Text-Induced Spelling Correction

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Tilburg,
op gezag van de rector magnificus,
prof. dr. F.A. van der Duyn Schouten,
in het openbaar te verdedigen ten overstaan
van een door het college voor promoties
aangewezen commissie
in de aula van de Universiteit
op vrijdag 2 december 2005 om 10.15 uur
door

Martin William Christian Reynaert

geboren op 2 april 1963
te Oostende, België

Promotores:	Prof. dr. W.P.M. Daelemans
	Prof. dr. H.C. Bunt
Copromotor:	Dr. A.J.M. van den Bosch

ISBN 90-9020100-9

Typeset in LaTeX
Printed by PrintPartners Ipskamp, Enschede
Cover Artwork: Kevin Reynaert

Acknowledgments

'You want to raise those values to a power.'
'Say again? A power... What power?'
'Don't know. Try something big. Power 10 might do.'
'What for? What are you thinking?'
'A power would drag the accumulated values apart.'
After-dinner talk between Caroline Vancoillie, my dear wife, and myself, september 2001.

Thus was TISC born ... Gestation was a prolonged affair with conception, as far as I can trace it, somewhere in late 1986 when I bought my very first computer during my year in Bangor, North-Wales. I had gone there to read Applied Linguistics for the degree of Master of Arts. The Licentiate in Translation I had just obtained in Antwerp, not being a full academic title, did not allow me to pursue a PhD, in Belgium. The M.A. would. This many years later I finally stand to obtain a PhD in the Netherlands. As Kurt Vonnegut would have said: 'So it goes...'

Since I have worked as a professional proofreader at the 'Gazet van Antwerpen', one of my first jobs, typos happen to trigger this special non-reading state of mind where all one does is see if all is as should be. Fact remains that trading in human proofreaders for computers at the end of the eighties was a bit premature, to say the least, as we shall show in these pages.

This dissertation would not have been written were it not for the kindness of Prof. Dr Walter Daelemans when he offered to be my PhD-promotor. With Walter, my copromotor and mainstay Dr Antal van den Bosch gave me my chance of a lifetime to pursue a PhD in the highly conducive and exhilarating setting which is the Induction of Linguistic Knowledge or ILK research team at Tilburg University. Thank you, Walter and Antal, for the trust, the many kindnesses and your guidance.

Further thanks go to Prof. Dr Harry Bunt for his willingness of being my other promotor. Our train conversations made travel lighter.

Thank you Harry, for this work benefits from your advice and your proofreading and editing experience.

I wish to thank the members of my reading committee for their efforts and astute and acute comments: Prof. Dr R. Harald Baayen, Prof. Dr Franciska de Jong, Dr Hans Paaijmans and Dr ir. Theo Vosse. Some of your comments will press me to continue searching for better answers than the ones given in these pages. A sincere thank you, all.

My post-doc colleague Dr Erwin Marsi I need to thank for his enthousiasm, warm smile and impromptu brain-storm sessions, week- or holiday. And also for setting aside his deepest principles in donning a tuxedo in order to be my paranymph. Thank you, Erwin.

Bertjan Busser deserves the same honourable mention for willing to be my other paranymph. Gratitude is further due for his expertise in keeping our computers fit and eager, for his good guidance in the deeper recesses of Linux and his patience with my initially often haphazard programming. Thank you Bertjan, may we stay room-mates for years to come.

Thanks are also due to the people at the Netherlands Organization for Scientific Research, better known locally as NWO, who set no age limit on joining a PhD-project. While I would now be the first to argue that one had better do this at a younger age, I am nevertheless truly grateful for having been allowed to join the VNC-NWO Prosody from Information in Text or PROSIT project, within which the current work was largely carried out. I remain indebted to Dutch society for providing this Belgian with these golden opportunities.

This work would also not have come about were it not that back there and then in Bangor, it turned out I liked computers. Even if producing hard-copy of the 100 page M.A. dissertation from the CP/M driven Amstrad machine took about 12 hours of screeching matrix-printing. Printing this PhD-dissertation is a matter of pain- and noiseless minutes these days. Preparing the present dissertation, however, cost far more time and far more processing power. Thanks go the good people at the Dutch Center for Mathematics and Computer Science or CWI for letting us have their to-be-replaced supercomputer, the 24-headed Medusa, on which most of the evaluations reported on in this work were run. It has been my privilege to be able to go off on a weekend in the secure knowledge that some twenty-odd processors were crunching away at providing the monday's thousands of evaluation results. Thank you Medusa, as well the various machines through which I spoke to you: pi0772, my trusted desktop at work, and Speedfox and Smartfox, at home.

Obtaining this PhD has further involved far more people than I can

possibly list and thank here. I could go as far back as my seventh year and thank my cub-scout Akela for urging us to hand in accounts of our sunday afternoon exploits in the dunes of Middelkerke, West-Flanders, Belgium. Which she handed back to us the next week with our errors marked. So, thank you, Chris.

In the here and now, thanks go to Iris Hendrickx for reading the manuscript and pointing out particularly impenetrable passages. No doubt some overly whirly and whorly turns of phrase remain. Maybe there is a drop of Celtic blood in me after all.

Thanks to Anne Adriaensen for her secretarial support and chats about sunny Greece. And to my former and present colleagues for their discussions, help and for making ILK and our section of the Faculty what it is: Toine Bogers, Sabine Buchholz, Sander Canisius, Hans van Dam, Federico Divina, Jeroen Geertzen, Yann Girard, Simon Keizer, Piroska Lendvai, Roser Morante, Reinhard Muskens, Mandy Schiffrin, Ko van der Sloot, Ielka van der Sluis, Elias Thijsse, Menno van Zaanen (for Tiger Rag, too, Menno!), Jorn Veenstra and Paul Vogt. Not to forget Rein Cozijn, Emiel Krahmer and Marc Swerts of our neighbouring section. Thank you, all.

Further thanks to the people in our companion research team, the Centre for Dutch Language and Speech or CNTS at the University of Antwerp, with which I was affiliated before joining ILK: Guy De Pauw, Bart Decadt, Véronique Hoste, Anja Höthker, Marie-Laure Reinberger and Eric Van Horenbeeck.

Erik Tjong Kim Sang and Jakub Zavrel I have known both at CNTS and ILK. Their joint Perl course in 2000 has helped direct my own course, for which I am truly grateful.

One of the great things of life in research is that we get researchers from abroad visiting: I thank Dr David Powers of Flint University, Australia, for an inspiring conversation in the early days of work on TISC. Also Dr Christer Johansson and Anders Nøklestad from Bergen University, Norway and Dr Gerhard van Huyssteen from North-West University, Potchefstroom Campus, South-Africa.

Thanks are especially due to all the people involved in the collection of the corpora I have been able to use. In particular to Dr Roeland Ordelmans for his continuing work on the Twente Corpus. Without corpora this work could not have been accomplished. In a way it answers the question raised by my supervisor in Bangor in 1987, Dr Carl James, when he asked: 'But how would you start tackling the lexicon?'. Thank you, James.

I must further thank Dr Robert C. Moore, Senior Researcher in the Natural Language Processing Group of Microsoft Research, for sharing

information about the spelling correction system he developed jointly with Dr Eric Brill and for thinking up a great way of providing interesting insights into spelling error data when the actual data cannot be shared.

Thanks, too, to Larry Wall for developing a post-modernist programming language in which this post-modernist actually manages to express his ideas and get things to work. Thanks, too, to the thousands of people who throughout the many years have developed, refined and redefined the UNIX and LINUX operating systems. Thank you, Linus Torvalds. I can only hope that one day the algorithm introduced in this dissertation somehow helps to pay off my debts to all these people.

My father-in-law, Jozef Vancoillie, I thank for the macro he wrote so that I could automatically evaluate the spelling-checker in Microsoft Office. And for the great many hours he put in extending my electronic library. Danke, Vake.

Special mention goes to the artist in the family, Kevin Reynaert, for graphically developing this idea of mine at short notice and turning it into a pleasing cover. Danke, Kevin.

To my mother, Agnes Coene, and to the memory of my father, Omer Reynaert, I dedicate this work. Primarily by means of example they taught us the value of hard work, deep love, good friendship and laughter. As my father used to say: 'If you cannot do it, it is high time you learned!'. He had at times a formidable voice. I wish I could hear it still. I can only hope to manage to teach my children as well. Danke, moemoe, vour al daje gieder vou me gedoan et!

I further dedicate this work to my dear wife and best companion, Caroline Vancoillie. Thank you, Caroline, for your love and patience, for bearing the brunt of child care and coping with me, besides your busy job farming all those eels and fishes. Danke, keppe! Thank you too for our darling daughters: Julie, Luca and Ymke. Meisjes, thank you for snuggling up to my chair to quietly sit and read or study the pictures during all these months. Thank you for the laughter, singing and dancing which cheers our lives. Danke, moksjes!.

