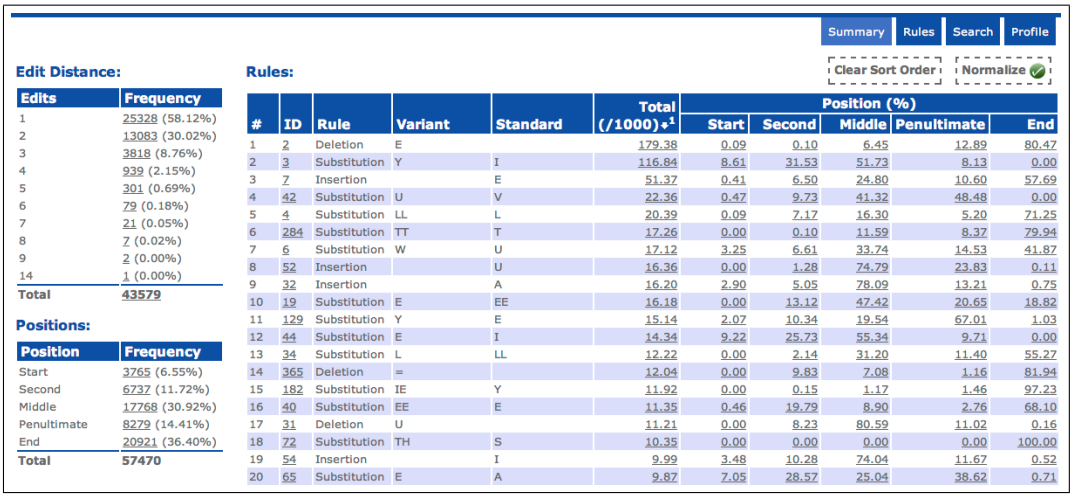


Figure 4.5: *Screenshot of DICER summary of the Innsbruck analysis.*

[21] `http://corpora.lancs.ac.uk/dicer/`

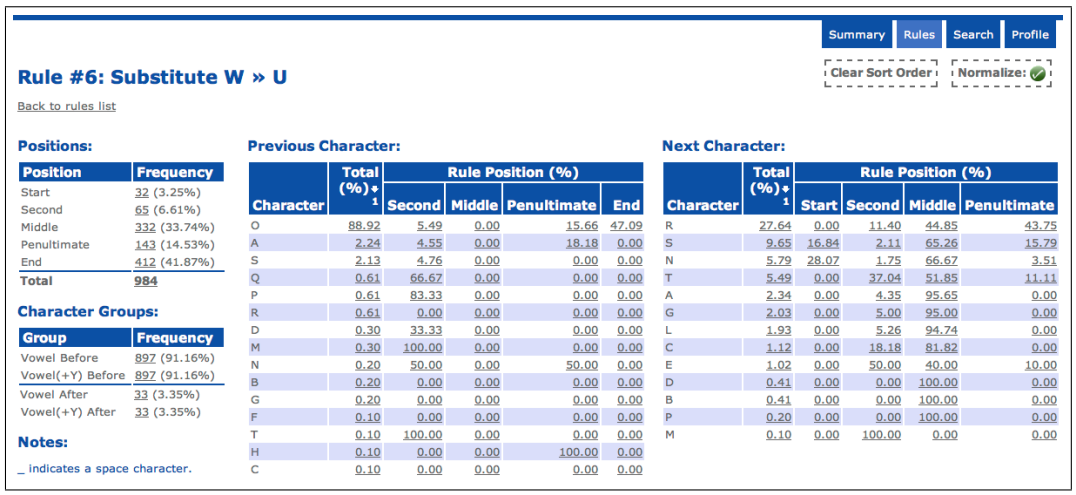


Figure 4.6: *Screenshot of DICER analysis for a specific rule in the Innsbruck analysis.*

4.5 Chapter Summary

Building on the analysis of EModE spelling characteristics presented in Chapter 3, this chapter has described specific methodologies which can be used in the normalisation of EModE spelling variation. How variants should be detected has been considered, with an efficient technique detailed for the key task of dictionary lookup. Several steps have been discussed which result in a ranked list of candidate normalisations being produced for a given variant. Previous normalisations can also be taken into account in the ranking of candidates – this is achieved through a training technique which is informed by the standard precision and recall metrics.

The methodologies described have been developed into a normalisation tool, VARD 2, which can be used to manually and automatically normalise spelling variation in EModE texts. The tool is highly customisable and trainable, resulting in the ability to tune the normalisation process for a specific corpus and the individual properties of its spelling variation. Another tool, DICER, can assist in this process by suggesting character edit rules based on previous variant and normalisation pairs.

The normalisation procedures resulting from the development presented in this chapter allow for delicate control of recall and precision when choosing variant normalisations. For example, a user may set a high normalisation threshold during automatic processing to ensure that normalisations are only made when the system

is highly confident of its top ranked candidate normalisation – this should result in high precision. In the next chapter, the normalisation procedures presented here are evaluated in these same terms: precision and recall.

Chapter 5

Evaluation of Spelling Variant Normalisation

The final research question established in Section 1.2, *RQ 3*, will be addressed in this chapter through the evaluation of the normalisation solution detailed in Chapter 4. Whilst the manual (interactive) normalisation procedure detailed in Section 4.3.1 is of use for single texts and small corpora, the primary concern is how well normalisation can be performed automatically. A well-performing automatic normalisation tool would allow for the large Early Modern English corpora now available (see Section 2.1.3) to have a significant amount of the spelling variation present within them normalised. This, in turn, would lead to more accurate corpus linguistic analysis; for example, Table 3.3 in Section 3.2.1 shows that even with partial normalisation, part-of-speech tagging accuracy is increased. Hence, the automatic normalisation procedure developed (see Section 4.3.2) shall be the main focus of the evaluation given here.

To measure performance, one can establish how close automatic normalisation is to manual normalisation of the same texts. A handful of manually normalised texts have already been utilised for the analysis presented in Chapter 3 and shall be re-used for the evaluations presented in this chapter. Both recall and precision shall be used to measure this closeness throughout. High recall is important as it measures the proportion of spelling variation that can be dealt with by automatic means. However, as mentioned previously in this thesis, high precision is likely to be more important when normalising EModE spelling variation. If it is found that only low precision is achieved, numerous real-word errors would be introduced

to the texts being normalised. This could result in the normalisation procedure actually having a detrimental effect on the accuracy of corpus linguistic methods. By measuring both recall and precision, it will be possible to observe how the two factors are balanced and also how different setups can be used to prioritise one over the other.

As discussed throughout this thesis, the first stage of any normalisation (or spellchecking) procedure is to first find in a text the words that require normalisation, i.e. spelling variants. We begin in Section 5.1 by evaluating the dictionary lookup method used in the developed normalisation tool, quantifying its limitations in terms of spelling variant detection. In Section 5.2 the central evaluation of the final automatic normalisation procedure will be presented. This will include examining the effectiveness of the training procedure and investigating how recall and precision can be controlled with the normalisation threshold. The use of the VARD 2 software (which utilises the normalisation methods developed – see Section 4.3) with a newly released Early Modern English corpus will be detailed in a case study in Section 5.3. The chapter's findings are then summarised in Section 5.4.

5.1 Spelling Variant Detection

The performance of one aspect of spelling variant detection has already been partially evaluated in Section 3.3.1's analysis of real-word spelling variants in three EModE corpora; the same dictionary for variant detection was used in this analysis as is used in VARD 2 (as described in Section 4.1.1). Another issue in variant detection, as indicated in Section 3.1, is words which do not require normalisation being marked erroneously as variants due to not being present in the system's dictionary. In this section, the effect of both real-word spelling variants and erroneously marked variants on the recall and precision of the developed solution's spelling variant detection process will be assessed. The effect of dictionary size will also be examined, looking at how both increasing and decreasing the number of words present impacts on variant detection.

As with the real-word spelling variant analysis in Section 3.3.1, manually normalised EModE texts can be used to establish how well the dictionary lookup spelling variant detection method is performing. The same three sources can

be utilised: the full Innsbruck Letters corpus, the Shakespeare sample and the Lampeter corpus sample. The Shakespeare and Lampeter samples are relatively small, but to an extent represent the corpora they are from. The Shakespeare samples are from five comedies evenly spread through Shakespeare's writing career, whilst the Lampeter samples are from three different domains in the corpus. Each sample contains 1,000 words of running text beginning at a randomly selected index. The normalisations present in the three sources can be examined and the list of variants normalised considered as the total number of variants detectable. By comparing the variants detected by the dictionary lookup method to the list of variants that should be detected, the three rates needed to calculate recall and precision can be retrieved: True Positives (TP), False Positives (FP) and False Negatives (FN). TP is the total number of detectable variants detected by our solution, FN is the number of detectable variants not detected (i.e. real-word variants), and FP is how many extra words have been detected as variants (i.e. erroneously marked variants). The TP, FN and FP rates for both variant types and tokens for the three datasets are shown in Table 5.1.

	Tokens			**Types**		
	TP	**FN**	**FP**	**TP**	**FN**	**FP**
Shakespeare	860	99	85	525	48	53
Lampeter	237	25	106	154	7	93
Innsbruck	38,674	5,066	3,208	12,715	805	1,984

Table 5.1: *Token and type true positive, false negative and false positive rates for spelling variant detection.*

Recall and precision scores can be calculated from the figures given in Table 5.1 using the same equations used previously in this thesis[1]. The recall and precision scores for each dataset are given in Table 5.2, again for both types and tokens. The results, on the whole, are good with the vast majority of variants detected and, except for the Lampeter dataset, only a small number of extra variants detected erroneously. The difference in precision scores for the Lampeter dataset can be explained by examining the normalised samples and observing that several Latin passages are present. The majority of the Latin words will be detected as variants

[1]For clarity: $recall = \frac{TP}{TP+FN}$ and $precision = \frac{TP}{TP+FP}$.

as they will not be present in the modern English word list. This high number of false positives, together with the low number of variants present, accounts for the lower precision scores reported.

	Tokens		Types	
	Recall	**Precision**	**Recall**	**Precision**
Shakespeare	0.897	0.910	0.916	0.908
Lampeter	0.905	0.691	0.957	0.623
Innsbruck	0.884	0.923	0.940	0.865

Table 5.2: *Token and type recall and precision for spelling variant detection.*

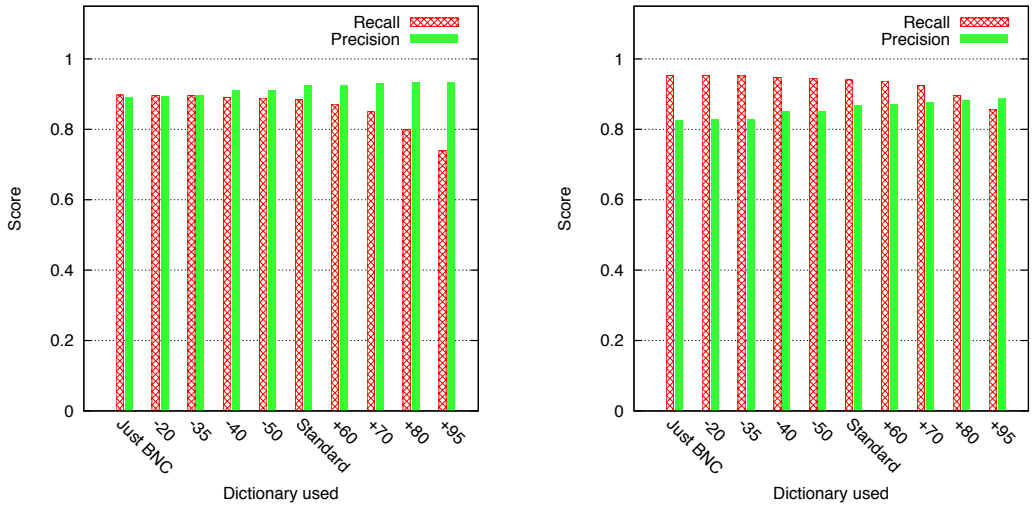
One factor which is likely to have an impact on spelling variant detection is the size of the dictionary used for looking up words. We can investigate this effect by repeating the recall and precision calculations (shown in Table 5.2) with different sized dictionaries. The dictionary collated for the developed normalisation solution contains 82,573 words. As discussed in Section 4.1.1, the majority of these words come from the SCOWL groups of words (the remainder are from the most frequent and distributed words in the BNC). The size of the dictionary can be reduced by omitting some of these word groups, and its size can be increased by adding more word groups available from SCOWL. Ten different dictionaries were created for the evaluation. These ranged from a dictionary containing no SCOWL groups at all (i.e. just containing the BNC word list) at just over 25,000 words, to a dictionary at just over half a millions words containing groups of the 95% most frequent English words. The full list of dictionaries used for the evaluation is given in Table 5.3. It should be highlighted that not only do the larger dictionaries contain many more words, they also contain increasingly infrequently used words. As discussed in Section 4.1.1, the frequency of words is important to consider as high frequency words are likely to appear in texts being normalised, whereas low frequency words are less likely to appear but may happen to have the same spelling as a variant of another word, hence causing a real-word spelling variant.

	SCOWL groups included	**Words**
Just BNC	None	26,097
-20	As -35, with the removal of *british-words.20* and *english-words.20*.	27,224
-35	As -40, with the removal of *british-words.35*, *english-contractions.35*, *english-upper.35*, *english-words.35* and *special-roman-numerals.35*.	29,901
-40	As -50, with the removal of *british-words.40*, *english-contractions.40*, *english-upper.40* and *english-words.40*.	53,605
-50	As Standard, with the removal of *british-words.50*, *british-words.55* and *english-words.50*.	59,440
Standard	As described in Section 4.1.1.	82,573
+60	As Standard, with the addition of *british-words.60*, *english-contractions.60*, *english-upper.60* and *english-words.60*.	96,257
+70	As +60, with the addition of *british-words.70*, *english-contractions.70*, *english-upper.70* and *english-words.70*.	136,581
+80	As +70, with the addition of *british-words.80*, *english-contractions.80*, *english-upper.80* and *english-words.80*.	282,054
+95	As +80, with the addition of *british-words.95*, *english-contractions.95*, *english-upper.95* and *english-words.95*.	502,606

Table 5.3: *Test dictionaries used for evaluating the effect of dictionary size on spelling variant detection.*

Using each dictionary, the recall and precision scores for spelling variant detection in the Innsbruck Letters corpus were calculated just as before. The results are shown in Figure 5.1. As can be seen, increasing the size of the dictionary increases the precision of detection; i.e. less extraneous variants are detected. At the same time recall falls, with a larger number of real-word spelling variants. Decreasing the dictionary size has the opposite effect with precision falling and

recall rising. The dictionary used in the developed solution (as described in Section 4.1.1), shown in Figure 5.1 as *Standard*, looks to have achieved a good balance between precision and recall. Using a larger dictionary would give only a small precision improvement (+0.009 for tokens and +0.022 for types by *+95*), but a relatively large decrease in recall would also occur (−0.145 for tokens and −0.081 for types by *+95*). Conversely, a smaller dictionary would give small recall gains (+0.013 for tokens and +0.012 for types by *Just BNC*), but a larger drop in precision (−0.034 for tokens and −0.040 for types by *Just BNC*). This view is backed up by calculating an F-Score for each dictionary evaluation. For tokens, the F_1-Score for *Standard* is highest at 0.904 (next best 0.900 for *-40*). For types, the F_1-Score for *Standard* is second highest at 0.9018 (just behind *+60* at 0.9020).



(a) Tokens (b) Types

Figure 5.1: *The effect of dictionary selection on spelling variant detection recall and precision in the Innsbruck Letters corpus.*

The recall scores, rather than the precision scores, will perhaps have more of an impact on the performance of automatic normalisation as a whole; the recall rates given represent the upper boundary to automatic normalisation coverage – a normalisation cannot be made if the variant is not first detected. The precision scores will only have an impact when the extraneous variants detected are subsequently normalised. If, in the majority of cases, suitable candidate normalisations cannot be found for these variants, then little impact will be made

on the precision of automatic normalisation – the extra variants will be left in the original form. The impact of variant detection on automatic normalisation will be examined in more detail in the following section.

5.2 Automatic Normalisation

Evaluating the performance of automatic normalisation is key to establishing how useful the developed normalisation tool would be for processing large Early Modern English corpora, where manual normalisation would be too time-consuming. This section will quantify how similar automatically normalised texts are to the same texts manually normalised. Particular attention will be given to how much of an improvement on performance can be achieved with different amounts of training (Section 5.2.1) and what effect the normalisation threshold has on the balance between recall and precision (Section 5.2.2).

The previously used Innsbruck Letters corpus (see Section 3.3.1) was utilised as test data for all of the experiments presented in this section. The corpus has been automatically normalised and then manually checked[2]. Evaluation will be based on the assumption that the normalisations present in the texts are accurate. It is likely that human error will have led to mistakes occurring, with missed variants and inappropriate normalisations made[3]. However, performance comparable to manual normalisation is an acceptable goal for automatic normalisation.

Testing was carried out with the corpus by first reseting it to its original form with no normalisations made (i.e. the manual normalisations were replaced with the original variant forms). Automatic normalisation was then performed, as described in Section 4.3.2 – the exact setup for automatic normalisation shall be described for each experiment. The automatically normalised texts can then be compared to the manually normalised texts and any differences noted. We evaluate the automatic normalisation performance in terms of recall and precision percentages, with the following definitions. Recall can be defined as the proportion of required (manual) normalisations that are made

[2]This is essentially equivalent to manual normalisation, to which it shall be referred to henceforth.

[3]Although, every effort was made to check as much of the normalisations as possible, with extensive time spent 'cleaning up' the texts during the combination of the parallel original and normalised lines.

correctly by automatic normalisation. As discussed in Section 5.1, the required normalisations will include normalisations of variants which are not detected by our system as being variants, i.e. real-word variants. These will count against the correct automatic normalisations as they cannot be normalised without first being detected; therefore, the number of real-word variants will impact on recall. Precision can be defined as the proportion of automatic normalisations made which match exactly to the corresponding required (manual) normalisation, i.e. are correct. Also discussed in Section 5.1 is the problem of extra variants being detected erroneously, i.e. words not in the dictionary that are valid (this may include, for example, proper names or foreign language words). If subsequently these extraneous variants are automatically normalised, these are obviously incorrect normalisations and this will impact on precision.

5.2.1 Training

It has been shown previously, particularly in Section 3.3.2, that different EModE corpora will contain different spelling variant properties. It has also been hypothesised that these differences will be even more apparent when there are differences between the genre, text type or time period of corpora. It is, therefore, unlikely that a generic EModE spelling variant normalisation tool could be built. Instead, any solution should be adaptable to the corpus which is having its spelling normalised. A key component in the adaptability of the normalisation solution described in Chapter 4 is how well the training procedure improves performance. This shall be evaluated here to establish if the normalisation procedure can be tuned to the Innsbruck Letters corpus's spelling variation.

To observe the effect of training on the performance of automatic normalisation, evaluation needs to be completed after the system has been trained on different amounts of data. To this end, the Innsbruck Letters corpus was split into samples of 1,000 words. The samples were created by first splitting the entire corpus into small segments (maximum 50 words). Each sample was then built by appending randomly selected segments until the length of the sample reached the target of 1,000 words. Samples were created until the segments remaining were insufficient to build another sample. This resulted in 179 samples being created, each containing 1,000 tokens of running text, including any normalisations made

marked up with XML tags[4] (as previously described). The samples were then split into two halves, one for training and one for testing[5]. Evaluation can then be completed by observing how well automatic normalisation performs on the test set after training the system on an increasing number of 1,000 word samples. The properties of the test set are given in Table 5.4, with the amounts of real-word spelling variants (FN) and erroneously detected variants (FP) also given. These are calculated in the same manner as described for Table 5.1.

	Tokens		**Types**	
	Freq.	**% of total**	**Freq.**	**% of total**
Words	87,996		12,794	
Normalised variants	20,943	23.80%	7,796	60.93%
Real-word spelling variants	2,420	2.75%	520	4.06%
Erroneously detected variants	1,503	1.71%	1,062	8.30%

Table 5.4: *Properties of the Innsbruck Letters corpus test set used for normalisation evaluation.*

A normalisation threshold was arbitrarily set at 50%[6], and the test set automatically normalised with no training at all (i.e. the normalisation tool was in its default state). Recall and precision were recorded as detailed above (p. 148) for both types and tokens. The first sample was then used as training data as described in Section 4.3.3. Then, the automatic normalisation test was performed again and recall and precision recorded as before. This procedure was repeated until no training data remained. The observed precision and recall scores after subsequent amounts of training are shown for tokens in Figure 5.2 and for types in Figure 5.3. For the first 11,000 training tokens, the samples were further split into 200 word segments to show increased detail in the early stages of training.

[4]XML tags were not included in the 1,000 word count.

[5]The actual split was 91 training samples and 88 test samples due to the samples originally being split into 16 groups (and then 8 groups used for each) to allow for evaluation with smaller amounts of training and test data. The effect on the final results compared to using a 89/90 split should be negligible.

[6]The effect of using different normalisation thresholds will be evaluated in Section 5.2.2

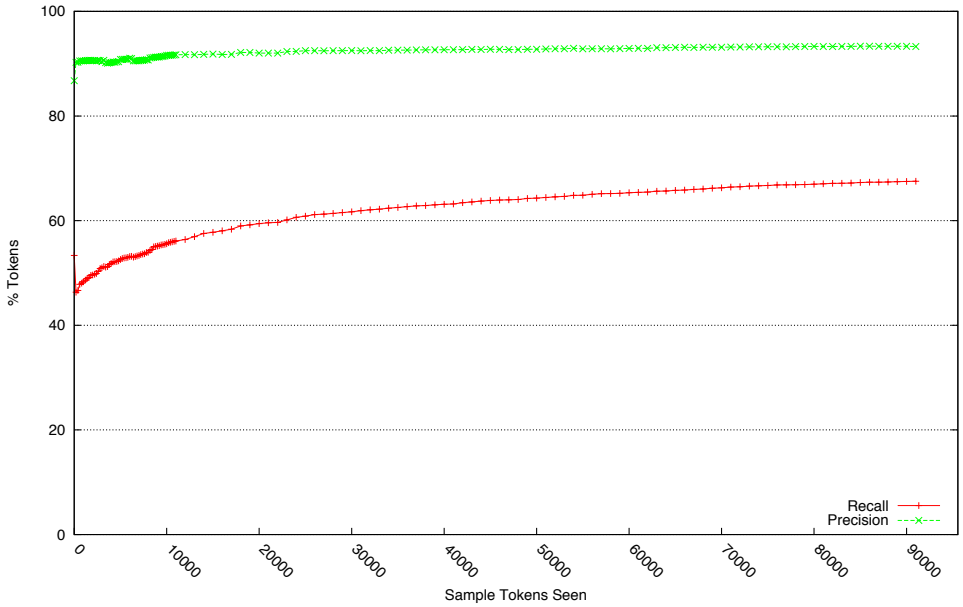


Figure 5.2: *Precision and recall of automatic normalisation on the Innsbruck Letters corpus with increasing training levels (tokens).*

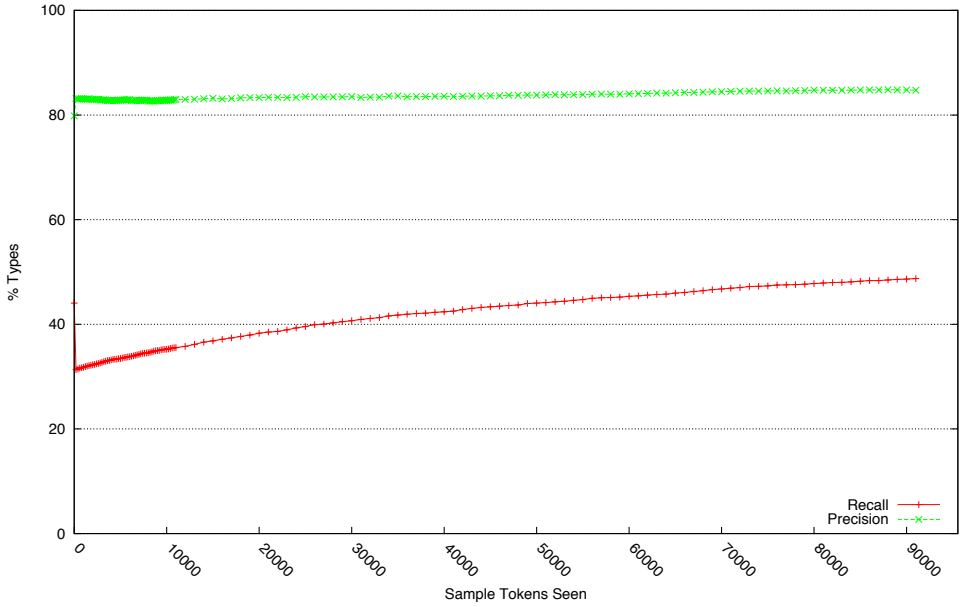


Figure 5.3: *Precision and recall of automatic normalisation on the Innsbruck Letters corpus with increasing training levels (types).*

With no training at all, recall begins at 53.34% for tokens (44.05% for types) and precision at 86.75% (79.84% for types). Then, with just one 200 word training sample, recall falls to 46.33% for tokens (31.32% for types) but precision increases to 90.26% (83.14% for types). This can be explained by the 'naïve' cumulative precision and recall initially attached to each method (0.5 for each). Even in the small amount of training data seen in the first sample, more appropriate cumulative recall and precision scores will be attached to each method. Thus, many normalisation candidates from low precision methods which previously scored above the normalisation threshold (50%) will now score lower. This will mean less normalisations are made (reducing recall) but those made are more likely to be correct (increasing precision). After the initial sample, recall steadily rises for both tokens and types, whilst precision increases very slightly. After all 91,000 tokens of training, recall increases to 67.54% for tokens (48.74% for types) and precision increases to 93.27% (84.74% for types). Half of the improvement in performance (in terms of tokens recall) is achieved by 13,000 words of training (56.94% recall, 91.73% precision). Three-quarters of the improvement is achieved by 34,000 words of training (62.39% recall, 92.58% precision). This shows that a relatively small amount of effort can be completed to gain the bulk of the benefit from training. After this, decreasing performance gains are observed. Overall, the results are fairly promising – particularly in the case of tokens. With quite a small amount of training, over 60% of required normalisations can be made correctly, whilst the number of incorrect normalisations made remains below 10%[7]. For types, results are still fairly respectable with nearly 50% of required normalisations made correctly after full training, and precision remaining over 80%. However, the types results serve to highlight the difficulty of the normalisation task in hand with many variants being unique (or very rare) in the corpus and thus being difficult to normalise and train for.

As discussed in Section 5.1, the detection of variants will impact on normalisation performance due to variants not being detected being impossible to normalise automatically (restricting recall) and variants being detected erroneously potentially being subsequently normalised (harming precision). We can observe how much of an effect detection performance has on the performance of automatic

[7]A higher normalisation threshold should increase precision, this will be evaluated in Section 5.2.2

normalisation by re-producing the analysis above but assuming a perfect detection method whereby all variants present are detected and no extra variants are erroneously detected. The results for this analysis are shown in Figure 5.4 for tokens and Figure 5.5 for types.

With this hypothetical variant detection method, results are somewhat improved. After full training, recall now stands at 76.94% for tokens (51.54% for types) and precision is 94.36% (88.39% for types). Interestingly, the levels of the increases in recall indicate that if real-word variant errors could be found, then the normalisation procedure could successfully normalise them. Furthermore, the small increases in precision indicate that some erroneously detected variants could be being automatically normalised – this shall be explored in more detail in the next section.

The results presented in this section show the value of the training procedure for automatic normalisation. Increases in performance of recall are observed whilst at the same time a relatively high precision is maintained (and even increased). We take the results of this training into the next section and investigate how recall and precision can be balanced by altering the normalisation threshold.

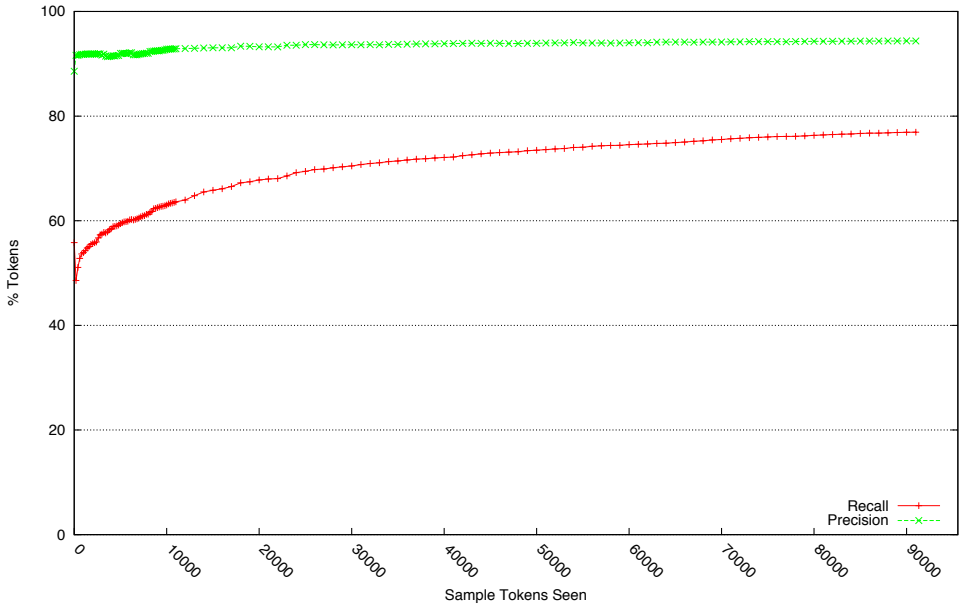


Figure 5.4: *Precision and recall of automatic normalisation with a hypothetical perfect variant detection method on the Innsbruck Letters corpus with increasing training levels (tokens).*

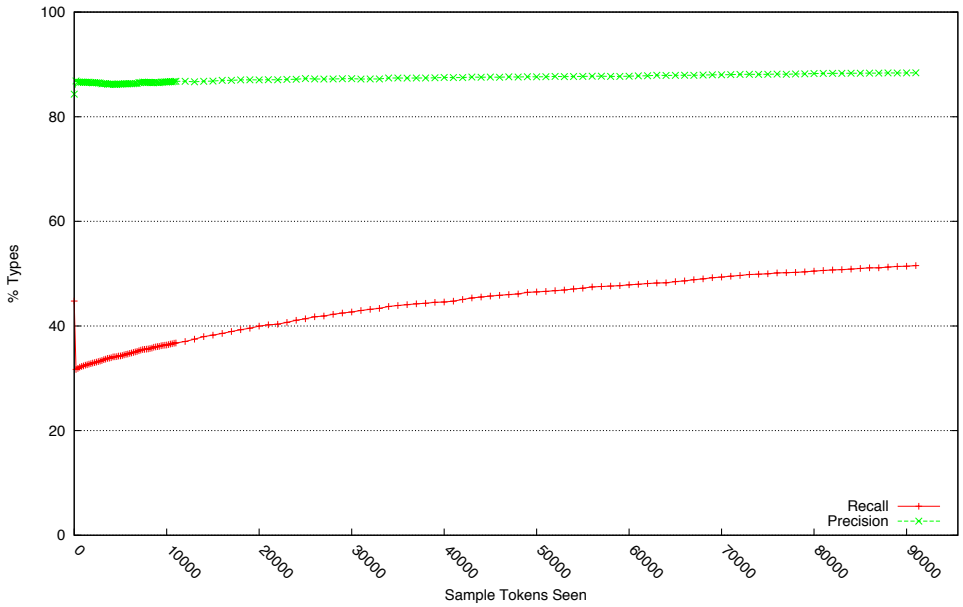


Figure 5.5: *Precision and recall of automatic normalisation with a hypothetical perfect variant detection method on the Innsbruck Letters corpus with increasing training levels (types).*

5.2.2 Normalisation Threshold

One of the key features of the developed normalisation procedure is the confidence scores attached to each candidate normalisation. This is used to rank the list of suggested candidates so that the most likely candidate is offered for automatic normalisation. Additionally, a threshold can be set for automatic normalisation whereby a normalisation is only made when the top ranked candidate has a confidence score over the provided threshold. In this section we shall evaluate the effect of this threshold in terms of automatic normalisation recall and precision. We use the same test set as used in the previous section (see Table 5.4) to calculate the recall and precision of automatic normalisation (as described on p. 148) at different normalisation thresholds. The automatic normalisation evaluation is performed after being trained on the full set of Innsbruck training samples used in the previous section. The results of this analysis are shown for tokens in Figure 5.6 and for types in Figure 5.7.