Final thanks go to the rest of our families and friends for being sympathetic and patient. Now I can finally say: 'It is done. Be at the defense if you want to see me in a tuxedo!'. Lest this ends up as a full autobiography, I had better stop here. I hope I have not overlooked too many people that really should have been mentioned.

Martin Reynaert
Tilburg - October, 11 2005

Contents

1

Spelling error detection and correction: a brief survey

Spelling error detection and correction of electronic text has been a topic for research from the very earliest days that computers began to be used to produce text. Kukich (1992), in her survey article that spans the research work done in the first thirty years or so, calls it 'a perennial research challenge' (p. 377). The present work continues the search for better ways of using computers to obtain less noisy texts produced by means of computers.

In this chapter we give an overview of the main issues that arise in the field of spelling error detection and correction. We illustrate these issues on the basis of the prior art. We first introduce some terminology in Section 1.1. In section 1.2 we focus on the issue of what constitutes a language's vocabulary. We give an overview of the types of errors one may encounter and outline different approaches to the string correction problem in Section 1.3. In Section 1.4 we give a brief historical overview of research into spelling error detection and correction. We outline the, as yet incomplete, shift of focus from non-word errors on their own, in isolation, to errors within the context they appear in. We also discuss developments in the noisy channel based approach to spelling correction. We conclude the section with an investigation into the relationship between errors, their context and the order in which spelling error correction systems present corrections to the user. In Section 1.6 we describe the contributions made by this dissertation. We conclude the chapter with an overview of the rest of this dissertation.

1.1 An introduction to the field and its basic terminology

This dissertation is about language. The main building blocks of language are words. Words can be spoken and heard; they are not the subject of this dissertation. We focus solely on words that are written and can be read. More particularly we focus on the fact that things may go wrong when words are written down. This may result in character strings that do not in fact exist in the language. We call these **non-words**. Following Daelemans (1987) non-words might be loosely defined as 'unintentional deviations from some spelling convention'. Sometimes things go wrong when words are written down in such a way that the word written down is actually a word which exists in the language but is in fact out of place as viewed from the **context**: the words surrounding this unintended word. This kind of error is called a **real-word error**. We do not focus on real-word errors in this work, though we briefly discuss them further in Section 1.3.1, where we also give some examples. Apart from that, the present work focuses exclusively on non-word errors.

Writing used to be done mainly by hand, arguably most often using a pen and paper. When accidentally a word was produced which did not exist in the language, this was most often without long-term consequence. Since the advent of computers, most writing is arguably done by means of a keyboard, and the result is a text in electronic format, which may then be committed to paper, but which is also likely to be further maintained in electronic format. Eventually enormous amounts of text thus produced have and are being collected and further disseminated. All the little everyday accidents of producing text by means of a keyboard are thereby kept alive and archived like the texts themselves. It is on these little accidents, their accumulation in huge collections of electronic text or corpora and on possible ways of reducing their impact on the quality of these corpora that we focus in this work.

We briefly mention other ways of producing electronic text in Section 1.4.3. This is mainly because interesting ways of reducing the impact of the typical types of errors introduced by these other text production methods have been proposed and because we can use some of the techniques and heuristics employed there for our own purposes.

In order to be able to study the impact of the accumulation of 'unintentional deviations from the spelling convention' on the quality of a corpus, one first needs to get a lever on how one is to decide what constitutes a deviation and what constitutes the norm in a language. In the next section, we treat this first step. In a second step, one then

needs ways of restoring the deviations to the norm. This is the main topic of this dissertation to which we devote Chapters 3, 4, 5 and 6. Studying the accumulation of deviations is a subtopic to which we devote Chapter 2. Studying the actual impact of this accumulation we defer to later work.

In terms of a spelling checking system, deciding which of the words in the text are acceptable and which are unacceptable given a particular language constitutes the error **detection** phase. The **correction** phase is where the deviations are restored to the norm. The correction system often returns several correct words for a particular word which the detection system decided is unacceptable. The words returned we call the **correction candidates**, further abbreviated as: CCs. Most spelling error correction systems list the CCs they return in a particular order, with the intention of presenting what is probably the best CC first. This process is called **ranking**. We get **best-first ranking** when the CC at the top or beginning of the list, presented first to the user, is actually the word that resolves the typo given the particular context the typo was in.

1.2 The vocabulary

Who is to say what words exist in a language? One person's view on the matter may differ widely from another's. A word perfectly acceptable to one may be completely unacceptable to another. One way to settle the issue may be to try and find out whether the word under consideration is perhaps a one-off accident or is in fact shared by many within the language community, in which case it may be said to be part of a convention. We can use corpora to try and find out. Given a corpus of many millions of running words, some words will have been used many times, others will occur less often and some perhaps not at all. When we count how often words are used in a corpus, we in essence compile a frequency list. In relation to word frequency lists we talk of **word types** as contrasted to **word tokens**. When one derives a frequency list from a text, the tokens are counted and each distinct type with its frequency count is added to the list. The word types that make up the list define the vocabulary as observed in the corpus. The list then contains all the **word forms** observed. The word forms retain their inflections in a frequency list; they are not typically reduced to their **lemma**, the word form under which the various inflected forms would fall in a dictionary. The vocabulary contained in a corpus should not be taken to constitute the language's vocabulary: there may actually be many words in the language that happen not to be in the corpus at all,

however large this corpus is. The same goes for the vocabulary provided to a spelling error detection and correction system. Even though such a system's **dictionary** may be very large, it cannot be complete.

1.2.1 Zipf

George Kingsley Zipf was one of the first to apply the statistical method to language and look at the number of occurrences of words and what these frequencies tell us about how language works. There is a tremendous body of literature on Zipf's work. An excellent bibliography covering all related work dating back as far as the 1890s is maintained by Wentian Li[1]. Proponents of Zipf today are Ferrer i Cancho and Solé (2001), Ferrer i Cancho and Solé (2002), Ferrer i Cancho and Solé (2003), Ferrer i Cancho (2005a), Ferrer i Cancho et al. (2005) and Ferrer i Cancho (2005b). An accessible summary of these papers is to be in Solé (Forthcoming).[2] These works are important in that they appear largely to vindicate Zipf's findings, though corrections to Zipf's derivations are proposed. Zipf's own formulations of his findings in the preface to Zipf (1935) actually suffice for our purposes. On the probability of seeing a particular word a particular number of times in a sample of English, he wrote (p. xi-xii):

> In any extensive sample of connected English, it will, in all probability, be found that the most frequent word in the sample will occur on the average once in approximately every 10 words, the second most frequent word once in every 20 words, the third most frequent word once in every 30 words, the 100th most frequent word once in every 1000 words, the nth most frequent word once in every $10n$ words; in brief, the distribution of English approximates with remarkable precision an harmonic series.

This has become known as **Zipf's first law**, which is an empirical observation and not a law in a rigorous sense. In the definition of the National Institute of Standards and Technology or NIST, a United States federal technology agency, the law reads[3]:

> Definition: The probability of occurrence of words or other items starts high and tapers off. Thus, a few occur very often while many others occur rarely.
>
> Formal Definition: $Pn \sim 1/n^a$, where Pn is the frequency of occurrence of the nth ranked item and a is close to 1.

[1] http://www.nslij-genetics.org/wli/zipf/
[2] http://www.isrl.uiuc.edu/~amag/langev/paper/sole_scalingLaw.html
[3] http://www.nist.gov/dads/HTML/zipfslaw.html

In the formal definition, the nth ranked item is the item that comes on the nth place in the frequency list, where all items that share the same frequency, share the same rank. A **Zipfian distribution** is defined as a distribution of probabilities of occurrence that follows Zipf's law.[4]

On the number of different words to be found in a sample, we read (p. xii):

> [...] one finds in English (or Latin or Chinese) the following striking correlation. If the number of different words occurring once in a given sample is taken as x, the number of different words occurring twice, three times, four times, n times, in the same sample, is respectively $1/2^2$, $1/3^2$, $1/4^2$, [...], $1/n^2$ of x, up to, but not including, the few most frequently used words; that is, we find an unmistakable progression according to the inverse square, valid for well over 95% of all the different words used in the sample.

On the length of words (p. xi):

> [...] it can be shown that the length of a word, far from being a random matter, is closely related to the frequency of its usage - the greater the frequency, the shorter the word.

This has come to be known as the 'law of abbreviation'. Zipf apparently never proposed a mathematical function to derive frequency from word length. This was recently done by Sigurd et al. (2004).