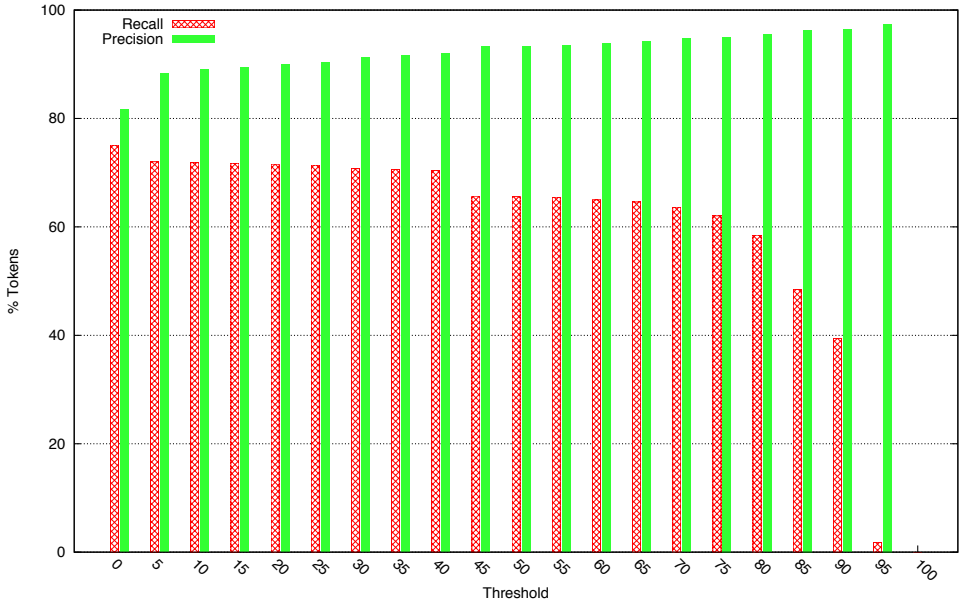


Figure 5.6: *Precision and recall of automatic normalisation on the Innsbruck Letters corpus with different normalisation thresholds (tokens).*

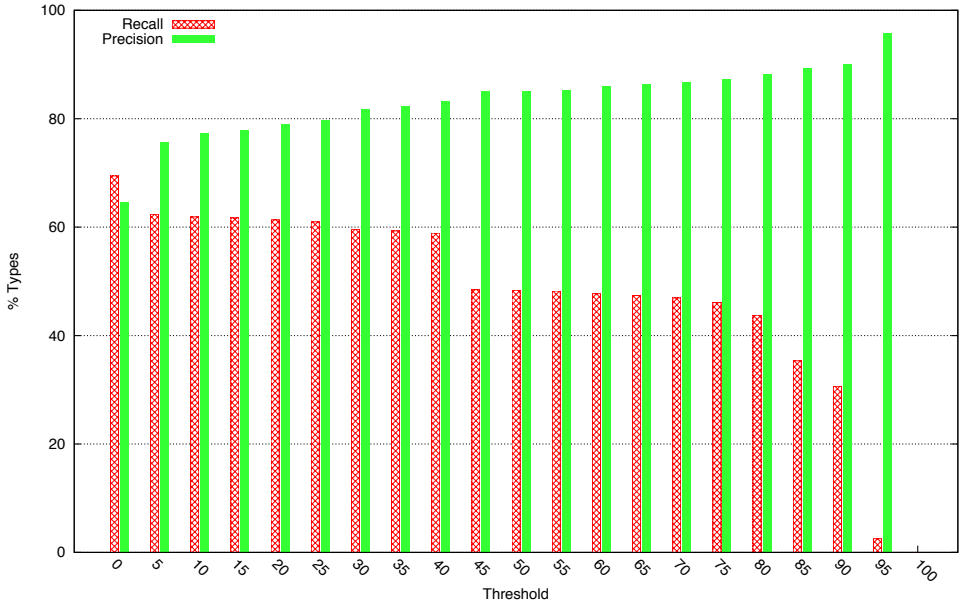


Figure 5.7: *Precision and recall of automatic normalisation on the Innsbruck Letters corpus with different normalisation thresholds (types).*

The results show the control one can have over the balance between precision and recall by altering the normalisation threshold. The maximum precision available, with a 95% threshold, is 97.39% for tokens (95.77% for types), but is of little use due to recall falling dramatically to only 1.78% (2.50% for types). Using a threshold of 75% yields a recall score of 62.02% for tokens (46.73% for types) and 95.02% (87.35% types) for precision. These figures could certainly be considered respectable performance for the difficult task of automatic normalisation. If for some reason recall was the priority, setting the threshold to 0% (effectively always allowing normalisation if at least one candidate is found), gives a recall score of 75.05% for tokens (69.43% for types) with precision inevitably falling, but still at 81.62% for tokens (64.62% types).

As with the training procedure evaluation, we can investigate the effect of the inaccuracy of spelling variant detection by producing the same results as above but with a hypothetical perfect spelling variant detection method. This will highlight the performance of just automatic normalisation. The results for this analysis are given in Figure 5.8 for tokens and Figure 5.9 for types.

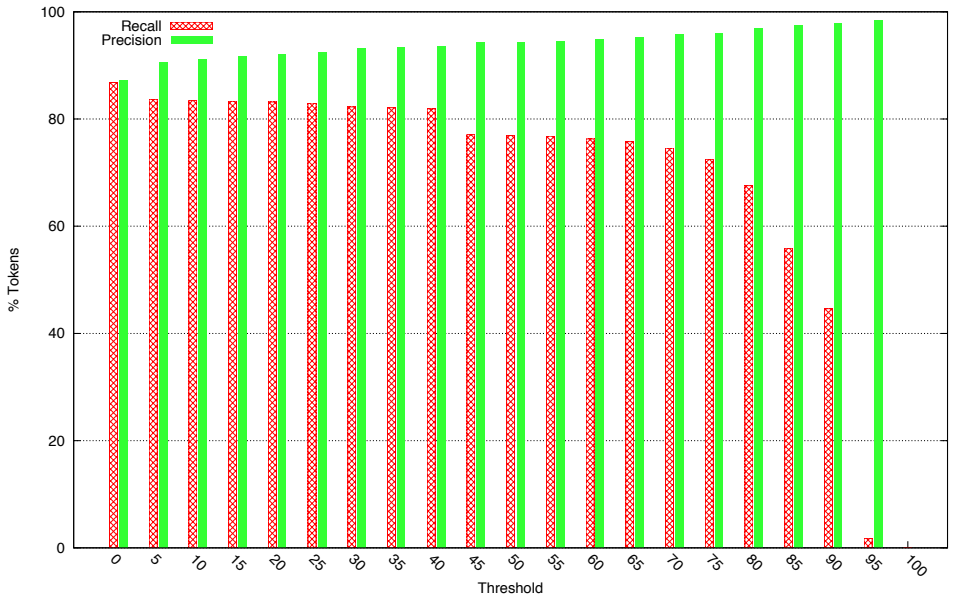


Figure 5.8: *Precision and recall of automatic normalisation with a hypothetical perfect variant detection method on the Innsbruck Letters corpus with different normalisation thresholds (tokens).*

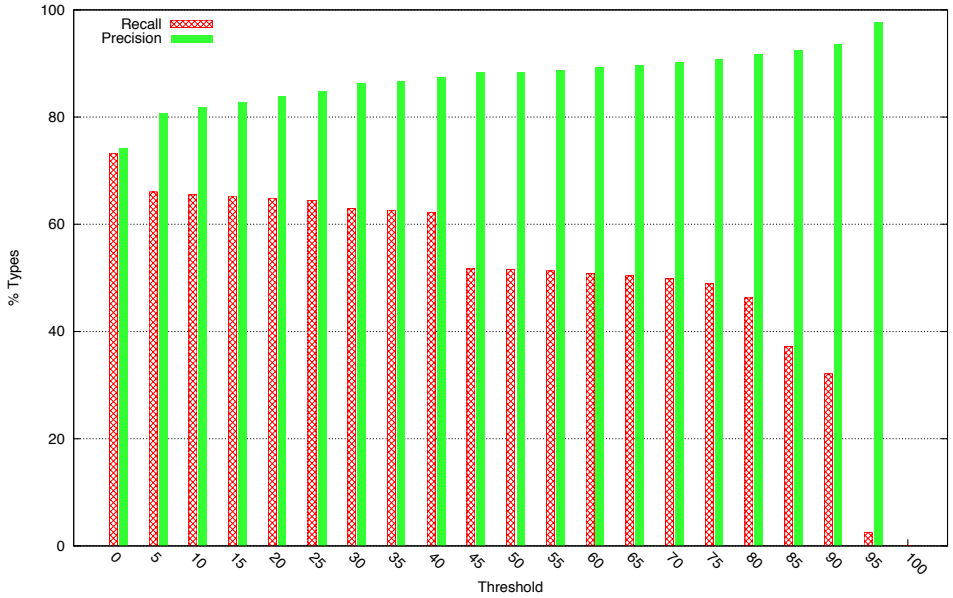


Figure 5.9: *Precision and recall of automatic normalisation with a hypothetical perfect variant detection method on the Innsbruck Letters corpus with different normalisation thresholds (types).*

The results here, albeit hypothetical, show considerable increases in recall (as in the corresponding training procedure experiment) with the recall at a 75% threshold rising to 72.47% for tokens (48.88% for types). Precision increases marginally also to 96.01% for tokens (90.68% for types). Furthermore, if recall is the priority, a 0% threshold now yields a recall score of 86.80% for tokens (73.26% for types) and a precision score of 87.22% (74.16% types). These results again suggest that if real-word spelling variants could be detected, the automatic normalisation procedure would normally be successful.

A potentially more serious issue is indicated by the marginal increases in precision if detection errors are ignored. As discussed in Section 5.1, erroneously detected variants are only problematic if they are subsequently normalised as any normalisation made will clearly be an error. The increases in precision shown in Figures 5.8 and 5.9 may indicate that many of the erroneously detected variants are being normalised, and thus essentially introducing real-word errors into the text. The scale of the problem is assessed in Figure 5.10 (tokens) and Figure 5.11 (types). Here the proportion of erroneously detected variants which are subsequently normalised is calculated at different normalisation

thresholds. Unsurprisingly, a lower threshold results in more of these extra variants being normalised (incorrectly), however for higher thresholds, relatively few are normalised. For the 75% threshold highlighted above, only 13.98% of the extra variant tokens detected are normalised (13.04% for types). It is also important to put these percentages into context. From Table 5.4, the actual number of erroneously detected variants is only 1,503 tokens (1.71% of all tokens) and 1,062 types (8.30% of all types). At a 75% threshold, the proportion of these variants normalised means that 210 real-word error tokens will be introduced, 0.24% of all tokens. The corresponding type figures are 138 real-word error types introduced, 1.08% of all types. Whilst the seriousness of introducing real-word errors during normalisation should not be underestimated, the low levels observed here should not pose considerable problems for subsequent corpus linguistic analysis[8].

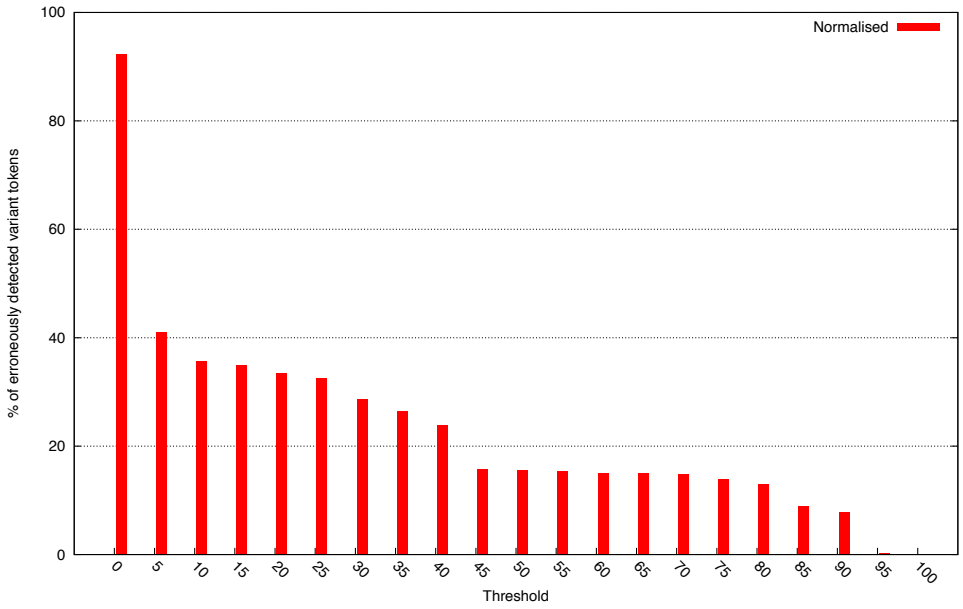


Figure 5.10: *Proportion of erroneously detected variants (tokens) in the Innsbruck Letters corpus which are subsequently automatically normalised at different normalisation thresholds.*

[8]It is also worth noting that observation of the actual normalisations made revealed that a large number could be considered to be appropriate, that is they may have been missed as being variants in the original manually checked normalisation.

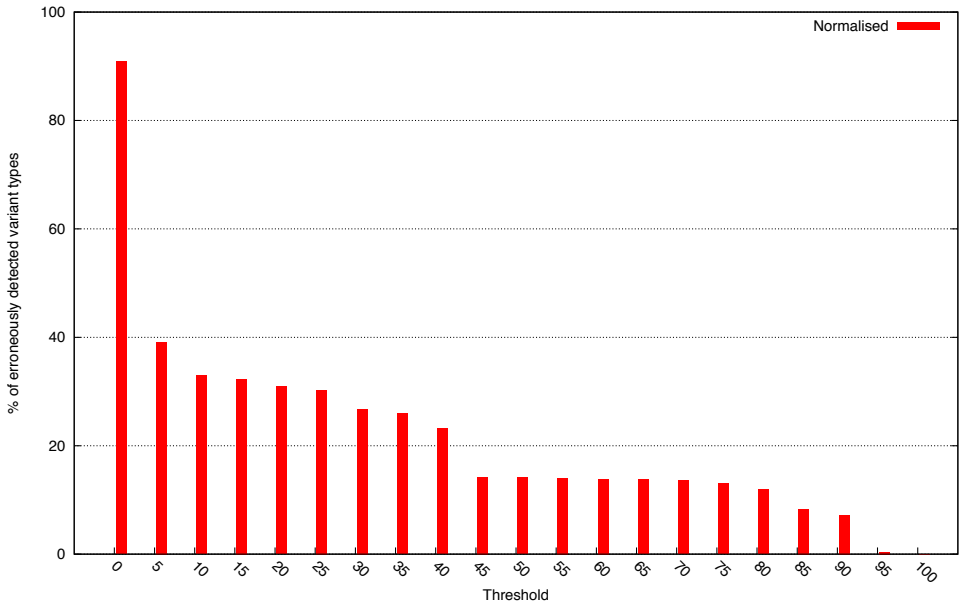


Figure 5.11: *Proportion of erroneously detected variants (types) in the Innsbruck Letters corpus which are subsequently automatically normalised at different normalisation thresholds.*

The results presented in this section have shown that the automatic normalisation procedure developed has the potential to be useful in dealing with a large proportion of the spelling variation found in EModE texts. At the same time, high precision of normalisation is maintained, and if precision is prioritised through a high normalisation threshold, then very few errors should be introduced during normalisation. Certainly, the normalised texts will be far closer to a modern equivalent version which, following the evaluation presented in Section 3.2, will improve the accuracy of subsequent corpus linguistic methodology. It is also worth noting that the automatic normalisation performed may be a first step in fully normalising the texts. Corrections and further normalisation could be performed manually using the interactive normalisation tool described in Section 4.3.1. In the next section, a case study is presented detailing the use of the full set of normalisation procedures detailed in Chapter 4, including the training procedure and automatic normalisation threshold already evaluated.

5.3 Normalising the Early Modern English Medical Text Corpus

In 2010 the Early Modern English Medical Texts (EMEMT) corpus was released (Taavitsainen & Pahta, 2010) as the second instalment of the Corpus of Early English Medical Writing (CEEM) (Taavitsainen & Pahta, 1997). The corpus contains two million words of text dated 1500–1700 from the specific domain of science and medicine. Two versions of the texts were included in the released corpus, one containing the original transcribed texts, the other containing the same texts but with spelling variation (partially) normalised. These were produced using the normalisation tool, VARD 2, described in Section 4.3. A full discussion of the process undertaken to normalise the texts is given by Lehto *et al.* (2010), a summary of the steps taken is provided here along with details of some of the issues encountered[9].

Before any normalisation was performed, 18.22% of tokens and 37.86% of types in the corpus were detected as variants (results are shown in Table 5.5). The aim of normalisation was to reduce this number as much as possible in order to have a version of the corpus containing much less spelling variation. This could in turn be used to produce more accurate searches as well as improving the performance of key word and collocational analysis (see Sections 2.1.3 and 3.2).

	Tokens		**Types**	
	Freq.	**% of total**	**Freq.**	**% of total**
Words	2,017,534		442,941	
Detected variants	367,597	18.22%	167,685	37.86%

Table 5.5: *Detected variants in the EMEMT corpus.*

At two million words, manually normalising each text in EMEMT was deemed too time-consuming. Therefore, texts were automatically normalised after training was performed by manually normalising a representative sample of the corpus. The process undertaken to perform this training will be described in Section 5.3.1. With the training complete, an initial automatic normalisation was performed,

[9]This section will have some overlap with Lehto *et al.* (2010), which was co-authored by Anu Lehto, myself, Maura Ratia and Paul Rayson.

the results of this were analysed and changes made to the normalisation tool to improve performance, this process will be described in Section 5.3.2. In order to balance the precision and recall of the final automatic normalisation, it was important to set a suitable normalisation threshold, the process of choosing this threshold will be described in Section 5.3.3. The final results of the full normalisation procedure are given in Section 5.3.4.

5.3.1 Training Through Manual Normalisation

Training was completed in two stages by Anu Lehto[10]. Initially, 24,000 words of training material were chosen by Anu from each category and fifty-year time period of the corpus. These were then manually normalised using the interactive mode of VARD 2 (see Section 4.3.1). Additionally, 24 samples of 500 words (12,000 words in total) were generated by randomly selecting small portions of text from the corpus (minus the texts which already had samples normalised), this was achieved by utilising the same method used to select training samples from the Innsbruck Letters corpus (see Section 5.2.1). These additional samples were also manually normalised by Anu using VARD 2.

During training, decisions were made to influence the final automatic normalisation and thus establish conventions for the normalisation procedure. These included:

- Not to normalise (archaic) personal pronouns such as *thou*, *ye* and *thee* as normalisation would affect their meaning. Although variants of the standard spellings can be normalised, e.g. *thyne* was normalised to *thine*.
- Archaic word endings of verbs such as *-th/-eth* were normalised to modern equivalents. For example, *sayth* was normalised to *says*.
- However, the high frequency *doth* and *hath* were not normalised because they refer to the singular and plural forms in the corpus and normalisation would mean this distinction was lost.

[10]From the Scientific Thought-styles team at the Research Unit for Variation, Contacts and Change in English at the University of Helsinki. It was more appropriate for Anu to perform the manual normalisation due to her familiarity with the texts being normalised.

- Abbreviations were left as found, e.g. *&* (representing *and* and *et*) and *Arist.* (*Aristotle*).

36,000 tokens of manually normalised text were now available for training the normalisation tool (see Section 4.3.3). With the initial set of training samples being equally spread across categories and time, and the second set of samples being made up of segments randomly distributed throughout the corpus, the set of training samples can be said to be representative of the corpus as a whole. The amount of training data was deemed sufficient following the evaluation of the training procedure on the Innsbruck corpus (see Section 5.2.1) showing that 75% of the performance improvement through training was achieved after just 34,000 tokens.

5.3.2 Initial Results and Analysis

With the training in place, an initial automatic normalisation of the entire EMEMT corpus was performed with an intermediate version of VARD 2, as described in Baron & Rayson (2009). The normalisation procedure in this version is very similar to that summarised in Section 4.2.4, with a notable exception being the procedure for improvement through training of the known variants list method, as described in Section 4.2.3. In the version used here, training normalisations are simply added to the known variants list if not already present, with no record kept of which specific normalisations have been successful. A normalisation threshold of 80% was chosen with the aim of keeping precision of normalisation as high as possible, whilst not reducing recall too much. The number of normalisations made and how many detected variants remain are shown in Table 5.6. 60.62% of the detected variant tokens (49.71% of detected variant types) in the corpus have been normalised, leaving 6.85% of the corpus's tokens still being detected as variants. What is not known is how often the normalisations made are correct, although the high threshold used, together with the evaluation of precision in the Innsbruck Corpus (see Section 5.2.2), indicate that the number of mistakes should be fairly low.

	Tokens		Types	
	Freq.	**% of total**	**Freq.**	**% of total**
Words	2,017,534		442,941	
Remaining (detected) variants	138,110	6.85%	83,151	18.77%
Normalised variants	222,842	11.05%	83,351	18.82%

Table 5.6: *Remaining and Normalised variants in EMEMT corpus after initial automatic normalisation.*

In order to assess the performance of the automatic normalisation and to attempt to find ways of improving performance, a qualitative analysis of the most frequently occurring spelling variants still remaining was undertaken by Anu Lehto. All variant word types with a raw frequency above 65 were scrutinised, and words relating to medical terminology were examined if their frequency was greater than 40. This resulted in 4,698 types being checked. Any missed normalisations or incorrectly made normalisations were noted and a list of issues created. In total, 126 missed normalisations and 20 erroneous normalisation were listed, the appropriate normalisation was also provided in each case. Context of each occurrence was examined to check for ambiguity between possible normalisations or between leaving the variant in its original form and normalising to another form.

For each issue found, an investigation was undertaken to establish why the normalisation was not made correctly and changes were made to ensure that normalisation could be completed correctly with a repeated automatic normalisation of the corpus. At this point, the completed training and automatic normalisation procedure (as detailed in Chapter 4) could be tested on the listed issues. The same set of training was used and automatic normalisation of each case attempted with a 75% normalisation threshold (the selection of this threshold will be explained in Section 5.3.3). Of the 126 missed normalisations, it was found that 61 were now successfully normalised by the fully developed procedure (e.g. *dramme* → *dram*). However, a single previously missed normalisation (*soden* → *sodden*) was now incorrectly normalised (to *seethed*). Of the remaining 64 missing normalisation cases, the following reasons were found for the correct normalisation not being made:

- In 11 cases, the top ranked candidate was correct, but the confidence score was slightly below the 75% normalisation threshold (all >65%). Example: *syckenes* → *sickness*.

- 22 cases did not have the correct normalisation as a candidate, or did but only at a low confidence score. Example: *sirrup* → *syrup*.

- There were 7 cases where the normalisation was not present in the modern word list, hence could not be found as a candidate. Example: *vnguentum* → *unguentum*.

- The remaining 24 cases were not normalised because the variant was not initially detected, i.e. they were real-word variants. Example *bin* → *been*.

In order for these relatively high frequency variants to be automatically normalised, manual changes were made to the known variants list and modern word list as follows:

- In 9 cases the normalisation was not made due to (probably erroneous) entries in the known variants list reducing the confidence score of the correct normalisation. These were removed from the known variants list. 7 of the removed entries were from the original default list (e.g. *mannes* → *man*, should be *man's*) and 2 were added to the list during training (e.g. *sheweth* → *shews*, should be *shows*).

- For the 7 cases where the normalisation could not be found as a candidate due to it not being available in the modern word list, each was added to the dictionary.

- The 24 real-word spelling variants were dealt with by simply removing the variants from the modern word list. This was safe to do as the context of each occurrence was checked during the qualitative analysis. For example, *wilt* was removed because all occurrences of the word were found to be variants of *will*.

- For the remainder of cases, and also as an additional step in some of the above cases, one instance of the normalisation was added to the known

variants list to allow it to be found as a candidate. Example: *vnces* → *ounces*.

For the 21 cases of erroneous normalisations being made (20 from the original list plus the single case added from the missed normalisations), all but one were correctly detected variants which required normalisation. However, another case (*sonne* → *son*) was ambiguous in what normalisation should be made (*son* or *sun* could be the appropriate modern equivalent). In this case the decision was made to leave occurrences in the variant form as automatic normalisation would not be able to distinguish which modern equivalent was more appropriate (context would need to be taken into account). The normalisation *sonne* → *son* was therefore removed from the known variants list[11], this resulted in no candidate normalisation for *sonne* achieving a high enough confidence score (above 75%) for automatic normalisation. For the one case where the variant was detected erroneously (*cresses* → *crosses*), the original form was added to the modern word list to stop the word being detected as a variant and subsequently normalised. One of the remaining cases was now normalised correctly due to a previous correction: *vvee* had been normalised to *wee*, when *we* is the correct modern equivalent, but *wee* was removed from the modern word list earlier as a real-word spelling variant. The remaining 18 cases were all variants normalised to an incorrect modern form due to (probably erroneous) entries in the known variants list, these entries were removed from the known variants list to allow for the correct normalisation to be made. 14 of the entries removed are included in the original default list (e.g. *howse* → *hows* – should be *house*) and 4 were added during training (e.g. *phisitio~s* → *physitions* – should be *physicians*).

Anu's analysis also included a list of several words which were not normalised, but due to ambiguity in whether normalisation should occur or in the number of possible normalisations that are appropriate, normalisation cannot be completed automatically. Some examples include:

- *bee* should nearly always be normalised to *be*, but a few instances of *bee* refer to the insect.

[11] *sonne* → *sun* was not present in the known variants list so must not have been encountered during training, if equal (or similar) numbers of each normalisation were made during training then it is very likely that the automatic normalisation would not be made due to the ambiguity reducing the confidence score for both candidates.

- Similarly, in the majority of cases *dye* should be normalised to *die*, but a small number of occurrences exist where the modern *dye* is the intended word.
- *flegme* nearly always should be normalised to *phlegm*, but a single case was found where *fleam* was the appropriate modern equivalent.
- the surname *Boyle* is also often a variant of *boil*.
- *mete* could already be in its modern form, but is also often a variant for both *meat* and *meet*.
- *heed* is commonly already in its modern form, but it also frequently occurs as a variant of *head*.

If the most common decision was taken as the correct procedure (e.g. always assume that *dye* should be normalised to *die*) then in the majority of cases the normalisation would be correct, and thus increase the proportion of variants normalised. However, in the few cases where that decision is incorrect an error will be introduced into the text – clearly undesirable. Unfortunately, automatic normalisation of variants of this type are impossible without considering the context in which they occur. One exception was made in the case of *bee*. 1,738 instances of *bee* were found in the corpus, of which only 5 were found to refer to the insect. Due to the high frequency of *bee* as a variant of *be*, it was decided to normalise all instances of *bee* to *be* and then manually post-edit the 5 cases where *bee* should remain. To achieve the automatic normalisation of *bee*, the word was simply removed from the modern word list so it would be detected as a variant.

5.3.3 Selecting a Normalisation Threshold

With a fully developed normalisation procedure trained with the manual normalisations described in Section 5.3.1 and the changes made indicated in the previous section, automatic normalisation of the entire EMEMT corpus could now be repeated. However, a normalisation threshold needed to be chosen to balance the precision and recall of automatic normalisations made (as evaluated in Section 5.2.2). In order to establish what threshold should be used, a randomly selected quarter of the training samples (9,000 words) was used as test data to

calculate the precision and recall of automatic normalisation. This was performed in the same manner as presented in Section 5.2.2. Training was first performed with the remaining three quarters of training samples (27,000 words) and none of the changes described in the previous section applied (thus avoiding any overlap of training and testing data). The results are given in Figure 5.12 for tokens and Figure 5.13 for types.

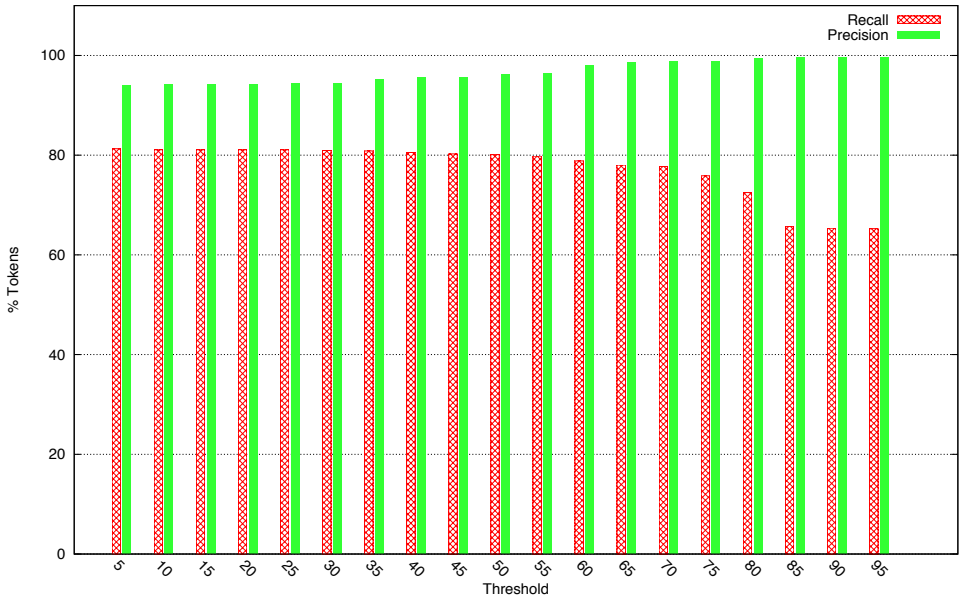


Figure 5.12: *Precision and recall of automatic normalisation on a 9,000 word sample of the EMEMT corpus at different normalisation thresholds (tokens).*

Whilst the precision and recall percentages look extremely promising, it should be noted that whilst the sample tested is a random sample of the entire corpus, thus arguably representative, its small size (9,000 tokens) equates to only 0.45% of the 2 million tokens in EMEMT. The results can, however, be used as an indicator of how the normalisation threshold will affect recall and precision and be used as a guide to select an appropriate threshold to use when automatically normalising the whole corpus. The compilers of EMEMT chose to use an automatic normalisation threshold of 75%, which, for the small test sample, yielded a recall score of 75.84% for tokens (72.95% for types) and a precision score of 98.83% for tokens (98.36% for types). High precision was prioritised in order to avoid words being normalised incorrectly.