An authoritative overview of all matters mathematical and statistical concerning word frequency distributions is given by Baayen (2001). The main subject of the book is the quest, within lexical statistics, for the formula that best predicts and fits the empirically observed data, given a particular corpus. Especially relevant in light of the present work is the fact that word frequency distributions are **Large Number of Rare Events** or LNRE **distributions**. LNRE distributions, due to Khmaladze (1987) (non vidi), are characterised by the presence of large numbers of words with very low probabilities. This entails that when sampling words the sample size has to be extremely large for the vocabulary size to stop increasing given an even larger sample. Also that the numbers of hapax legomena, dis legomena, etc. are non-negligible, statistically. Baayen (2001) reviews the various proposals for extensions or adjustments to Zipf's laws made throughout the years in light of the statistical properties of LNREs.

We refer time and again to Zipfian distributions throughout this work. In the following two subsections we take a closer look at what Zipf's work entails for the size of a language's vocabulary and for the

[4] http://www.nist.gov/dads/HTML/zipfian.html

distribution of words within texts and we describe what the implications of these entailments are for designing a spelling error detection and correction system.

1.2.2 The size of the vocabulary

The growth rate of the vocabulary, i.e. how many previously unseen words are seen given ever larger samples of text in a particular language, is one of the key issues addressed in lexical statistics. The growth rate can be estimated by the ratio of hapax legomena to the number of tokens (Baayen, 2001) (p. 50). The size of the vocabulary, i.e. the number of word types, is required by the definition of LNREs, due to Khmaladze (1987) (non vidi), to be infinite (Baayen, 2001) (p. 56).

Looking for applications and explanations of Zipf's law, Powers (1998) studies what happens to the acquisition of new vocabulary by taking successively double-sized samples from a corpus. He observes that words tend to enter the vocabulary faster than they tend to repeat, as is evident from the fact that the number of words of frequency 1 tends to increase as the size of the sample increases. He concludes that given that language is productive, and that an unbounded lexicon model has been indicated (or at least seems possible) in his experiments, this trend may well continue indefinitely, although it does seem to slow as the sample size is increased.

In another study of Zipf's Law, Kornai (2002) (p. 83–84) also concludes that there are an infinite number of words in English: 'This conclusion was arrived at not on the basis of productive morphological processes, but rather by inspecting the characteristic properties of large corpora, and deriving the open vocabulary result from these properties.' Nevertheless: 'results support the conclusion that the main grammatical source of infinite vocabulary growth is productive generative morphology, in particular compounding'. If anything at all should be regarded differently, he argues: 'it should be numerals, typos, eye-dialect [e.g. *Arrrrrrrnnnnnold*], direct quotations from other languages, and other arguably extragrammatical material that can be seen as contaminating the basic vocabulary pattern'.

Implications for spelling error detection and correction If a language's vocabulary is infinite, it follows that no dictionary representing the language can ever be complete. So, the best way we have of capturing as much of the language's vocabulary is by deriving the lexicon from as large a collection of text in the language as possible. The results obtained by Powers (1998) show that this too will never be enough, but that nevertheless given a sufficiently large corpus, the effect of incompleteness grows smaller. The effect should grow smaller

because words not seen given e.g. a one billion word corpus, are the more unlikely to be present in a far shorter text to be spelling checked.

It is as if we verbatim followed the 'recommendations' by Kornai (2002) in designing the system we propose. We reasoned we might want to build a spelling checking system for a language for which there are, or we have, no resources such as dictionaries available. Dictionaries are expensive. We refer the reader to McIlroy (1982) who relates all the design decisions, tribulations and trials he went through in developing the English dictionary for the first generation spelling error detector SPELL, which was also incorporated into the second generation spelling error corrector ISPELL. McIlroy (1982) notes: 'The word list for the UNIX spelling checker, SPELL, was developed from many sources over many years.' We do not have years to compile our own dictionaries for English and Dutch. By using corpus-derived lexicons we hope to overcome the limitations of traditional spelling checking systems' dictionaries, among other things the almost complete lack of names. To again quote McIlroy (1982): 'Where does one stop in accepting words? Dictionary makers try to cover everything but proper nouns in a broad range of fields. Proper nouns however abound in real text; a spelling checker that ignores them will be weak indeed'.

Further in line with Kornai (2002) we treat numerals differently. As we explain in Chapter 3, we do not disregard these but regard whatever string of digits encountered as a single, arbitrarily chosen, digit.

Typos we naturally treat differently, as they are the object of our attention, but also a source of noise we have to contend with in that they will be present in our lexicon. We cannot therefore 'trust' the lexicon and require ways of circumventing this noise.

Eye-dialect and direct quotations from other languages may or may not be integrated in our lexicons, as is explained in Chapter 3. In Chapter 4 we evaluate our approach for both English and Dutch. Direct quotations from other languages in effect gave rise to the work on multilingual spelling correction which we present in Chapter 6.

Implications of using corpus-derived lexicons In the above we have used the terms dictionary and lexicon to refer to the vocabulary available to a spelling error detection and correction system. This vocabulary is traditionally referred to as the system's dictionary. The word dictionary as we use it here complies well with sense 3c of the word as defined in Webster's Third New International Dictionary (1981): 'a vocabulary of accepted terms'. It is only in Cucerzan and Brill (2004), as far as we are aware, that this was first made explicit; a spelling correction system's dictionary had always been assumed 'trusted' before.

Using a corpus to derive the vocabulary no assumption of trust can be made. Noise in the corpus will find its way into the derived vocabulary. We therefore refer to the vocabulary available to our system as to its 'lexicon', in compliance with the second sense of the word in Webster's: 'the vocabulary of a language, of an individual speaker, of a set of documents, of a body of speech, of a subject, or of an occupational or other group'. In deriving the system's vocabulary from a corpus, from 'a set of documents', it can therefore be adapted to the purposes at will, but for reasons which are made clearer in Chapter 2, cannot be trusted in the same way as a dictionary.

Recent work by Comeau and Wilbur (2004) explores the possibility of identifying non-word spelling errors in a large corpus without making use of a trusted dictionary. The corpus is MEDLINE, a collection of more than 12 million references and abstracts covering recent life science literature. They argue convincingly that it would be unrealistic to try and maintain a dictionary with all properly spelled words, as the advance of research and the creativity of the researchers continually expands the vocabulary. So they use the corpus to try and detect misspelled words within it. This they do on the basis of a measure called *strength of context*, due to Kim and Wilbur (2001). The measure is a nonnegative real number the size of which reflects how strongly the word associates with the other words appearing in the documents in which it occurs. This score is typically low for misspellings as these occur less frequently than the correct variant and so do not have the same opportunity to build a consistent, reliable context. Though different from the cooccurrence counts we introduce in Chapter 3 and use throughout in our approach, the underlying ideas are very similar.

1.2.3 The bursty nature of words

In a study of the effects of lexical specialization on the growth curve of the vocabulary, Baayen (1996) (p.473) writes: 'this paper provides ample evidence that once a word has been used it is much more likely to be used again than the [random distribution of words] model predicts'. This is further elaborated in Baayen (2001), where Chapter 5 deals with the non-random nature of word usage and where adjusted LNRE models are proposed that can account for the locally concentrated, underdispersed use of key words.

Kleinberg (2002) studies this phenomenon over time in text collections such as his personal emails and research paper archives. A 'burst of activity' in the frequency of certain features signals the emergence of a new topic. Modelling this then allows for the detection and tracking of topics, e.g. particular news events.

Curran and Osborne (2002) explore whether a very large corpus (1.145 billion word tokens) can eliminate the sparseness problems associated with estimating unigram probabilities in empirical natural language processing. The **data sparseness problem** is described by Manning and Schütze (1999) (p. 198–199) as follows:

> While there are a limited number of frequent events in language, there is a seemingly never ending tail to the probability distribution of rarer and rarer events, and we can never collect enough data to get to the end of the tail.

This is in fact another paraphrase of Zipf's first Law as well as a commentary on it. The tail consists of **hapax legomena**: words that have been seen only once in the whole corpus and, however large the corpus: 'hapax legomena often constitute half of the types, but only a fraction of the tokens'. (Manning and Schütze, 1999) (p.199). Curran and Osborne (2002) study the convergence to the gold-standard unigram probability for words given ever larger subpartitions of the whole corpus, the gold-standard being the unigram probability as measured given the whole corpus. Most interesting, they find, is the convergence behaviour of 'rare but not necessarily unusual words, which is where using a large corpus should be most beneficial in terms of reducing sparseness' (p. 128). An example of such words is *tightness*, which appears 2,652 times in the corpus: this particular example failed 'spectacularly to converge' (p. 129), it being 'an extreme example of the case where a word is seen very rarely, until it suddenly becomes very popular' (p. 129). They conclude that this is due to **burstiness**: 'the fact that word occurrence is not independently and identically distributed' (p. 129). Proper names and topic-specific nouns and verbs exhibit the most bursty behaviour: newspaper articles are naturally clustered together as events occur over time.