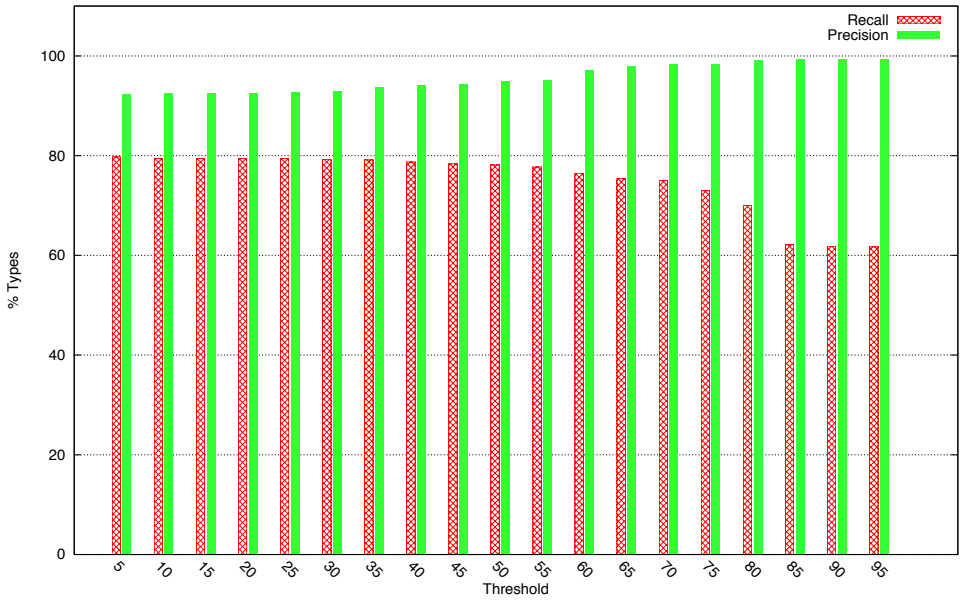


Figure 5.13: *Precision and recall of automatic normalisation on a 9,000 word sample of the EMEMT corpus at different normalisation thresholds (types).*

5.3.4 Summary and Results of Final Automatic Normalisation

For the production of the final released normalised version of the corpus, the VARD 2 tool (introduced in Section 4.3) was used to process the entire corpus. It was first trained with the manually standardised samples described in Section 5.3.1 and manual changes were made to the known variants list and modern word list as described in Section 5.3.2. VARD 2's batch processing mode (see Section 4.3.2) was then used with a normalisation threshold of 75% set (see Section 5.3.3) to automatically normalise each text in the corpus. XML tags were introduced to the texts for every normalisation made in order to keep a reference to the original form (see Section 4.3.1).

The final automatic normalisation results in 72.89% of detected variant tokens (61.04% of detected variant types) being normalised. The remaining detected variants after normalisation are shown in Table 5.7. As can be seen, less than 5% of the tokens in the normalised corpus are now detected as variants, a reduction of 13.28% from the original text.

	Tokens		Types	
	Freq.	**% of total**	**Freq.**	**% of total**
Words	2,017,534		442,941	
Remaining (detected) variants	99,649	4.94%	65,317	14.75%
Normalised variants	267,948	13.28%	102,368	23.11%

Table 5.7: *Remaining and Normalised variants in EMEMT corpus after final automatic normalisation.*

Of course, the percentage of tokens which are still variants is only an estimate. As highlighted in Section 5.1, the detection procedure has some failings with regards to real-word spelling variants not being detected and extra variants being erroneously detected. Although these two problems should, to an extent, offset each other. EMEMT will obviously contain a large amount of specialised medical terminology. It is likely that the more general modern word list used in the detection procedure will not contain many of these terms and will hence mark the words as variants. Furthermore, EMEMT is known to contain large amounts of Latin vocabulary, the majority of this will also not be covered by the modern word list used. This is exemplified with the proportion of variants normalised differing between texts. The text *Uery brefe treatise* (by Christopher Langton, 1547) has 89.15% of its variant tokens normalised (78.88% of variant types), whereas the text *Names of herbes* (by William Turner, 1548) only has 57.64% of its variant tokens normalised (31.94% of variant types). Turner's text is known to list alphabetically plants with their descriptions and specific names in several languages. The special terminology and foreign language text will result in many words not being present in the detection method's modern word list and thus marked as variants (42.99% of the original tokens are detected as variants, 80.15% of types). The normalisation method will not have seen the vast majority of these words as variants during training, so should (correctly) not normalise them, hence the low proportion of variants normalised. Langton's text is more general in its scope and vocabulary and contains only a few Latin terms. Therefore, the number of words likely to be erroneously marked as variants will be much lower (31.02% of the text's tokens are originally marked as variants, 65.08% of types). As more of the variants detected

are likely to actually be variants, more should be (correctly) normalised; which the higher percentage normalised demonstrates.

Regardless of the precise amount of spelling variation remaining, it is clear that a large amount of the spelling variation has been normalised successfully. The precision of the normalisation should be high due to the high normalisation threshold chosen. Evaluation of the normalisation threshold, albeit on a small sample, showed that very high precision was achieved. This also corroborates with the high precision scores found with the Innsbruck evaluation (see Section 5.2.2). Furthermore, only a very small number of erroneous normalisations were found in the qualitative analysis of the most frequent spelling variants. Whilst a fully normalised and manually checked corpus would be ideal for the release of the EMEMT corpus, the automatically normalised texts produced here are a reasonable substitute. It has been shown previously (Section 3.2) that even dealing partially with spelling variation will increase the accuracy of subsequently used corpus linguistic tools.

5.4 Chapter Summary

The aim of this chapter was to evaluate the developed EModE spelling normalisation methodology described in Chapter 4. Of particular interest was how well automatic normalisation could be performed, as accurate automatic normalisation would enable the processing of the large EModE corpora available. To this end, evaluations of three key aspects of the automatic normalisation procedure have been performed.

How accurately variants are detected was established with three different sources of spelling variants. As well as the real-word error rates already evaluated in Section 3.3.1, it was confirmed that extra words were also erroneously detected as spelling variants. As already discussed in Section 3.1, these are likely to include proper nouns, words from other languages (such as Latin and French), and archaic and obsolete words such as *betwixt*. However, despite these problems, the dictionary lookup method was shown to achieve fairly high recall and precision scores of around 90%. Using larger or smaller dictionaries was shown to only improve recall or precision metrics slightly, and always at the expense of the other

metric. For any further improvement to be made on these figures, contextual information would need to be incorporated (as described in Section 2.2.4).

The next key aspect of the normalisation procedure evaluated was the effect of training. Being able to train the normalisation procedure to be able to deal with a particular corpus and improve the performance of normalisation is a key concern due to the variety of EModE texts and corpora available. It has been highlighted throughout this thesis that building a static and generic EModE spelling normalisation tool would be ill-advised due to each corpus having its own characteristics. The training procedure was evaluated with the Innsbruck Letters corpus, which provides a valuable source of manually checked normalisations to compare automatic normalisations against. It has been shown that, as increasing amounts of training data are seen by the system, the proportion of spelling variants which can be successfully normalised increases. Importantly however, high precision is maintained throughout the training process, in fact the training improves this also, although to a lesser degree. It was also shown that the improvement in performance made by training decelerates as more training data is seen, with the bulk of the performance increase made before 40% of the training samples were processed.

The final aspect of the normalisation procedure evaluated was the control of automatic normalisation precision and recall through the normalisation threshold. Analysis showed that higher precision could be achieved at the expense of recall, and vice versa. It was found that with training and a normalisation threshold of 75%, automatic normalisation of the Innsbruck test set achieves 62% recall and 95% precision. Using different thresholds yields a whole range of results, with recall ranging from less than 2% (95% threshold) to above 75% (0% threshold) and precision ranging from 81.62% (0% threshold) to 97.39% (95% threshold). How spelling variant detection errors impact on the full automatic normalisation procedure was also evaluated in this section. It was found that if real-word spelling variants were somehow detected successfully, then the majority would be successfully normalised. It was also found that extra erroneously detected variants were not normalised in the majority of cases; only 14% of the extra variants were normalised with a normalisation threshold of 75%, this equates to just 0.24% of all tokens in the corpus.

Also presented in this chapter was a case-study of the use of the developed spelling normalisation procedures in the release of the EMEMT corpus. The developed software, VARD 2 (Section 4.3), was used to manually normalise representative training samples, which were then used to inform the automatic normalisation of the entire corpus – also performed by VARD 2. The final normalisation procedure reduced the proportion of detected spelling variant tokens from over 18% to below 5%, with just under 73% of spelling variants dealt with. The resulting normalised texts have been released, alongside the original texts, as an alternative version of the corpus which can be used to achieve more accurate results for searches and advanced corpus linguistic techniques such as key word and collocation analysis.

The results presented in this chapter address *RQ 3* (Section 1.2), with it being shown that a large amount of spelling variation can be dealt with by the developed normalisation tool, and with high precision. This was particularly apparent in the release of the normalised EMEMT corpus, which had a large amount of its spelling variation automatically normalised by the tool developed. Throughout the evaluation, it has been shown that high precision of normalisations is maintained, even when normalisation performance is increased in terms of recall through the training procedure. Furthermore, the confidence scores and normalisation threshold allow for the control of the balance between precision and recall, depending on the user's needs. Very high precision scores can be achieved with a high normalisation threshold set, although this will inevitably reduce the number of normalisations made.

Chapter 6

Conclusions

The final chapter shall conclude the thesis with a summary of the research undertaken and its findings. This will include reviewing the research questions posed at the beginning of the thesis and assessing how well they have been addressed. The various contributions made by the research shall then be reviewed to assess the impact of the work undertaken. Finally, the thesis will be completed with a look to future avenues of research which could be explored to extend upon this thesis and address its limitations.

6.1 Summary of Work

The focus of the first strand of research was to establish a better understanding of both the characteristics of EModE spelling variation and the problems it creates for corpus linguistic analysis. This began in the background research (Chapter 2) with an overview of the EModE period, with particular attention given to its inherent spelling variation and the reasons behind it. An overview was also given of the EModE corpora available and the potential issues caused by historical spelling variation on the performance of standard corpus linguistic methodology.

Many of the observations presented in previous research were based on assumptions and qualitative analysis. The aim of Chapter 3 was to provide quantitative analyses to verify and extend these observations. Several analyses were performed, beginning with an examination of the levels of spelling variation present in several EModE corpora. For the first time (on such a large scale), it was shown that the corpora examined contained a significant number of spelling variants and that the number of spelling variants reduced over the EModE period.

The amount of spelling variation present, particularly in earlier texts, served to highlight how difficult the task of spelling normalisation is – for several samples, over half of the tokens present were detected as spelling variants. Next, the effect of these high levels of spelling variation on the performance of corpus linguistic methodology was examined, with the evaluation of two automated corpus linguistic techniques. Part-of-speech (POS) tagging was investigated first and it was found that tagging accuracy reduced substantially when applied to EModE texts. It was also found that normalising spelling variation both manually and automatically substantially increased tagging accuracy, proving that the spelling variation was having a detrimental effect on performance. Key word analysis was also investigated by comparing key word lists produced before and after spelling variation was normalised. It was found that the ranking of key words was considerably affected by spelling variation, and that as spelling variation reduces, the ranking correlation between the two key word lists improves. These two evaluations confirmed that that spelling variation has a negative impact on the accuracy of corpus linguistic methodology. Combined with the finding of high levels of spelling variation in several EModE corpora, a strong case has been made for the necessity of spelling normalisation to reduce the number of spelling variants in EModE texts. Performing the POS tagging or key word analysis over normalised versions of the texts would improve the accuracy of results.

The second strand of research centred on the development of a spelling normalisation tool which could alleviate the highlighted problems caused by spelling variation. In Section 2.2, previous research dealing with modern spelling problems was discussed, with focus on the issues surrounding the detection of spelling errors and finding suitable corrections. Several previous studies which utilised these modern spellchecking techniques and other methods to deal with historical spelling variation (in English and other languages) for various purposes were introduced in Section 2.3. None were found to deal with the precise task of automatically normalising spelling in EModE texts sufficiently. Although, there were several cases of modern spellchecking techniques being adapted to historical spelling variation, with some success.

Through the background research, several specific issues and assumptions related to modern spellchecking were highlighted. In Section 3.3, how these issues and assumptions applied to EModE spelling variation was investigated. One

prominent area of research in modern spellchecking is the problem of detecting and correcting real-word spelling errors (see Section 2.2.4). An analysis was performed to assess the comparable problem of *real-word spelling variants* in EModE texts. Three sources of EModE spelling variants were examined and the number of variants which matched modern words was counted in each. It was found that the proportion of real-word spelling variants counted in the three sources was between 9.5% and 11.6%; much less than the proportion of spelling errors which equated to real-word errors found in modern texts. This was an important result as it substantially reduced the need for the developed EModE spelling normalisation tool to require contextual information when detecting spelling variants to normalise.

Further investigations of EModE spelling variants were conducted using the DICER analysis tool, the development of which is described in Section 4.4. An analysis was carried out to establish whether two 'rules-of-thumb' commonly used in modern spellchecking could be applied to EModE spelling normalisation. One assumption used in many applications is that the large majority of spelling errors will be a single edit away from the intended word, making correction of errors much simpler. It was found that the proportion of EModE spelling variant normalisations requiring more than one edit was around 40%. Clearly, just searching for modern equivalents that are one edit away from the spelling variant would severely restrict normalisation performance. Another assumption commonly used, particularly in phonetic matching techniques, is that the majority of spelling errors will be correct in the first letter. It was found that this assumption holds for EModE spelling variation, with less than 7% of spelling variants having variation in the first character (slightly less than the percentages found for modern spelling errors). This finding allows for methods utilising this assumption to be applied to EModE spelling variation more effectively.

Rule-based approaches are commonly used for spelling related issues such as OCR and typing errors. DICER was used to investigate character edit rules to establish whether such approaches would be applicable to EModE spelling normalisation. Several specific character edit rules were found which corroborated previous observations of common patterns in EModE spelling variation; such as the interchangeability of *i* / *y* and *u* / *v*, as well as the inconsistency in the presence of a final *e*. Several other rules were also found for the three spelling

variant datasets analysed, these differed from the rules found in DICER analyses of modern spelling errors. The findings suggest that a rule-based approach could be used to find variant normalisations. However, it was also found through the DICER analyses, and elsewhere, that different sources of EModE spelling variation have their own characteristics and there are many subtle differences in the typical properties of spelling variants present in different EModE corpora. This diversity between EModE texts highlights the need for any spelling normalisation solution to be adaptable, so that a wide range of EModE texts can be dealt with effectively.

In Chapter 4, the actual development of the spelling normalisation procedures was described. Decisions made during this development have been based on the characteristics of EModE spelling variation previously established. For example, the spelling variant detection method involves the looking up of each word from a text in a modern word list. Each word is dealt with in isolation and no context is taken into account. The decision not to include contextual information was based on the evaluation highlighted above, which indicated that the problem of real-word spelling variants in EModE is much less severe than the problem of real-word spelling errors in modern spellchecking. After finding spelling variants, the normalisation procedure consists of several steps. Firstly, a list of candidate normalisations is created using three methods: a known variants list containing previously seen normalisations and a manually created set of variant–normalisation pairs (adapted from Rayson *et al.*, 2005), a phonetic matching technique based on the Soundex algorithm, and a character edit rules based approach. The list of candidates is then ranked based on a confidence score which is calculated using an edit distance similarity measure, the predicted recall and precision of each candidate and a cumulative recall and precision based on previous normalisations seen through training. The ranked list of candidate normalisations can then be suggested for the variant in an interactive setting. Alternatively, for automatic normalisation, the highest ranked candidate for each variant can be used.

The developed methods have been incorporated into a spelling normalisation tool, VARD 2, which can be used to both manually and automatically normalise spelling variation in single texts or entire corpora. The manual processing mode, along with an additional training mode and several customisation options, allows VARD 2 to be adapted and trained for a particular corpus. This adaptability is a

key factor due to the wide range of EModE corpora available and the previously discussed differences between the spelling variation they contain. The batch processing mode offered by the tool allows for multiple texts to be automatically normalised with a normalisation threshold set which dictates what a candidate's confidence score must reach before normalisation occurs. Texts outputted from VARD 2 contain normalisations marked-up with XML tags, thus retaining the original spelling. This is essential to maintain linguistic authenticity of the text; normalisation is for the purpose of improving the performance of automated corpus linguistic tools, which will process the modern equivalents instead of the variant spellings. The XML markup also allows for the exploration of spelling patterns, such as those found in the various analyses presented in Chapter 3.

Chapter 5 presented the evaluation of the spelling normalisation procedures developed. The spelling variant detection method was evaluated with precision and recall scores of around 90% found. The recall scores relate directly to the levels of real-word spelling variants found to exist in EModE corpora, whereas the precision scores relate to a different problem of extra spelling variants being erroneously detected – these are likely to include proper nouns, words from other languages (e.g. Latin and French) and archaic and obsolete words such *betwixt* and *howbeit*. The recall score of 90% represents an upper bound to automatic normalisation coverage, as a normalisation cannot be made if the variant is not first detected. The extra variants erroneously detected will only pose a problem if they are subsequently normalised – clearly any normalisation would be an error. The effect of dictionary size was also evaluated for the variant detection method. It was found that the dictionary used by default in VARD 2 was suitable in terms of its precision and recall balance. Larger dictionaries were found to only offer small improvements in precision but with substantial drops in recall, whilst smaller dictionaries only contained marginal increases in recall at the expense of larger decreases in precision.

The automatic normalisation procedure was evaluated next, with focus on the effect of training the tool with manual normalisations and adjusting the normalisation threshold to balance precision and recall. The Innsbruck Letters corpus, which has been automatically normalised and then manually checked (Markus, 2000; 2002), was used for the evaluations. The normalisations found in the corpus can be compared to the automatic normalisations made by VARD 2,

with recall and precision scores calculated. The training procedure was evaluated by training VARD 2 on 1,000 word random samples from half of the Innsbruck corpus and after each sample measuring the recall and precision on automatic normalisation of the other half of the Innsbruck corpus. It was found that with each training sample, automatic normalisation performance improved. A performance increase in terms of recall was most notable, with precision gaining only slight increases – albeit from a considerably higher starting score (compared to recall). Recall increased from 46.3% to 67.5% and precision from 90.3% to 93.3%. It was also found that the bulk of performance improvement was gained in earlier training, with 75% of the performance increase observed before 40% of the training samples were processed. The normalisation threshold was evaluated with the Innsbruck data also. After the tool had been trained on the same half of the corpus, the other half was used as test data to determine the precision and recall of automatic normalisation with different normalisation thresholds set. Results showed recall ranging from less than 2% (95% recall) to above 75% (0% threshold) and precision ranging from under 82% (0% threshold) to over 97% (95% threshold). With a threshold of 75% set, 62% recall is achieved with 95% precision – encouraging scores for the difficult task in hand. Precision is likely to be of greater importance when automatically normalising texts, so the high precision scores achievable are particularly promising. The extra erroneously detected spelling variants will affect the precision scores presented if the words are subsequently normalised, with real-word errors being introduced into the text. However, an additional evaluation found that relatively few (less than 15% even at a mid-ranged normalisation threshold) are automatically normalised. This equates to 0.24% of all tokens in the Innsbruck corpus.

The final evaluation presented centred on the Early Modern English Medical Texts (EMEMT) corpus, which was recently released with a normalised version alongside the original transcribed texts. The normalisation procedure was completed using VARD 2. Its interactive mode was used to manually normalise a representative training sample, which was then used to inform the automatic normalisation of the whole corpus with the tool's batch processing mode. The final normalised texts have had 73% of the spelling variants detected within them normalised, reducing the number of detected variants from over 18% to below 5%. The use of the normalisation procedures developed on an actual released

EModE corpus demonstrates the value of the research undertaken. Furthermore, a qualitative review of the normalisations made by VARD 2 was performed by a member of the EMEMT compilation team (see Section 5.3.2). Whilst only the most frequently occurring variants were assessed, very few errors in the normalisations made were found to exist. This indicates again the high precision achieved during automatic normalisation.

6.2 Research Questions Revisited

We now turn to how the work undertaken has addressed the research questions established in Section 1.2, repeated here for convenience.

RQ 1 *How extensive is Early Modern English spelling variation in terms of the levels of variation appearing in Early Modern English corpora and how large an impact does this spelling variation have on corpus linguistic methodology?*

This research question has been addressed in Chapter 2 and Chapter 3. The high levels of spelling variation had already been commented upon by various researchers (e.g. Görlach, 1991; Vallins & Scragg, 1965) but only quantified on a small scale by Schneider (2001) with the ZEN corpus (1670–1799). Section 3.1 describes the undertaking of a large-scale quantitative analysis of the spelling variant levels in several EModE corpora dating between 1410 and 1800. It was shown that the predicted high-levels of spelling variants did exist in all corpora, particularly in earlier periods. It was also shown that spelling variant levels decline significantly throughout the EModE period, slowing as English spelling becomes standardised by the end of the 18th century. The potential impact of this spelling variation on corpus linguistic methodology was first discussed in Section 2.1.3, and then evaluated in Section 3.2 for part-of-speech (POS) tagging and key word analysis. It was found that the accuracy of POS tagging was reduced to below 82% on a 5,000 word sample of Shakespeare text. However, dealing with spelling variation increased accuracy to nearly 89%. For key word analysis, two key word lists were produced for an EModE corpus before and after spelling normalisation took place. A distinct impact on the ranking of words in the key word lists was observed, which could only be attributable to the spelling variation contained in

the original texts. It was also found that as the levels of spelling variation reduces, less impact on the key word list rankings is observed.

RQ 2 *What are the characteristics of Early Modern English spelling variation and how will these affect the application of modern spellchecking techniques to historical spelling normalisation?*

Through the background research (presented in Section 2.2), several key areas of modern spellchecking were established where the properties of EModE spelling variation needed to be known to judge how modern spellchecking techniques could be applied to EModE spelling normalisation. In Section 3.3, these properties were investigated using various sources of spelling variant normalisations – some from the use of the developed normalisation tool and others from external sources. Also utilised was the same variant detection method used in the final normalisation tool, and the DICER analysis tool (see Section 4.4). Decisions in the development of the spelling normalisation methods (described in Chapter 4) based on these findings included:

- Not to prioritise the inclusion of contextual information during spelling variant detection due to the finding that far fewer real-word spelling variants exist in EModE texts than real-word spelling errors exist in modern texts.
- To use edit distance as an indicator of a correct spelling variant normalisation candidate, but to not limit normalisation suggestions to modern words which are only one edit away from the variant form – this was due to the finding that around 40% of EModE spelling variant normalisations were more than one edit away.
- To utilise the assumption in phonetic matching that, in the vast majority of cases, the first letter is 'correct', this following the finding that less than 7% of spelling variants required action on the first letter.
- To use a rule-based method with specific character edit rules relating to EModE spelling variation – DICER analysis showed that such character edit rules exist.

- To produce a normalisation tool that could be trained and customised to adapt it to different texts and corpora, this was following various analyses indicating subtle diversity between different EModE spelling variation sources.

RQ 3 *What levels of performance can the developed normalisation tool achieve with different levels of training, particularly in terms of precision and recall, when automatically normalising spelling variation in Early Modern English corpora?*

This research question was addressed in Chapter 5, with various evaluations of the VARD 2 spelling normalisation tool. It was shown that not only is normalisation recall vastly improved through training, the already high precision of normalisations made also increases slightly. Balance between precision and recall can also be achieved through the setting of a normalisation threshold. With this threshold set at 75% and training performed on half of the Innsbruck Letters corpus (which contains manually checked normalisations), automatic normalisation of the remaining half of the corpus achieves a recall score of 62%, with precision of normalisation made at 95%. These encouraging levels of performance were repeated in the automatic normalisation of the EMEMT corpus which was released with the VARD 2 normalised texts alongside the original transcriptions. In the final normalisation of the corpus, 73% of the detected spelling variants were normalised. Furthermore, qualitative analysis (performed by Anu Lehto) of the normalisations made revealed very few errors.

6.3 Contributions

The following contributions can be attributed to the research presented in this thesis.

Spelling variation normalisation method and tool

The research presented in this thesis has led to the development of a spelling normalisation tool which has been shown to be useful when dealing with EModE spelling variation. Use of the normalisation tool on EModE corpora will lead to the improved accuracy of corpus linguistic methodology, but at the same time

maintain linguistic authenticity through the retention of the original spelling forms. The tool can be customised and trained to be used with different EModE corpora and an interactive mode allows for assisted manual normalisation. Automatic normalisation can be performed on any size of corpus, with precision and recall balance controllable through a normalisation confidence threshold.

Use of spelling normalisation in research projects

The usefulness of the tool for the task of spelling normalisation has been highlighted with its performance in automatically normalising the spelling variation present in the EMEMT corpus.

The normalisation tool has been made available for academic research, with significant interest expressed through over 80 requests to download the software. An early version of VARD 2 was used in Jane Demmen (2009)'s "corpus-based investigation of 'key' word clusters [...] in the dialogue of male and female characters in Shakespeare's plays", to avoid the issue of spelling variation affecting word frequencies, which would have a cumulative effect on key word clusters.

Highlighting the issues presented by historical spelling variation

It has been proven that spelling variation has a considerable impact on advanced corpus linguistic techniques. This has been shown specifically to be the case for part-of-speech annotation and key word analysis. Historical corpus linguistic researchers should be aware of these issues when performing their analyses and consider normalising the spelling variation in their data. Even dealing in part with variation will improve results, which has been shown to be possible without considerable effort.

Understanding and quantifying the properties of EModE spelling variation

Through the various analyses presented in Chapter 3, greater knowledge and appreciation of the spelling variation that exists in EModE texts can be gained. The findings can inform the development of techniques for spelling normalisation, both in the research presented in this thesis and future studies looking to deal with spelling variation. Not only this, the results offer intriguing linguistic insights into the orthography of EModE texts, providing quantitative results that corroborate

previous qualitative observations and compare historical spelling variation to modern spelling errors.

Standardising other forms of spelling variation

The VARD 2 tool is highly adaptable for use with the wide range of EModE corpora, this adaptability also allows for the tool's use to normalise spelling variation found in any language variety through customisation and training. DICER can also be used to assist in this process by finding entries for the rule-based method. For example, experiments have been undertaken with the tool(s) to normalise spelling in child language data (Baron & Rayson, 2009), second language learner data (Rayson & Baron, 2011) and SMS spelling variation (Tagg *et al.*, 2010).

Exploration of spelling patterns

The two tools developed from the research presented in this thesis, VARD 2 and DICER, enabled the analysis of EModE spelling variation described in Chapter 3. They can also be used in similar ways to perform a wide range of analyses to explore historical spelling patterns, as well as spelling patterns in other language varieties. For example, both tools have already been used for the creation of an SMS spelling taxonomy (Tagg *et al.*, 2010) and to explore the spelling trends in second language learner data (Rayson & Baron, 2011).

6.4 Future Work

The following would be interesting research avenues to explore to extend and improve upon the research presented in this thesis, and to address its limitations.

DICER and VARD combination

One potentially fruitful line of research would be to investigate incorporating the DICER analysis directly into VARD 2's training procedure. Currently, DICER can be used with VARD 2, but only in a manual process. This includes creating a DICER analysis from VARD 2's output, examining the extensive results DICER produces and deciding on a list of suitable rules. These rules can then be added to VARD 2 for use when finding normalisation candidates. Ideally, this process

would be performed automatically with new rules created as normalisations are made. The specific context of the rules would need to be taken into account, i.e. the position of rule application and the surrounding characters. Additional information would also need to be stored for each rule to keep track of how successful it has been in previous normalisations – this could potentially be achieved with a similar method to that which is currently used to keep track of the four normalisation methods' cumulative precision and recall (see Section 4.2.3).

Including contextual information

Whilst it was found that real-word spelling variants were relatively uncommon in EModE texts (compared to real-word spelling errors in modern texts), it was also found that if the spelling variants could be detected, a normalisation can often be made. Therefore, introducing contextual information in the spelling variant detection procedure could potentially increase the recall of spelling variant normalisation. Adding contextual information could also improve performance in other areas of spelling normalisation, such as the ranking of normalisation candidates. For example, words which if used as the normalisation would appear out of place in the surrounding context could be ranked lower than words which would fit. The application of contextual rules would not be a simple task; current research is far from a full solution to modern context-sensitive spelling detection and correction, and very little research has applied techniques to historical spelling variation.

Other modern spellchecking methods

There are numerous additional methods from modern spellchecking research, some of which have been described in Section 2.2, which could be useful for historical spelling normalisation. Whilst some methods would be less effective with EModE spelling variation due to its properties, as investigated in Section 3.3, some may be useful for improved variant detection and for the suggestion and ranking of normalisation candidates. Further analysis and evaluation will be required to assess different methods' applicability.

Machine learning

Whilst the training procedure used in VARD 2 produces encouraging results (see Section 5.2.1), there are several well researched machine learning methods which could be applied to ranking normalisation candidates. It would be important that any method employed did not sacrifice precision for the sake of recall, and control of the balance between precision and recall during automatic normalisation would need to be maintained through confidence scores, or similar, still being attached to candidates. Any machine learning technique would also need to be trainable as individual normalisations are made for the purposes of assisted manual normalisation.

References

Acharyya, S., Negi, S., Subramaniam, L.V. & Roy, S. (2009). Language independent unsupervised learning of short message service dialect. *International Journal on Document Analysis and Recognition*, 12(3): 175–184.

Agarwal, S., Godbole, S., Punjani, D. & Roy, S. (2007). How much noise is too much: A study in automatic text classification. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, 3–12, IEEE Computer Society, Washington, DC, USA.

Aho, A.V. & Corasick, M.J. (1975). Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6): 333–340.

Al-Sa'di, R.A. & Hamdan, J.M. (2005). "Synchronous online chat" English: Computer-mediated communication. *World Englishes*, 24(4): 409–424.

Anderwald, L. & Szmrecsanyi, B. (2009). Corpus linguistics and dialectology. In A. Lüdeling & M. Kytö, eds., *Corpus Linguistics: An International Handbook*, vol. 2 of *Handbooks of Linguistics and Communication Science*, chap. 53, 1126–1139, Mouton de Gruyter, Berlin and New York.

Archer, D. (2006). Tracing the development of 'advocacy' on two nineteenth century English trials. In M. Dossena & I. Taavitsainen, eds., *Diachronic Perspectives on Domain-Specific English*, Linguistic Insights, Peter Lang, Bern.

Archer, D., McEnery, T., Rayson, P. & Hardie, A. (2003). Developing an automated semantic analysis system for Early Modern English. In D. Archer, P. Rayson, A. Wilson & T. McEnery, eds., *Proceedings of Corpus Linguistics 2003*, 22–31, Lancaster University, Lancaster, UK.

Archer, D., Culpeper, J. & Rayson, P. (2009). Love – 'a familiar or a devil'? An exploration of key domains in Shakespeare's comedies and tragedies. In D. Archer, ed., *What's in a Word-list? Investigating Word Frequency and Keyword Extraction*, Digital Research in the Arts and Humanities, Ashgate.

Asonov, D. (2010). Real-word typo detection. In *Natural Language Processing and Information Systems*, vol. 5723 of *Lecture Notes in Computer Science*, 115–129, Springer-Verlag, Berlin and Heidelberg.

Atwell, E. & Elliott, S. (1987). Dealing with ill-formed English text. In R. Garside, G. Leech & G. Sampson, eds., *The Computational Analysis of English: A Corpus-Based Approach*, 120–138, Longman, New York, NY, USA.

Aw, A., Zhang, M., Xiao, J. & Su, J. (2006). A phrase-based statistical model for SMS text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, 33–40, Association for Computational Linguistics, Morristown, NJ, USA.

Baker, P. (2009). The BE06 Corpus of British English and recent language change. *Internation Journal of Corpus Linguistics*, 14: 312–337.

Barber, C. (1997). *Early Modern English*. Edinburgh University Press, Edinburgh, 2nd edn.

Baron, A. & Rayson, P. (2009). Automatic standardisation of texts containing spelling variation: How much training data do you need? In M. Mahlberg, V. González-Díaz & C. Smith, eds., *Proceedings of Corpus Linguistics 2009*, University of Liverpool, Liverpool, UK.

Bennett, P., Durrell, M., Scheible, S. & Whitt, R.J. (2009). Annotating a multi-genre corpus of Early Modern German. In M. Mahlberg, V. González-Díaz & C. Smith, eds., *Proceedings of Corpus Linguistics 2009*, University of Liverpool.

Bennett, P., Durrell, M., Scheible, S. & Whitt, R.J. (2010). Annotating a historical corpus of German: A case study. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner & D. Tapias, eds., *Proceedings of the Seventh conference on International*

Language Resources and Evaluation (LREC'10): Workshop 4: Language Resource and Language Technology Standards – state of the art, emerging needs, and future developments, European Language Resources Association (ELRA), Valletta, Malta.