Implications of word burstiness for spelling error detection and correction As words are bursty in nature, it follows that given a typo in its input document context, it is likely that the context may yield the correct form. This need not be the exact same word form, in Dutch e.g. it is likely the word recurs as part of a compound, or perhaps even more likely that the error occurs in a compound, containing as constituent part the correct word occurring elsewhere in the text in its free form. The latter is currently mere impression, based on our own observations: we cannot, to date, offer a quantification of this observation. Nevertheless, we incorporate in our spelling correction system a preprocessing phase that studies the input document and provides both detection and correction phases with information

about resembling words and their input document frequency.

This idea is by no means new, it was already employed to good effect by Morris and Cherry (1975). They used the document to be spell-checked itself to derive a ranked list to be presented to the writer. The ranking was based on 'string peculiarity', as derived by a specific measure from the document since the system had no other dictionary. The top of the list contained the 'outliers', in most cases: typos.

1.3 Errors in electronic text

Work on automatically identifying errors in electronic text goes back to the very first days when text was produced electronically. We cannot review all the early work here. A very comprehensive survey of the field prior the beginning of the nineteen-nineties is provided by Kukich (1992). We refer to some of the early work in what follows when it is relevant in that it introduced useful terminology or important techniques. Following linguistic convention we mark all typos by an initial asterisk (Crystal, 1985): *tiepographical. If this is contrasted to its correct form, the latter is presented in italics: *typographical*. The example we just gave was made up.[5] This is the only time we will use a fabricated typo. All other examples appearing in this work are **real-world errors**. These are errors which are attested: we found them either in electronic corpora or in printed material and can therefore show them in their original context. **Real-world** errors should not be confused with **real-word** errors, for which we provide some more examples in the next subsection.

1.3.1 Real-word errors

CANDIDATE FOR A PULLET SURPRISE
I have a spelling checker,
It came with my PC.
It plane lee marks four my revue
Miss steaks aye can knot sea.
– by Jerrold H. Zar

Mistyping a word may result in a word which does not exist in the language or one which does, according to convention. This gives rise to the distinction between what are termed non-word errors and real-word errors.

Real-word errors are most commonly termed **confusables** or **confused words**. They require different strategies for detection than

[5] However, at the time of writing (07-04-2005) it produced a 'Googlewhackblatt', which is the Google equivalent of a *hapax legomenon*: Google returned exactly 1 hit for this single word query.

pubic administration [1]	pubic hair [2]	pubic support [1]
pubic areas [1]	pubic hearing [2]	pubic use [1]
pubic debate [1]	pubic holidays [2]	slumping pubic [2]
pubic details [1]	pubic offering [1]	wants pubic [1]
pubic enquiry [1]	pubic outlays [1]	wide-ranging pubic [1]
pubic finances [2]	pubic releases [1]	
pubic funds [4]	pubic sector [4]	

TABLE 1.1 Selection of *public* confusable in Reuters RCV1. Between square brackets we present the word bigram's corpus frequencies.

non-word errors do and, in particular, cannot be resolved by the state-of-the-art spelling checker systems available today. Some types of confusable, usually a limited list of very commonly confused words, are handled by grammar checkers. A grammar checker uses a real-word's context to decide on whether or not the word or word form was used correctly, from the language's grammar point of view. In this it performs context-dependent real-word correction, which we discuss further in Subsection 1.4.1.

A real-word error may be one where a correct, but semantically implausible word was used instead of the more probable alternative word. In: 'I have seen it two' the last word has confusingly been substituted for 'too'. Real-word errors also often involve syntactic errors, where e.g. the verb does not agree with the subject of the sentence ('He drink too much'), or where e.g. a preposition is used instead of a verb: 'By my guest!'.

We present a selection of bigrams with a confusable for *public* for scrutiny and disambiguation by our readers in Table 1.1. The confusable had an overall corpus frequency count of 29 occurrences, only a fraction of which were not confused. This means the confused usage is actually more prevalent in this corpus than the regular usage, which probably occurs but rarely. This is certainly not to say that real-word errors are rare: according to Kukich (1992) they may account for fully 40% of the errors found in a corpus. Our examples were taken from the Reuters RCV1 Corpus, which we discuss in more depth in Chapter 2, where we also describe the other corpora used in this study. When we mention the corpus frequency for an example, we present it between square brackets.

1.3.2 Types of non-word errors

Kukich (1992) (p. 387) discusses three different types of non-word errors, largely on the basis of what caused the error. She states that often a distinction is made between **mistypings**: mechanical mishaps related

to the keyboard (e.g. *speel for *spell*), and **cognitive errors**: caused because the writer does not know how the word is to be spelled correctly (e.g. *sattelite for *satellite*). A subset of the latter category are **phonetic errors**, where the writer substitutes a phonetically correct but orthographically incorrect sequence of letters for the intended word (e.g. *parashooter for *parachuter*). As was already stated by Damerau (1964) (p. 171): 'it is usually not possible to determine the source of the error from the output'. It is nevertheless relevant to note that as can be seen from the example we just gave, phonetic errors tend to distort a word's spelling more than the other two categories usually do. For the sake of brevity, we here simply refer to all errors which result in a non-word as 'typos'. However, to avoid confusion between typos and word types, where applicable, we may also refer to them as **erroneous word forms** or simply **errors**.

In relation to spelling error detection and correction systems, another error typology is handled. This typology is commonly attributed to early work by Damerau (1964), but is attributed to Gates (1937) (non vidi) by Pollock and Zamora (1984). This typology distinguishes between 4 **edit operations**: string manipulation operations necessary to transform a typo into its correct form.

- **deletion**: 1 or more characters are missing: *abot for *about*
- **insertion**: 1 or more characters were added: *aboaut for *about*
- **substitution**: 1 or more characters were replaced by others: *obout for *about*
- **transposition**: 2 or more characters swapped places: *abuot for *about*

The error types are actually named from the point of view of the correct word form: one would need to insert the character *u* to turn *abot into the correct *about*, but the error is called a deletion error. In Damerau (1964) the correction system proposed actually handled only single character errors of the first three categories and 2 adjacent characters in the case of transpositions. These limitations no longer apply today, but we will refer to these more limited operations as **Damerau edits**, as they commonly are.

Ingels (1997) conducted a survey of all the error types present in two corpora. His definition of lexical errors includes not only the standard in-word errors but also between-word or segmentation errors. These are **split words**, a single word that was written as two or more tokens and **run-ons**, two or more words that were written as one token. Ingels provides a fine analysis of the reasons why most spelling correction systems available today cannot handle segmentation errors. As the

first reason, he states that they are less frequent than spelling errors in naturally occurring texts. But the most likely reason is the complexity inherent in the segmentation problem, where taking account of the context is essential. If one cannot assume that white space defines the word boundaries, the number of possible edits to consider rises exponentially in the length of the strings under consideration and the traditional approach to spelling error correction loses applicability.

One can use the number of edits necessary to transform one string into another to express the distance between and thereby compare the two strings, as we discuss in the next section.

1.3.3 Levenshtein distance

Given two strings, we would like to be able to say how much they resemble each other. **String distance** is a metric to express this resemblance. Wagner and Fischer (1974) named one of the most popular classes of algorithms for finding string distance the *minimum edit distance* algorithm: i.e. what is the *minimum* number of editing operations needed to transform string A into string B? The operations are: adding a character or *insertion*, removing a character or *deletion* and replacing a character by another or *substitution*. Note that *transposition* is not one of the operations. A cost or weighting factor can be assigned to each operation. The simplest form of this was proposed by Levenshtein (1965): each operation has a cost of 1. It is this cost we will use in our study. Throughout this work we will refer to it as the LD, short for **Levenshtein Distance**. Given the typo: *seroius and its correct form *serious*, we will say they are at an LD of 2 to each other: two editing operations are needed to transform the one into the other. A transposition therefore has a cost of 2: the way it is usually implemented transpositions are actually handled by a deletion at the cost of 1 and an insertion, at a further cost of 1.

The minimum edit distance is computed by *dynamic programming*, first introduced by Bellman (1957): a table driven method is applied to solve problems by combining solutions to subproblems. The table stores the results obtained by earlier calculations on part of the problem, for further use in solving the full problem. We refer to Jurafsky and Martin (2000) (pp. 155–156) for further explanation and exemplification of how the algorithm works. We use the minimum edit distance algorithm in conjunction with the algorithm we propose: they complement each other. Why this is and how it works is explained in Chapter 3. A further source rich in LD information, examples and implementations is the web

site by Michael Gilleland[6].

When two strings have an LD of 0, they are **exact matches** of one another. Approximate matches, which we discuss in the next section, have an LD larger than 0.

1.3.4 Approximate matching

A superb in-depth overview of the state of the art in approximate string matching can be found in Navarro (2001). **Approximate matches** are strings that are similar but not identical to the string one looks for. The overview focuses on approximate matching techniques for online searching in text. Online searching stands in contrast to offline searching, where the text can be preprocessed to build an index to it. An index can be seen as a frequency list built from the text which apart from the words and their frequencies further contains pointers to the position(s) in the text where each word occurs. The index speeds up search in the text, but building it has its own cost. Navarro (p. 33) specifically mentions that the field of indexed approximate string matching is quite immature and that the problem is very important because the texts in some applications are so large that no online algorithm can provide adequate performance. The core correction mechanism we propose in Chapter 3 in fact allows for index-based approximate search. We further note that Navarro states explicitly (p. 38) that 'although transpositions are of interest (especially in case of typing errors), there are few algorithms to deal with them'. Our core correction mechanism in fact excels at identifying transpositions.

An earlier paper by Zobel and Dart (1995) discusses various techniques to find approximate matches in large lexicons. For retrieval from document database systems using words, both personal name matching and spelling correction can be required on the same data. This may have various causes: the correct spelling may be unknown, the spelling of a given word may vary or there just might not be a single accepted spelling. The authors estimate that in one particular electronic text collection, containing the Commonwealth Acts of Australia, one in five of the distinct words are spelling errors. In Chapter 2 we assess the incidence of typos in the Reuters RCV1 corpus.

Zobel and Dart (1995) also give an overview of approximate string matching techniques. These break down into two families: **string similarity measures** and **phonetic methods**.

String similarity measures may be based on the edit distance we discussed above, or on n-gram similarities: an n-gram of a string s is any

[6] http://www.merriampark.com/ld.htm

substring of s of some fixed length n. A simple such measure is to choose n and count the number of n-grams two strings have in common. But *water* has as many n-grams in common with itself as with *waterline*. A more elaborate version which takes account of the length of the strings is due to Ukkonen (1992).

Phonetic methods produce a **similarity key** from a string. This key can then be compared to the other keys derived from e.g. the names in a database. The basic idea is to obtain the same key for words or names that sound alike. The oldest phonetic method, SOUNDEX, due to Odell and Russell (1918/1922), was patented as early as 1918. It uses codes based on the sound of each letter to translate a string into a canonical form. The first character is retained by the algorithm, all the rest are translated into a numerical code and then the whole is truncated to be at most four characters long. It was designed to bucket together names with a similar pronunciation. It was rather crude, so refinements have been proposed, the most popular of which is described by Knuth (1981). There is also the PHONIX variant (Gadd, 1990), which applies far more transformations to the string to be converted in a code. The details are of no further interest to us here. Both SOUNDEX and PHONIX are geared to English names, however. METAPHONE (Philips, 1990) and DOUBLE METAPHONE (Philips, 2000) are phonetic methods that better account for international names. The METAPHONE variants have been integrated in ASPELL[7] which in time will, according to some sources at least, replace ISPELL[8] as the default Open Source spelling checking and correction system, i.e. the one that will perform spelling correction on LINUX systems unless the user specifies which spelling checker he wants to use. ASPELL's author, Kevin Atkinson, states that his spelling checker merges the METAPHONE algorithm and ISPELL's near miss strategy which is inserting a space or hyphen, interchanging two adjacent letters, changing one letter, deleting a letter, or adding a letter[9]. This means that ISPELL is essentially an implementation of the Damerau edits. ASPELL combines the Damerau edits and a pronunciation model. Interestingly, the evaluations by Zobel and Dart (1995) show that phonetic techniques are inferior to string distance measures. As we will evaluate both ISPELL and ASPELL in Chapter 4, we will there see how this combination performs.

Real world application Grannis et al. (2004) compare the real world performance of approximate string comparators in the context of link-

[7] http://aspell.sourceforge.net/
[8] http://lever.cs.ucla.edu/fmg-members/geoff/ispell.html
[9] http://aspell.net/metaphone/

ing databases containing patient records from two hospitals. Due to the disparity of database systems used, there is often the need to find a match between records on the basis of e.g. the patient's name and a few more data fields such as date of birth, social security number, street name, etc. Given the variability inherent to names this often requires applying approximate string comparators. They discuss how yet another phonetic similarity method they compare with, the New York State Identification and Intelligence System algorithm, NYSIIS, is used for what in database record linkage studies is known as **blocking**. This is effectively a technique to reduce the search. As is explained in Baxter et al. (2003): when databases need to be linked, potentially every record from the one database needs to be compared to every record in the other. The number of record comparisons then grows quadratically with the number of records to be matched and this quickly becomes computationally infeasible for large data sets. Traditional record linkage systems therefore use a record attribute to split the database into blocks. An example of a record attribute would be the first four characters of the surname: all records whose surname shares the first four would then constitute one block. Only for the records within a particular block will the finer exact and approximate matching strategies then be applied. This is in line with Zobel and Dart (1995), who split up the task of finding approximate matches in large lexicons into two parts: first apply cheaper techniques to reduce the search space, then apply finer string similarity measures to refine the search. In a way, that is precisely what we propose in Chapter 3, though with a non-phonetic similarity key.

We have now introduced basic terminology, discussed how strings can be compared and how comparable strings can be identified in text. In the next section we look at the problems of detecting and correcting typos in isolation or within a context.

1.4 Brief historical overview

Research on spelling correction has over the past 50 years spawned a respectable body of literature. We cannot review it all here, so we limit ourselves to referring the reader to major papers for those topics we consider peripheral to the research conducted here. Prior work that is related to what we set out to present here, we discuss briefly.

1.4.1 Isolated word correction versus context-dependent correction

Kukich (1992) defines three increasingly broader and harder problems (as described by Jurafsky and Martin (2000) (p. 143–144):

- non-word error detection: detecting spelling errors that result in non-words (like *graffe for *giraffe*)
- isolated-word error correction: correcting spelling errors that result in non-words, for example correcting *graffe to *giraffe*, but looking only at the word in isolation
- context-dependent error detection and correction: using the context to help detect and correct spelling errors even if they accidentally result in an actual word (real-word errors). This can happen from typographical errors (insertion, deletion, transposition) which accidently produce a real word (e.g., *there* for *three*), or because the writer substituted the wrong spelling of a homophone or near-homophone (e.g., *dessert* for *desert*, or *piece* for *peace*).

We quote verbatim from Jurafsky and Martin (2000) in order to show that somehow something seems to have slipped from the attention of researchers. Indeed, in the third category we might expect to see the example from the first two repeated but expanded with some context, e.g. 'Due to its long neck the *graffe is the tallest land animal' or 'A *graffe is when a politician tells the truth'[10]. It appears the focus in spelling correction research shifted to the third category at about the beginning of the nineteen-nineties, but largely to the exclusion of non-word errors. It seems the consensus was pretty much that non-word error correction was 'solved' and it was time to tackle the harder problem of detecting and correcting real-word errors. And so we witness what we think was a rather unfortunate move, that 'context-dependent' error correction became to be associated with solving the real-word problem, exclusively (Mays et al. (1991), Golding (1995), Golding and Schabes (1996), Golding and Roth (1999)). In Carlson et al. (2001) this is even extended to "context-sensitive *text* correction", which then covers anything to do with errors involving 'improper use of real words'. Clearly, progress has and is being made in solving real-word errors. But what about the non-words, why is it the state-of-the-art spelling checkers still work at the isolated-word level? So we see that state-of-the-art research into non-word correction as reported by Brill and Moore (2000) still regards adding a language model to the system as something of an optional add-on, to be tried out, not to be taken as an essential and integral part of a spelling correction system.

Note that the shift in research focus was not due to Kukich, who stated explicitly that:

[10] I.e.: 'A gaffe is when a politician tells the truth.' attributed to Michael Kinsley (b. 1951), U.S. journalist. Guardian (London, Jan. 14, 1992). Source: http://www.bartleby.com/66/73/32773.html.