Bhardwaj, A., Farooq, F., Cao, H. & Govindaraju, V. (2008). Topic based language models for OCR correction. In *AND '08: Proceedings of the second workshop on Analytics for noisy unstructured text data*, 107–112, ACM, New York, NY, USA.

Brants, T. & Franz, A. (2006). *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia.

Brill, E. & Moore, R.C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 286–293, Association for Computational Linguistics, Morristown, NJ, USA.

Britto, H., Galves, C., Ribeiro, I., Augusto, M. & Scher, A. (1999). Morphological annotation system for automatic tagging of electronic textual corpora: from English to Romance languages. In *Proceedings of the 6th International Symposium of Social Communication*, 582–589, Santiago, Cuba.

Burnard, L. (2007). *Reference Guide for the British National Corpus (XML Edition)*. `http://www.natcorp.ox.ac.uk/docs/URG/`.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S. & Basu, A. (2007). Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3): 157–174.

Church, K., Hart, T. & Gao, J. (2007). Compressing trigram language models with Golomb Coding. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 199–207, Association for Computational Linguistics, Prague, Czech Republic.

Church, K.W. & Gale, W.A. (1991). Probability scoring for spelling correction. *Statistics and Computing*, 1(2): 93–103.

Clark, A. (2003). Pre-processing very noisy text. In *Proceedings of the Workshop on Shallow Processing of Large Corpora at Corpus Linguistics 2003*, Lancaster University, Lancaster, UK.

Cohen, J.D. (1997). Recursive hashing functions for n-grams. *ACM Transactions on Information Systems*, 15(3): 291–320.

Cohen, W.W., Ravikumar, P. & Fienberg, S.E. (2003). A comparison of string distance metrics for name-matching tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, 73–78.

Cook, P. & Stevenson, S. (2009). An unsupervised model for text message normalization. In *Proceedings of the NAACL HLT Workshop on Computational Approaches to Linguistic Creativity*, 71–78, Assocation for Computational Linguistics, Boulder, Colorado.

Craig, H. & Whipp, R. (2010). Old spellings, new methods: automated procedures for indeterminate linguistic data. *Literary and Linguistic Computing*, 25(1): 37–52.

Craig, W.J., ed. (1914). *The Complete Works of William Shakespeare*. Oxford University Press, London.

Crystal, D. (2004). *A Glossary of Netspeak and Textspeak*. Edinburgh University Press.

Crystal, D. (2008). *Txting: The Gr8 Db8*. Oxford University Press.

Cucerzan, S. & Brill, E. (2004). Spelling correction as an iterative process that exploits the collective knowledge of web users. In D. Lin & D. Wu, eds., *Proceedings of EMNLP 2004*, 293–300, Association for Computational Linguistics, Barcelona, Spain.

Culpeper, J. (2002). Computers, language and characterisation: An analysis of six characters in Romeo and Juliet. In U. Merlander-Marttala, C. Ostman &

M. Kytö, eds., *Conversation in Life and in Literature: Papers from the ASLA Symposium*, vol. 15, 11–30, Universitetstryckeriet, Uppsala.

Culpeper, J. (2007). A new kind of dictionary for Shakespeare's plays: An immodest proposal. *SEDERI*, 17: 47–73.

Culpeper, J. & Kytö, M. (1997). Towards a corpus of dialogues, 1550-1750. *Language in Time and Space. Studies in Honour of Wolfgang Viereck on the Occasion of His 60th Birthday*, 60–73.

Damerau, F.J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3): 171–176.

Davies, M. (2008). *The Corpus of Contemporary American English (COCA): 400+ million words, 1990-present*. `http://www.americancorpus.org`.

de Grazia, M. & Stallybrass, P. (1993). The materiality of the Shakespearean text. *Shakespeare Quarterly*, 44(3): 255–283.

Demmen, J. (2009). *Charmed and chattering tongues: Investigating the functions and effects of key word clusters in the dialogue of Shakespeare's female characters*. Master's thesis, Lancaster University, Lancaster.

Dorr, B.J., Jordan, P.W. & Benoit, J.W. (1999). A survey of current paradigms in machine translation. vol. 49 of *Advances in Computers*, 1–68, Elsevier.

Driscoll, D. (2002). The ubercool morphology of internet gamers: A linguistic analysis. *Undergraduate Research Journal for the Human Sciences*, 1.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1): 61–74.

Fisher, J.H. (1977). Chancery and the emergence of standard written English in the fifteenth century. *Speculum*, 52(4): 870–899.

Fisher, J.H. (1984). Caxton and Chancery English. In R.F. Yeager, ed., *Fifteenth-Cenutry Studies*, Archon Books, Hamden, Connecticut, USA.

Fisher, J.H. (1992). A language policy for Lancastrian England. *PMLA*, 107(5): 1168–1180.

Fontenelle, T. (2006). Contextual spelling in the 2007 Microsoft Office system. Microsoft Developer Network (MSDN) blog, `http://blogs.msdn.com/b/correcteurorthographiqueoffice/archive/2006/06/05/617653.aspx`.

Fossati, D. & Eugenio, B. (2007). A mixed trigrams approach for context sensitive spell checking. In *CICLing '07: Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, 623–633, Springer-Verlag, Berlin, Heidelberg.

Fossati, D. & Eugenio, B.D. (2008). I saw tree trees in the park: How to correct real-word spelling mistakes. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis & D. Tapias, eds., *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, European Language Resources Association (ELRA), Marrakech, Morocco.

Fredkin, E. (1960). Trie memory. *Communications of the ACM*, 3(9): 490–499.

Fries, U. & Schneider, P. (2000). ZEN: Preparing the Zurich English Newspaper corpus. In F. Ungerer, ed., *English Media Texts - Past and Present*, vol. 80 of *Pragmatics & Beyond New Series*, chap. 1, 3–24, John Benjamins, Amsterdam.

Gadd, T.N. (1988). 'fisching fore weds': phonetic retrieval of written text in information systems. *Program: electronic library and information systems*, 22(3): 222–237.

Gadd, T.N. (1990). Phonix: the algorithm. *Program: electronic library and information systems*, 24(4): 363–369.

Garside, R. (1987). The CLAWS Word-Tagging System. In R. Garside & G. Sampson, eds., *The Computational Analysis of English - A Corpus-Based Approach*, 30–41, Longman, New York, NY, USA.

Garside, R. & Smith, N. (1997). A hybrid grammatical tagger: CLAWS4. In R. Garside, G. Leech & A. McEnery, eds., *Corpus Annotation: Linguistic Information from Computer Text Corpora*, 101–121, Longman, London.

Giusti, R., Jr, A.C., Muniz, M., Cucatto, L. & Aluísio, S. (2007). Automatic detection of spelling variation in historical corpus: An application to build a Brazilian Portuguese spelling variants dictionary. In M. Davies, P. Rayson, S. Hunston & P. Danielsson, eds., *Proceedings of Corpus Linguistics 2007*, UCREL, Lancaster University, Lancaster, UK.

Golding, A.R. (1995). A bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the 3rd Workshop on Very Large Corpora*, 39–53, Boston, MA, USA.

Golding, A.R. & Roth, D. (1999). A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1): 107–130.

Golding, A.R. & Schabes, Y. (1996). Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, 71–78, Association for Computational Linguistics, Morristown, NJ, USA.

Görlach, M. (1985). Renaissance English (1525-1640). In S. Greenbaum, ed., *The English Language Today*, English in the international context, Pergamon Press, Oxford, New York.

Görlach, M. (1991). *Introduction to Early Modern English*. Cambridge University Press, Cambridge.

Gotscharek, A., Neumann, A., Reffle, U., Ringlstetter, C. & Schulz, K.U. (2009). Enabling information retrieval on historical document collections: the role of matching procedures and special lexica. In *AND '09: Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, 69–76, ACM, New York, NY, USA.

Granger, S., ed. (1998). *Learner English on Computer*. Studies in Language and Linguistics, Longman, London and New York.

Granger, S. & Rayson, P. (1998). Automatic profiling of learner texts. In S. Granger, ed., *Learner English on Computer*, chap. 9, 119–131, Longman, London and New York.

Grefenstette, G. & Tapanainen, P. (1994). What is a word, what is a sentence? Problems of tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, 79–87.

Hanson, A.R., Riseman, E.M. & Fisher, E. (1976). Context in word recognition. *Pattern Recognition*, 8(1): 35–45.

Hardie, A. (forthcoming). CQPweb - combining power, flexibility and usability in a corpus analysis tool.

Hauser, A., Heller, M., Leiss, E., Schulz, K.U. & Wanzeck, C. (2007). Information access to historical documents from the Early New High German period. In L. Burnard, M. Dobreva, N. Fuhr & A. Lüdeling, eds., *Digital Historical Corpora - Architecture, Annotation, and Retrieval*, no. 06491 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Dagstuhl, Germany.

Hodge, V.J. & Austin, J. (2001a). An evaluation of phonetic spell checkers. Tech. Rep. YCS338, Department of Computer Science, University of York.

Hodge, V.J. & Austin, J. (2001b). A novel binary spell checker. In *ICANN '01: Proceedings of the International Conference on Artificial Neural Networks*, 1199–1204, Springer-Verlag, London, UK.

Hoffmann, S., Evert, S., Smith, N., Lee, D. & Prytz, Y.B. (2008). *Corpus Linguistics with BNCweb - a Practical Guide*. Peter Lang, Frankfurt am Main.

Hofland, K. & Johansson, S. (1982). *Word frequencies in British and American English*. The Norwegian Computing Centre for the Humanities, Bergen, Norway.

Ihaka, R. & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3): 299–314.

Järvelin, A., Järvelin, A. & Järvelin, K. (2007). s-grams: Defining generalized n-grams for information retrieval. *Information Processing & Management*, 43(4): 1005–1019.

Jones, M.P. & Martin, J.H. (1997). Contextual spelling correction using latent semantic analysis. In *Proceedings of the fifth conference on Applied natural language processing*, 166–173, Association for Computational Linguistics, Morristown, NJ, USA.

Jurafsky, D. & Martin, J. (2000). *Speech and Language Processing*. Prentice Hall, New Jersey, USA.

Kendall, M.G. (1938). A new measure of rank correlation. *Biometrika*, 30: 81–89.

Kernighan, M.D., Church, K.W. & Gale, W.A. (1990). A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics*, 205–210, Association for Computational Linguistics, Morristown, NJ, USA.

Knuth, D.E. (1973). *The Art of Programming*, vol. 3: Sorting and Searching. Addison-Wesley, Reading, MA, USA.

Kobus, C., Yvon, F. & Damnati, G. (2008). Normalizing SMS: are two metaphors better than one? In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, 441–448, Association for Computational Linguistics.

Koolen, M., Adriaans, F., Kamps, J. & de Rijke, M. (2006). A cross-language approach to historic document retrieval. *Advances in Information Retrieval*, 407–419.

Kruskal, J.B. (1983). An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM Review*, 25(2): 201–237.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4): 377–439.

Kytö, M. & Rissanen, M. (1993). General introduction. In M. Rissanen, M. Kytö & M. Pallander-Collin, eds., *Early English in the computer age: explorations through the Helsinki corpus*, 1–17, Mouton de Gruyter, Berlin.

Kytö, M. & Voutilainen, A. (1995). Applying the Constraint Grammar Parser of English to the Helsinki Corpus. *ICAME Journal*, 19: 23–48.

Kytö, M., Rissanen, M. & Wright, S., eds. (1994). *Corpora across the Centuries: Proceedings of the First International Colloquium on English Diachronic Corpora*, Rodopi, Amsterdam, St. Catherine's College, Cambridge.

Lass, R. (1999). *The Cambridge History of the English Language: Volume III, 1476-1776*, chap. Phonology and Morphology. Cambridge University Press, Cambridge.

Leech, G. & Smith, N. (2000). *Manual to accompany The British National Corpus (Version 2) with Improved Word-class Tagging*. UCREL, Lancaster University, Lancaster, UK, `http://ucrel.lancs.ac.uk/bnc2/bnc2postag_manual.htm`.

Leech, G., Rayson, P. & Wilson, A. (2001). *Word Frequencies in Written and Spoken English: based on the British National Corpus*. Longman, London.

Lefer, M.A. & Thewissen, J. (2007). Orthographic and morphological errors in learner writing. automatic and manual annotation methods: a match made in heaven? In *ICAME 28 Abstracts*.

Lehto, A., Baron, A., Ratia, M. & Rayson, P. (2010). Improving the precision of corpus methods: The standardized version of Early Modern English Medical Texts. In I. Taavitsainen & P. Pahta, eds., *Early Modern English Medical Texts: Corpus description and studies*, 279–290, John Benjamins, Amsterdam.

Levenshtein, V.I. (1966). Binary codes capable of correcting insertions, deletions and reversals. *Cybernetics and Control Theory*, 10(8): 707–710.

Lieberman, E., Michel, J.B., Jackson, J., Tang, T. & Nowak, M.A. (2007). Quantifying the evolutionary dynamics of language. *Nature*, 449(7163): 713 – 716.

Lopresti, D. (2008). Optical character recognition errors and their effects on natural language processing. In *AND '08: Proceedings of the second workshop on Analytics for noisy unstructured text data*, 9–16, ACM, New York, NY, USA.

Mahlberg, M. (2007). Clusters, key clusters and local textual functions in Dickens. *Corpora*, 2(1): 1–31.

Mangu, L. & Brill, E. (1997). Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning*, 734–741, Morgan Kaufmann.

Markus, M. (1999). Innsbruck Computer-Archive of Machine-Readable English Texts. In *Innsbrucker Beitraege zur Kulturwissenschaft, Anglistische Reihe*, vol. 7, Leopold-Franzens-Universitaet Innsbruck, Institut fuer Anglistik, Innsbruck.

Markus, M. (2000). Normalizing the word-forms in Ayenbite of Inwyt. In I. Taavitsainen, T. Nevalainen, P. Pahta & M. Rissanen, eds., *Placing Middle English in Context*, vol. 35 of *Topics in English Linguistics*, 181–198, Mouton de Gruyter, Berlin and New York.

Markus, M. (2002). Towards an analysis of pragmatic and stylistic features in 15th and 17th century English letters. In P. Peters, P. Collins & A. Smith, eds., *New frontiers of corpus research*, 179–198, Rodopi, Amsterdam.

Martins, B. & Silva, M.J. (2004). Spelling correction for search engine queries. In J.L. Vicedo, P. Martínez-Barco, R. Muñoz & M. Saiz Noeda, eds., *Advances in Natural Language Processing*, vol. 3230 of *Lecture Notes in Computer Science*, 372–383, Springer, Berlin and Heidelberg.

Mays, E., Damerau, F.J. & Mercer, R.L. (1991). Context based spelling correction. *Information Processing and Management: an International Journal*, 27(5): 517–522.

McEnery, A.M. & Wilson, A. (2001). *Corpus Linguistics*. Edinburgh University Press, Edinburgh, 2nd edn.

McEnery, T. (2006). *Swearing in English: Bad Language, Purity and Power from 1586 to the Present*. Routledge, London.

Mihov, S. & Schulz, K.U. (2004). Fast approximate search in large dictionaries. *Computational Linguistics*, 30(4): 451–477.

Mihov, S., Mitankin, P., Gotscharek, A., Reffle, U., Schulz, K.U. & Ringlstetter, C. (2007). Tuning the selection of correction candidates for garbled tokens using error dictionaries. In *Finite State Techniques and Approximate Search, Proceedings of the First Workshop on Finite-State Techniques and Approximate Search*, 25–30, Borovets, Bulgaria.

Mitton, R. (1987). Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information Processing and Management*, 23(5): 495–505.

Mitton, R. (1996). *English Spelling and the Computer*. Studies in Language and Linguistics, Longman, London and New York.

Mitton, R. (2008). Ordering the suggestions of a spellchecker without using context. *Natural Language Engineering*, 15(2): 173–192.

Mitton, R. (2010). Fifty years of spellchecking. *Writing Systems Research*, 2(1): 1–7.

Moore, G.E. (1965). Cramming more components onto integrated circuits. *Electron*, 38(8).

Needleman, S.B. & Wunsch, C.D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3): 443–453.

Nevalainen, T. (1997). Ongoing work on the Corpus of Early English Correspondence. *Language and Computers*, 18: 81–90.