> A context-based correction technique would not only address the problem of real-word errors, i.e. errors that result in another valid word, but it would also be helpful in correcting those non-word errors that have more than one potential correction. [...] Developing context-based correction techniques has become the foremost challenge for automatic word recognition and error correction in text. (Kukich, 1992) (p.379)

A notable exception to this development within the English research community is the work by Vosse (1992) and Vosse (1994) who developed a grammar-based spelling error correction system for Dutch. The batch-oriented system, called CORR^{ie} was developed to detect and correct morpho-syntactic errors in Dutch texts, i.e. errors within context. It includes a spelling corrector based on trigram and triphone analysis (van Berkel and de Smedt, 1988). A trigram analysis determines which combinations of three characters occur with what frequency in the words in the language. To select correction candidates the less frequent combinations are actually used. A triphone analysis looks at the phonemic transcription of words' syllables. A set of rules then allows for the identification of homophonous words, so this works for errors which result on completely homophonous spellings. Vosse's system further includes a shift-reduce parser based on Tomita (1986) which interacts with the spelling corrector and handles certain types of structural errors. The system therefore deals with both non-word and real-word errors. Both modules have been integrated with a compound analyzer and a dictionary of 275,000 word forms into a program for stand-alone proof-reading of Dutch texts. This work had major impact in that it was followed by a European Union sponsored project[11] called SCARRIE with the aim of developing a high-quality proofreading tool for the Scandinavian publishing industry as a whole. In this project researchers, companies and publishing houses from Denmark, Norway and Sweden participated.

The system we propose is context-sensitive but focuses exclusively on non-word errors. It is context-sensitive in that it has a language model, but this language model is not probabilistic in the Bayesian sense. It is also context-sensitive in that it ranks its CCs according to evidence gathered from the immediate input text context for the typo it corrects.

Huang and Powers (2001) remark

> 'Note however, that all spelling correction is context-sensitive - the difference with confused words is that the identification of spelling errors is also context-sensitive.

[11] http://www.hltcentral.org/projects/detail.php?acronym=SCARRIE

This indeed highlights a major difference: for detecting a real-word error you cannot rely on the technique commonly employed to detect non-word errors: see if it exists in the dictionary or not. We would like to add that to accurately correct a non-word error, you cannot do without context either. A spelling correction system often returns more than one correct word for a typo. In the absence of context, all you can do, for those errors for which more than one correction candidate is returned, is list them and, at best, list them in the order of which is more probable than the other. What we propose to do is to list them in the order that is most likely, given the particular context they appeared in.

As the system we propose does not have a trusted dictionary, identification of the spelling errors is also a context-sensitive affair. Unfortunately, this does not entail that the strategy we currently employ to detect the typos also allows for detecting real-word errors. Though we have great hopes of extending our system with strategies that do allow for the detection and correction of real-word errors, that is beyond the scope of this dissertation.

1.4.2 The noisy channel modelling approach to spelling correction

Kernighan et al. (1990) were first to suggest the noisy channel approach to spelling correction. The basic assumption of this approach is that the typist knows what words he wants to type, but some noise is added on the way to the keyboard, in the form of typos and spelling errors. It is further assumed that using classic Bayesian inference the intended correction can often be recovered from the typo by finding the correction that maximises the prior probability of a particular CC in light of the likelihood that a character x was mistyped as y. The prior probability for the CC is obtained by counting how often it occurs in a corpus and normalizing this count, i.e. dividing the count by the total count of all word tokens. The resulting probability then falls between 0 and 1. The likelihood that x is mistyped as y is estimated from a corpus of typo/correct word pairs. In Kernighan et al. (1990) this was based on the simplifying assumption that the correct word differs from the misspelling by only a single insertion, deletion, substitution or transposition. Nevertheless, the system, called CORRECT, was capable of correcting the greater part of typos in text, given that most typos fall within that range, as we will show in Section 2.2.5. CORRECT first generates a list of CCs from the typo by applying all possible single character transformations and repeatedly seeing if this results in a word present in its dictionary. So for the non-word *acress the following list of CCs

is retrieved: actress, cress, caress, access, across, acres. This was not new, in fact, it was what the UNIX spelling checker SPELL did. The new contribution was that, in a second step, the CCs were *ranked*: offered to the user in the order of which CC is more likely to have been intended by the typist when (s)he mistyped the word. This ranking is done on a probabilistic basis. Using 4 confusion matrices, square 26×26 tables which represent the number of times one of the 26 alphabetical characters was incorrectly deleted or inserted after another or substituted for the other or transposed with the other, the probability for e.g. *across* to have been intended when *acress was actually typed, is calculated. The actual counts in the matrixes are derived from the large corpus of errors, either manually or by iteratively using the spelling correction algorithm itself, which results in the confusion matrixes being learned. This model in fact predicts that *acres* was the intended word, a fact contradicted by the original context, which clearly points to *actress* having been the intended word: '... was called a "stellar and versatile *acress whose combination of sass and glamour has defined her ..." '. However, in 87% of the cases where the error had 2 possible corrections, CORRECT agreed with the majority consensus of three human judges who had access to the contexts.

Brill and Moore (2000) and its successor (Toutanova and Moore, 2002), represent the state-of-the-art in spelling correction as proposed in the literature. Brill and Moore (2000) extend the noisy channel model to be able to handle an arbitrary number of edits. In particular they improve the channel model for spelling correction, by learning generic string to string edits, along with the probabilities for these edits. The system is trained to correct generic single word spelling errors and is conditioned on the position in the string that the edit occurs in. While people rarely, they assume, mistype *antler* as *entler, *reluctant* is often mistyped as *reluctent. So this error is modelled as P(ent—ant), i.e. the **probability** that *ent* will have been typed rather than *ant*, rather than simply P(e—a) as was done in the earlier models. Also, account is taken of the positional information, the place in the word where this kind of error is most likely to occur. This model is further equipped with a probabilistic word trigram language model, which helps performance. Toutanova and Moore (2002) slightly improve on the results of Brill and Moore (2000) obtained without the language model, by further adding a pronunciation model to the system.

Language-specific training data What all noisy channel models have in common is that training data is required to derive the likelihoods that character(s) x will be replaced by y. Kernighan et al. (1990)

built confusion matrices for English on the basis of one year's Associated Press newswire text. Brill and Moore (2000) used 8,000 common English spelling errors to train their model. Training data for a noisy channel model is necessarily language-specific and so is the derived spelling correction system. In the next subsection we take a closer look at the matter of ranking the CCs.

1.4.3 Context and ranking

Context-less ranking We think it is only by virtue of adding a language model and by taking the error's context into account that the misprediction of e.g. *acres* instead of *actress* for *acress can possibly be overcome. Having an improved isolated-word error model for noisy channel spelling correction by itself will not remedy this. Not taking into account the context beyond the misspelled word will always result in the same CC to be ranked first.

Let us consider the following laboratory sentence:

> Her vehement *onjections to these painful *onjections were based on solid medical evidence, as well as a hearty dislike of needles.

Whether these errors should be resolved to either *objections* or *injections* is fully determined by their immediate context. It seems to us nothing else can be called upon: both are keyboard adjacency cases: both *b* and *n* and *i* and *o* are adjacent on the keyboard. In a noisy channel system the likelihood of *ob* versus *in* to be rendered as *on* would determine the ranking, in combination with the prior probability derived from a corpus. The Reuters corpus frequency of the words is injections [346] and objections [1357]. This might guide the ranking of proposed candidates, but would be very much dependent on the background corpus used. Adding a language model would nevertheless help best: a bigram language model would likely tell that *vehement objections* is a far more probable combination than *vehement injections* while *painful injections* is far more probable than *painful objections*.

Context-guided ranking An accessible study which provides a sound overview of the issues involved in generating and ranking spelling error corrections is Tillenius (1996). He evaluates the generation and ranking of spelling error corrections for Swedish. The corrections were indicated by the original context the errors appeared in, culled from newspaper articles and student essays. The error list contained 729 items. For ranking the correction candidates he explores the use of modified edit distance (where the cost per edit is dependent on the kind of edit required), word frequency and word bigram frequencies. The word bigrams did not perform well. He concludes that for spelling

correction of texts on any subject, a much larger list than his 200,000 word pair list seems to be necessary. Combining edit distance and word frequencies worked best, on the isolated word level. He states that in order to attain further improvement, context information is probably needed. The best-first ranking score attained was 77%, when counting only the typos for which the correct word appeared in the dictionary.

Agirre et al. (1998) study what kind of knowledge facilitates best-first ranking in non-word context-sensitive correction (they boldly reclaim the term) for general unrestricted texts. The kinds of knowledge they consider are: syntagmatic knowledge in the form of part-of-speech (POS) tags and paradigmatic knowledge represented as a measurement of 'affinity distance' between nouns in Wordnet. Further, they employ general and document word-occurrence frequency rates. The various knowledge sources are used to rank the CCs returned by ISPELL. They fine-tune their systems and combinations thereof on fabricated evaluation sets and finally test on 158 real-world typos derived from magazine text in the Bank of English Corpus[12]. The correct forms for these were indicated by the context. They employed a heuristic that eliminated all CCs that were at an LD of 2 or more. Another heuristic was to not include words shorter than 4 characters long. Use of both the paradigmatic knowledge and the general frequencies (derived from the Brown Corpus) were found not to be useful. The syntagmatic knowledge and document frequencies helped, although the latter very little as the documents were very short. They generally conclude results improve as more context is used.

The use of context to resolve OCR errors The types of errors that may be produced by typing are different from the types that are produced when previously printed or even handwritten text is reproduced in electronic format by Optical Character Recognition (OCR) devices. These first produce a digital image of the text and then OCR-software determines which characters are most likely to constitute the image. Depending on the quality of the printed text and on the quality of the software this process has smaller or greater chances of successful recognition.

Taghva and Stofsky (2001) discuss the typical problems associated with correcting OCR-induced errors. They state that 'traditional spelling correction is performed by isolating a word boundary, checking the word against a collection of commonly mispelled words, and performing a simple four-step procedure: insertion, deletion, substitution, and transposition of all the characters'. In OCR text, word isolation

[12] http://titania.cobuild.collins.co.uk/boe-info.html

is much more difficult because errors can include the substitution and insertion of numbers, punctuation and other non-alphabetic characters. Second, the mapping of characters may not be one-to-one: an *m*, for instance, is often rendered as *iii*. Third, words may be broken, e.g. *program* being recognised as *pr˜ gram* (OCR-software commonly returns a tilde for characters it is insufficiently sure of). Last, OCR errors may vary from device to device, document to document and font to font. In the paper, they introduce a dynamic confusion construction mechanism that learns this variation. Taghva et al. (2004) gives an overview of 15 years of research in the field, especially of OCR errors in relation to Information Extraction.

This work is relevant here in that, in contrast to most research in non-word errors resulting from typed input, their systems study the input context and derive useful information from it, in order to facilitate the correction. This is a strategy we also employ in the system we propose. Their systems further need to cope with word segmentation errors, a feature notably absent from most spelling checkers today, as we discussed above, but also in part implemented in the system we propose. Finally, we think it should be relatively easy to extend the system we propose to also handle OCR-induced errors. That was not attempted within the work reported on in the present study, however, largely for lack of suitable evaluation material.

1.5 Introduction to Text-Induced Spelling Correction

We call the spelling detection and correction system we propose in this dissertation **Text-Induced Spelling Correction** or, in short: TISC.

TISC should be pronounced as 'tisk'. This is a verb, denoting the sound a human proofreader may make when he, again, finds a particularly recurrent typo. It is a verb that has not found its way into either the Oxford English Dictionary or Merriam-Webster's, which is a striking reminder no dictionary will ever be complete.

Why 'Text'? We might have used the word 'corpus' instead of 'text', but that would have deprived us of the connotation with human proofreaders. Also, a corpus may come in highly marked-up form, depending on what it is intended for. We use raw text only, the kind of text one could produce using an old-fashioned type-writer in the days when no scores of fancy fonts or layouts were at our finger-tips' disposal.

Why 'Induced'? This actually comprises both uses of the verb *induce*:

- lead or cause (sb. to do sth.)
- bring about

In the first reading, TISC is induced to rank a particular CC first on the basis of the input text, i.e. the immediate as well as the wider context of the typo.

In the second reading, TISC is induced from text, i.e. the language knowledge incorporated in the system has fully been derived from a large body of text in that language. On the basis of this knowledge decisions are taken as regards to validating a particular word form or sending it on to be corrected, which again is achieved on the basis of that knowledge.

1.6 Contribution of this dissertation

In this section we list the contributions of this dissertation.

The main contribution we make in the present work is that we propose a new approximate string matching algorithm and use it as the basis for a context-sensitive spelling error detection and correction system for non-word errors that is, in unsupervised ways, fully derived from large corpora of text in a particular language. Context-sensitivity is achieved largely through the incorporation of word bigrams in the Text-Induced Spelling Correction system's lexicon.

On the basis of extensive evaluations, we show that the core correction algorithm we propose, which is based on a similarity key derived from the actual characters that constitute a word, is sufficient to resolve the errors one encounters in edited text. We further show that the similarity key based correction algorithm we propose requires no training and is thereby largely language-independent. We illustrate this on the basis of evaluations for both English and Dutch, which allow us to show that it is feasible on this basis to perform spelling error detection and correction to levels surpassing those of the state-of-the-art systems available today. We finally show that the algorithm may perform equally well on mixed language tasks, performing correction to the same levels as in the monolingual task when the text to be spelling corrected is mixed English-Dutch text and the lexicon is a mixed English, Dutch and French lexicon.

1.7 Overview of this dissertation

In Chapter 2, we discuss the corpora we used for this study and perform an in-depth analysis of a large corpus of non-word errors derived from the Reuters RCV1 corpus.

In Chapter 3 we present our spelling detection and correction system, which we have called 'Text-Induced Spelling Correction' or TISC.

In Chapter 4 we conduct in-depth evaluations of TISC and present

the results on a variety of tasks. We evaluate TISC and state-of-the-art systems available today on both English and Dutch and compare their performances.

In Chapter 5 we evaluate the evaluations and conduct a survey of the various metrics one may use for evaluating a spelling detection and/or correction system. We compare with state-of-the-art-systems proposed in the literature.

Chapter 6 is devoted to multilingual spelling error detection and correction and the issues involved: we determine whether prior language detection is a necessary prerequisite.

Chapter 7 presents our conclusions and suggests future directions.

1.8 Summary

In this first chapter we have provided an introduction into the field of spelling error detection and correction. We have provided some necessary terminology, after which we have discussed what distinguishes an erroneous word variant from a conventionally accepted form. We have shown that the work by Zipf provides insight into what constitutes a language's vocabulary and into the distribution of words within the vocabulary. We have from these insights derived implications for the design of a spelling error detection and correction system. Next we have given an overview of approximate string matching techniques and how they relate to spelling error detection and correction. We have briefly positioned our own core correction mechanism based on a non-phonetic similarity key within this field. We have then surveyed some of the history of spelling error detection and correction research. We have concluded that whereas research into real-word error detection on the basis of context has drawn a lot of attention, research into non-word spelling correction has largely stuck on the isolated word level. We have then discussed the noisy channel modelling approach to spelling error correction and pointed out that this is necessarily a language-specific technique. We have discussed the relationship between ranking the correction candidates and taking account of the typo's context. We have then briefly introduced the name for the system we propose: Text-Induced Spelling Correction. We have finally discussed the main contributions of this work.

Before we can move on to the full description of TISC in Chapter 3, we first describe the corpora used in this research and perform an in-depth analysis of the extent of non-words in a large, English, contemporary corpus in Chapter 2.

2

Overview of corpora used and detailed error study

The work presented here uses as primary resources large collections of text available in electronic format. In this chapter we first give an overview of the various corpora we used. Next we study the extent of typos occurring in a large contemporary English corpus. Apart from this corpus, which we set aside for evaluation purposes, all the other English corpora were used for development purposes. For Dutch, we followed the same methodology.

2.1 The corpora

2.1.1 Corpora for development purposes

The following corpora were used to extract the lexicons and other information for our spelling checking and correction system. We used primarily English and Dutch corpora for building a spelling correction system for both languages. In Chapter 6 we present a multilingual version, which besides Dutch and English, also has French in its dictionary.

English

The American English corpus we used was the New York Times (1994-2002) material available in the LDC Gigaword Corpus (Graff, 2003). We further refer to this corpus as NYT.

For British English we used the British National Corpus, a balanced corpus of contemporary written and spoken British English (Leech, 1992)[1]. We refer to this corpus as BNC.

[1] http://www.natcorp.ox.ac.uk/index.html

Dutch

For Dutch we used both the ILK Corpus[2] and the Twente Corpus[3] (TWC). The ILK Corpus is a collection of southern Dutch regional newspapers, which we supplemented with an assortment of about 100 books, mainly novels, and 5 years of Dutch Roularta magazines. The Roularta magazines constitute a series of Belgian Dutch weekly magazines devoted to current affairs, industry, financial affairs and leisure. The TWC comprises a number of national Dutch newspapers, teletext subtitling and autocues of broadcast news shows and news data downloaded from the WWW. We also used the Twente Update, TWC2, which represents national Dutch newspapers from the year 2002. We reserved the Update for special purposes, as we explain in Chapter 4.

French

For French we used 8 years ('91-'98) of Roularta Magazines[4]. These constitute a series of Belgian French weekly magazines devoted to current affairs, industry, financial affairs and leisure.