Nevalainen, T. (2006). *An Introduction to Early Modern English*. Edinburgh Textbooks on the English Language, Edinburgh University Press, Edinburgh.

O'Rourke, A.J., Robertson, A.M., Willett, P., Eley, P. & Simons, P. (1997). Word variant identification in Old French. *Information Research*, 2(4).

Osselton, N.E. (1998). Spelling-book rules and the capitalization of nouns in the seventeenth and eighteenth centuries. In M. Rydén, I.T.B. van Ostade & M. Kytö, eds., *A Reader in Early Modern English*, Peter Lang, Frankfurt am Main.

Pagel, M., Atkinson, Q.D. & Meade, A. (2007). Frequency of word-use predicts rates of lexical evolution throughout Indo-European history. *Nature*, 449(7163): 717–720.

Pearson, K. (1904). On the theory of contingency and its relation to association and normal correlation. *Drapers' Company Research Memoirs (Biometric Series)*, 1.

Pedler, J. & Mitton, R. (2010). A large list of confusion sets for spellchecking assessed against a corpus of real-word errors. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner & D. Tapias, eds., *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, European Language Resources Association (ELRA), Valletta, Malta.

Perera, K. (1986). Language acquisition and writing. In P. Fletcher & M. Garman, eds., *Language Acquisition: Studies in first language development*, chap. 23, 494–518, Cambridge University Press, 2nd edn.

Peterson, J.L. (1980). Computer programs for detecting and correcting spelling errors. *Communications of the ACM*, 23(12): 676–687.

Peterson, J.L. (1986). A note on undetected typing errors. *Communications of the ACM*, 29(7): 633–637.

Pfeifer, U., Poersch, T. & Fuhr, N. (1996). Retreival effectiveness of proper name search methods. *Information Processing and Management*, 32(6): 667–679.

Pilz, T. & Luther, W. (2009). Automated support for evidence retrieval in documents with nonstandard orthography. In S. Featherston & S. Winkler, eds., *The Fruits of Empirical Linguistics*, vol. Volume 1 of *Process*, Mouton de Gruyter, Berlin and New York.

Pilz, T., Luther, W., Fuhr, N. & Ammon, U. (2006). Rule-based search in text databases with nonstandard orthography. *Literary and Linguistic Computing*, 21(2): 179–186.

Pilz, T., Ernst-Gerlach, A., Kempken, S., Rayson, P. & Archer, D. (2008). The identification of spelling variants in English and German historical texts: Manual or automatic? *Literary and Linguistic Computing*, 23(1): 65–72.

Pilz, T., Buck, C. & Luther, W. (2009). Working with nonstandard documents: A server-based trainable fuzzy search-plugin for Mozilla Firefox. In M. Mahlberg, V. González-Díaz & C. Smith, eds., *Proceedings of the Corpus Linguistics Conference: CL2009*.

Pittman, J.A. (2007). Handwriting recognition: Tablet PC text input. *Computer*, 40(9): 49–54.

Pollock, J.J. & Zamora, A. (1984). Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, 27(4): 358–368.

Pooley, N., Alcock, K., Cain, K., Hardie, A., Hoffmann, S. & Rayson, P. (2008). Variability in child language. In *Posters at ICAME 2008 Conference*, Ascona, Switzerland.

Potter, S. (1969). *Changing English*. The Language Library, André Deutsch, London.

Pravec, N.A. (2002). Survey of learner corpora. *ICAME Journal*, 26: 81–114.

Rabinovitch, S. (2007). Thousands of hyphens perish as English marches on. Reuters news article, `http://www.reuters.com/article/idUSHAR15384620070921`.

Rayson, P. (2008). From key words to key semantic domains. *International Journal of Corpus Linguistics*, 13(4): 519–549.

Rayson, P. (2009). Wmatrix: a web-based corpus processing environment.

Rayson, P. & Baron, A. (2011). Automatic error tagging of spelling mistakes in learner corpora. In F. Meunier, S. De Cock, G. Gilquin & M. Paquot, eds., *A Taste for Corpora. In honour of Sylviane Granger*, John Benjamins, Amsterdam.

Rayson, P., Archer, D., Piao, S.S.L. & McEnery, T. (2004). The UCREL Semantic Analysis System. In *Proceedings of the workshop on Beyond Named Entity Recognition Semantic labelling for NLP tasks in association with 4th International Conference on Language Resources and Evaluation (LREC 2004)*, 7–12, Lisbon, Portugal.

Rayson, P., Archer, D. & Smith, N. (2005). VARD versus Word: A comparison of the UCREL variant detector and modern spell checkers on English historical corpora. In *Proceedings of Corpus Linguistics 2005*, University of Birmingham, Birmingham, UK.

Rayson, P., Archer, D., Baron, A., Culpeper, J. & Smith, N. (2007). Tagging the Bard: Evaluating the accuracy of a modern POS tagger on Early Modern English corpora. In M. Davies, P. Rayson, S. Hunston & P. Danielsson, eds., *Proceedings of Corpus Linguistics 2007*, UCREL, Lancaster University, Lancaster, UK.

Reffle, U., Gotscharek, A., Ringlstetter, C. & Schulz, K. (2009). Successfully detecting and correcting false friends using channel profiles. *International Journal on Document Analysis and Recognition*, 12(3): 165–174.

Reynaert, M. (2008a). All, and only, the errors: more complete and consistent spelling and OCR-error correction evaluation. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis & D. Tapias, eds., *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, European Language Resources Association (ELRA), Marrakech, Morocco.

Reynaert, M. (2008b). Non-interactive OCR post-correction for giga-scale digitization projects. In A. Gelbukh, ed., *Proceedings of the Computational Linguistics and Intelligent Text Processing 9th International Conference, CICLing 2008*, vol. 4919 of *Lecture Notes in Computer Science*, 617–630, Springer, Berlin and Heidelberg.

Richardson, M. (1980). Henry V, the English Chancery, and Chancery English. *Speculum*, 55(4): 726–750.

Ringlstetter, C., Schulz, K.U. & Mihov, S. (2006). Orthographic errors in web pages: Toward cleaner web corpora. *Computational Linguistics*, 32(3): 295–340.

Ringlstetter, C., Hadersbeck, M., Schulz, K.U. & Mihov, S. (2007a). Text correction using domain dependent bigram models from web crawls. In *Proceedings of IJCAI-07 Workshop on Analysis for Noisy Unstructured Text Data (AND-07)*, 47–54, Hyderabad, India.

Ringlstetter, C., Reffle, U., Gotscharek, A. & Schulz, K. (2007b). Deriving symbol dependent edit weights for text correction: The use of error dictionaries. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, 639–643, IEEE Computer Society, Washington, DC, USA.

Rissanen, M. (1999). *The Cambridge History of the English Language: Volume III, 1476-1776*, chap. Syntax. Cambridge University Press, Cambridge.

Robertson, A.M. & Willett, P. (1992). Searching for historical word-forms in a database of 17th-century English text using spelling-correction methods. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, 256–265, ACM, New York, NY, USA.

Robertson, A.M. & Willett, P. (1993). A comparison of spelling-correction methods for the identification of word forms in historical text databases. *Literary and Linguistic Computing*, 8(3): 143–152.

Russell, R.C. (1918). United States patent 1261167.

Russell, R.C. (1922). United States patent 1435663.

Sampson, G. & Babarczy, A. (2003). A test of the leaf-ancestor metric for parse accuracy. *Journal of Natural Language Engineering*, 9(4): 365–380.

Samuels, M.L. (1963). Some applications of Middle English dialectology. *English Studies*, 44(1): 81–94.

Sánchez-Marco, C., Boleda, G., Fontana, J.M. & Domingo, J. (2010). Annotation and representation of a diachronic corpus of Spanish. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner & D. Tapias, eds., *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, European Language Resources Association (ELRA), Valletta, Malta.

Schaback, J. & Li, F. (2007). Multi-level feature extraction for spelling correction. In C. Knoblock, D. Lopresti, S. Roy & L.V. Subramaniam, eds., *Proceedings of IJCAI-07 Workshop on Analysis for Noisy Unstructured Text Data (AND-07)*, 79–86, Hyderabad, India.

Schmied, J. (1994). The Lampeter Corpus of Early Modern English Tracts. In M. Kytö, M. Rissanen & S. Wright, eds., *Corpora across the Centuries: Proceedings of the First International Colloquium on English Diachronic Corpora*, Rodopi, Amsterdam, St. Catherine's College, Cambridge.

Schneider, P. (2001). Computer assisted spelling normalization of 18th century English. In P. Peters, P. Collins & A. Smith, eds., *New Frontiers of Corpus Research: Papers from the 21st International Conference on English Language Research on Computerized Corpora, Sydney 2000*, vol. 36, 199–211, Rodopi, Amsterdam.

Scott, M. (2004). *WordSmith Tools version 4*. Oxford University Press.

Sebba, M. (2007). *Spelling and Society*. Cambridge University Press, Cambridge.

Shaklee, M. (1980). The rise of standard English. In T. Shopen & J.M. Williams, eds., *Standards and Dialects in English*, chap. 2, 33–62, Winthrop Publishers, Cambridge, MA, USA.

Shaw, P. (2008). Spelling, accent and identity in computer-mediated communication. *English Today*, 24(2): 42–49.

Shortis, T. (2007). Gr8 Txtpectations: The Creativity of Text Spelling. *English, Drama, Media*, 8: 21–26.

Simpson, J., Weiner, E. & Durkin, P. (2004). The Oxford English Dictionary today. *Transactions of the Philological Society*, 102(3): 335–381.

Sinclair, J. (1991). *Corpus, Concordance, Collocation*. Oxford University Press.

Singh, I. (2005). *The History of English*. Hodder Arnold, London.

Smith, N., McEnery, T. & Ivanic, R. (1998). Issues in transcribing a corpus of children's handwritten projects. *Literacy and Linguistic Computing*, 13(4): 217–225.

Sofkova Hashemi, S. (2003). *Automatic Detection of Grammar Errors in Primary School Children's Texts: A Finite State Approach*. Ph.D. thesis, Department of Linguistics, Göteborg University.

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15: 72–101.

Sproat, R., Black, A.W., Chen, S., Kumar, S., Ostendorf, M. & Richards, C. (2001). Normalization of non-standard words. *Computer Speech and Language*, 15(3): 287–333.

Strohmaier, C., Ringlstetter, C., Schulz, K.U. & Mihov, S. (2003a). A visual and interactive tool for optimizing lexical postcorrection of OCR results. In *Proceedings of the IEEE Workshop on Document Image Analysis and Recognition, DIAR'03*.

Strohmaier, C.M., Ringlstetter, C., Schulz, K.U. & Mihov, S. (2003b). Lexical postcorrection of OCR-Results: The web as a dynamic secondary dictionary? In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, vol. 2, 1133–1133, IEEE Computer Society, Washington, DC, USA.

Swan, M. (2005). *Practical English Usage*. Oxford University Press, 3rd edn.

Taavitsainen, I. & Pahta, P. (1997). Corpus of Early English medical writing 1375-1750. *ICAME Journal*, 21: 71–81.

Taavitsainen, I. & Pahta, P., eds. (2010). *Early Modern English Medical Texts: Corpus description and studies*. John Benjamins, Amsterdam.

Tagg, C., Baron, A. & Rayson, P. (2010). "I didn't spel that wrong did i. Oops": Analysis and standardisation of SMS spelling variation. In *ICAME 31 Abstracts*, 108–109, Gießen, Germany.

Taghva, K. & Stofsky, E. (2001). OCRSpell: an interactive spelling correction system for OCR errors in text. *International Journal on Document Analysis and Recognition*, 3: 125–137.

Tavosanis, M. (2007). A causal classification of orthography errors in web texts. In C. Knoblock, D. Lopresti, S. Roy & L.V. Subramaniam, eds., *Proceedings of IJCAI-07 Workshop on Analysis for Noisy Unstructured Text Data (AND-07)*, 99–106, Hyderabad, India.

Thompson, K. (1968). Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6): 419–422.

Thurlow, C. (2003). Generation Txt? The sociolinguistics of young people's text-messaging. *Discourse Analysis Online*, 1(1).

Tribble, C. (2000). Genres, keywords, teaching: towards a pedagogic account of the language of project proposals. In L. Burnard & T. McEnery, eds., *Rethinking language pedagogy from a corpus perspective: papers from the third international conference on teaching and language corpora*, Łódź Studies in Language, 75–90, Peter Lang, Hamburg.

Trudgill, P. (1999). *The Dialects of England*. Blackwell Publishing, 2nd edn.

Trudgill, P. & Chambers, J., eds. (1991). *Dialects of English: Studies in grammatical variation*. Longman Linguistics Library, Longman, London and New York.

Vallins, G.H. & Scragg, D.G. (1965). *Spelling*. André Deutsch, London.

Varnhagen, C., Mcfall, G., Pugh, N., Routledge, L., Sumida-Macdonald, H. & Kwong, T. (2009). "lol": new language and spelling in instant messaging. *Reading and Writing*, Online First.

Verberne, S. (2002). *Context-sensitive spell checking based on word trigram probabilities*. Master's thesis, Taal, Spraak & Informatica, University of Nijmegen.

Voutilainen, A. & Heikkilä, J. (1993). An English Constraint Grammar (ENGCG) a surface-syntactic parser of English. In U. Fries, G. Tottie & P. Schneider, eds., *Creating and Using English Language Corpora: Papers from the 14th International Conference on English Language Research on Computerized Corpora, Zürich, 1993*, 189–199, Rodopi, Amsterdam.

Wagner, R.A. & Fischer, M.J. (1974). The string-to-string correction problem. *Journal of the ACM*, 21(1): 168–173.

Wales, K. (2000). North and South: An English Linguistic divide. *English Today*, 16(1): 4–15.

Wilcox-O'Hearn, A., Hirst, G. & Budanitsky, A. (2008). Real-word spelling correction with trigrams: a reconsideration of the mays, damerau, and mercer model. In *CICLing'08: Proceedings of the 9th international conference on Computational linguistics and intelligent text processing*, 605–616, Springer-Verlag, Berlin, Heidelberg.

Wong, W., Liu, W. & Bennamoun, M. (2006). Integrated scoring for spelling error correction, abbreviation expansion and case restoration in dirty text. In C. Peter, P.J. Kennedy, J. Li, S.J. Simoff & G.J. Williams, eds., *Proceedings of the Fifth Australasian Data Mining Conference (AusDM2006)*, CRPIT, 83–89, ACS, Sydney, Australia.

Wong, W., Liu, W. & Bennamoun, M. (2008). Enhanced integrated scoring for cleaning dirty texts. *IJCAI Workshop on Analytics for Noisy Unstructured Text Data (AND), 2007*, 55–62.

Wu, S. & Manber, U. (1992). Fast text searching allowing errors. *Communications ACM*, 35(10): 83–91.

Yannakoudakis, E.J. & Fawthrop, D. (1983a). An intelligent spelling error corrector. *Information Processing & Management*, 19(2): 101 – 108.

Yannakoudakis, E.J. & Fawthrop, D. (1983b). The rules of spelling errors. *Information Processing and Management*, 19(2): 87–99.

Yule, G. (1944). *The Statistical Study of Literary Vocabulary*. Cambridge University Press, Cambridge.

Yvon, F. (2010). Rewriting the orthography of SMS messages. *Natural Language Engineering*, 16(02): 133–159.

Zamora, E.M., Pollock, J.J. & Zamora, A. (1981). The use of trigram analysis for spelling error detection. *Information Processing & Management*, 17(6): 305 – 316.

Zhu, C., Tang, J., Li, H., Ng, H.T. & Zhao, T. (2007). A unified tagging approach to text normalization. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 688–695, Association for Computational Linguistics, Prague, Czech Republic.

Zipf, G.K. (1932). *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press, Oxford, England.

Zobel, J. & Dart, P. (1995). Finding approximate matches in large lexicons. *Software: Practice and Experience*, 25(3): 331–345.

Zobel, J. & Dart, P. (1996). Phonetic string matching: Lessons from information retreival. In *Proceedings of the 18th ACM SIGIR International Conference on Research and Development in Information Retreival*, 166–173, Zurich, Switzerland.