2.1.2 Corpora for evaluation purposes

English

All evaluation materials for English were derived from the Reuters RCV1 (Lewis et al., 2004), formally known as RCV1-V1 (Release date 03-11-2000, Format version 1, correction level 0). This corpus covers the period 20-08-1996 to 19-08-1997 and contains about 810,000 Reuters, English Language News stories. The website[5] devoted to the corpus provides a breakdown in number of words and paragraphs per article, among other statistics. Section 2.2 deals with the error list we obtained from this corpus. In Chapter 4 we discuss how we built a benchmark test set of nearly 3,000 typos within their original context. We provide more detailed information about the benchmark set there.

Dutch

For Dutch we collected newspaper articles containing typos from the daily free tabloid *Metro*[6]. The material and manner of collecting is described in Chapter 4.

[2] http://ilk.uvt.nl/ilkcorpus/
[3] http://wwwhome.cs.utwente.nl/~druid/TwNC/TwNC-main.html
[4] http://www.roularta.be/en/products/
[5] http://about.reuters.com/researchandstandards/corpus/statistics/
[6] http://www.metropoint.com/

Corpus	Lang.	Mb	Tokens	Types	Word bigr.
NYT	AE	5,570	1,106,376,695	1,863,802	41,611,447
BNC	BE	567	113,165,579	651,321	12,141,343
R-RCV1	IE	714	134,031,130	1,626,038	12,555,943
ILK	D	1,748	314,051,047	2,747,341	32,767,046
TWC	D	2,014	365,545,491	2,607,305	33,912,967
TWC2	D	510	92,793,519	914,026	10,946,486
ROUL	F	273	52,722,253	422,682	5,452,266

TABLE 2.1 Corpora Statistics: Corpus, language (AE: American English, BE: British English, IE: International English, D: Dutch, F: French), size in Megabytes, number of word tokens, number of word types, number of word bigrams.

2.1.3 Corpora preprocessing

All corpora were preprocessed in the same manner. All XML or other tags were discarded. Each corpus was then tokenized by applying a rule-based tokenizer[7]. A tokenizer divides text in tokens and sentences. A token may be a word, a number or a punctuation sign. The tokenizer inserts a space to set apart punctuation signs that are attached to words and marks sentence boundaries explicitly. We further normalized each corpus by replacing all word-external punctuation by a single unique mark, a period, which then signifies that there had been some form of punctuation at that point. All strings containing only digits were reduced to a single arbitrarily chosen digit, *3*, which then signifies there had been some number at that point. *N*-gram frequency lists were derived by means of the CMU Statistical Toolkit (Clarkson and Rosenfeld, 1997).

2.1.4 Corpora statistics

Statistics regarding corpora sizes and numbers of words in these corpora are presented in Table 2.1. The discrepancy in number of types between the BNC and RCV1, which are comparable in byte and token size, can be explained by the fact that the latter did not get the digits and punctuation normalization preprocessing, as we reserved the RCV1 for evaluation purposes and did not use it for lexicon development.

[7] Developed at ILK, University of Tilburg, by Dr. Sabine Buchholz, to whom we are indebted for its use

2.2 Reuters RCV1: corpus and typos

In this section we take a close look at one particular English corpus and determine the extent of non-word errors in it. Reuters is the world's largest international text and television news agency, with over 2,000 journalists, photographers and camera operators in 190 offices, serving 151 countries (Rose et al., 2002). We chose the Reuters Corpus Volume I (Lewis et al., 2004), formally known as RCV1-V1 and further referred to as RCV1, to study the extent of erroneous word forms in a contemporary real world corpus.

2.2.1 Preliminaries

Damerau (1964) (p. 171) wrote:

> An inspection of those items rejected [by the retrieval system's index storage program] because of spelling errors showed that over 80 percent fell into one of four classes of single error - one letter was wrong, or one letter was missing, or an extra letter had been inserted, or two adjacent characters had been transposed.

Peterson (1986), on the basis of two error lists derived from keyboarded word lists containing 155 and 360 errors respectively, revises the 'over 80%' upwards to 92.9% for the shorter and 94.7% for the longer list. Pollock and Zamora (1984) confirm a figure between 90 and 95% on the basis of 50,000 single-error misspellings culled semi-automatically from 250 million words of text from scientific and scholarly databases. Their study is without doubt the most comprehensive study of typos in real-world databases ever published. The study formed the basis for the development of the SPEEDCOP correction algorithm, based primarily on the transition probabilities derived from the statistics gathered. From the statistics two, complementary, similarity keys were derived. In that the system is extremely language-dependent and handles only the Damerau edits, we do not discuss it in more detail. A fine summary is provided in Kukich (1992).

Damerau and Mays (1989) present some statistics on the error types observed in three different corpora. We reproduce these in Table 2.2, summed, disregarding their original three different sources. The authors caution for the statistics of the category 'Word division': for preprocessing the data they used various programs which might have treated hyphenation and line ends differently. They further explain that it is problematical to know whether some words, e.g. frozen expressions such as *abovementioned*, should be 'run together, hyphenated or left as individual words'. Note that these statistics represent a list of 502 misspellings.

Category	Original description	Number	Percentage
Split word	Word division	180	35.86%
Deletion	Missing letter	115	22.91%
Substitution	Wrong letter	63	12.55%
Insertion	Additional letter	56	11.16%
Transposition	Transposition	51	10.16%
Multiple error	Multiple error	37	7.37%

TABLE 2.2 Error-type statistics due to Damerau and Mays (1989). Shown are the names of the error types as handled in this work, the original paper's descriptions, number of errors found and their percentage of the total amount.

Kukich (1992) (pag. 389) remarks that data concerning the frequency of occurrence of spelling errors is not abundant and that the few data points that do exist must be qualified by

- the size of the corpus from which they were drawn
- the text entry mode of the corpus
- the date of the study.

She qualifies the latter by stating that 'newer studies for some genres, such as edited machine-readable text, probably reflect lower error rates due to the availability of automatic spelling checkers'.

In the next subsections we study a large contemporary corpus, most of which we assume has been keyboard input. We will see that Kukich's qualification does not find a basis in fact.

2.2.2 Reuters RCV1: facts and figures

The over 800,000 news stories in RCV1 were effectively sent in from all over the world. As such, we believe, it offers a cross-section of the various Englishes and represents 'International English' as employed by professional writers.

We limited this study to the word type list of word forms beginning with a lowercase character. This reduced the 1,626,038 items type frequency list (which was not normalized as regards digits) to 215,046 items. This was further reduced to 159,085 items by setting aside what can only be described as systematic bad tokenization cases. Any corpus seems to have its peculiarities which may not be provided for in the tokenizer.

We worked our way through the entire reduced list, aiming at completeness. We did this manually, i.e. unaided by an existing computer spelling checker, as we did not want to allow for the possibility that results would be influenced by some system's bias. We might have used

the GNU spelling checker ispell, for instance, just like Kernighan et al. (1990) used its predecessor spell to scan through 44 million words of Associated Press newswire to locate non-word strings in the text. Part of ispell's bias in this would be that it can be made to accept words it can split in two words in its dictionary. Wrongly concatenated strings such as *ifthe or *wasno it would therefore accept. Also, it was noted by Zobel and Dart (1995) that ispell's dictionary contains some spelling errors. If these happened to recur in our list, these too would go unnoticed.

Results presented are not valid for capitalized words, which contain the bulk of proper and other names. Names naturally allow for considerable variation. This is primarily because there are no orthographical rules for names. Neither are there generally agreed on transliteration rules for names from other languages or scripts. And names often record historical changes that have occurred within a language. The main reason we excluded the bulk of names by focusing on lowercase first character words, however, lies in the fact that we wish to measure if the availability of spelling checkers has indeed reduced the presence of non-word errors in corpora. The spelling checkers which have been available typically do not deal with names, or only to a limited extent.

2.2.3 Non-word errors

We aimed at identifying as many as possible of the non-word errors. What we consider 'admissible' variants, e.g. those produced by the differences between accepted British and American spelling, were left unmarked.

In all, we marked 33,488 word types as being typos: erroneous variants of other words. This error list constitutes just more than 21% of the 159,085 items long type list, an unexpectedly high proportion. Not included in this number is what we consider bad tokenization cases, abbreviations or foreign words. These we will briefly typify later in this section. In terms of tokens, 33,488 typos means that the Reuters rcv1 corpus contains 1 erroneous form per 400 running words with a lowercased first character.

The top three most frequently misspelled words and their corpus frequencies were *goverment [482], *milion [372], *occured [331].[8]

In Subsection 2.2.5 we give an overview of the types and number of occurrences of the errors for which we provided a correction: 12,094 items or over 36% of the error list. This list represents a random subset of the full error list. Whenever the word form gave rise to the least

[8] We present the corpus frequency of a word within square brackets.

doubt about the correct resolution of the error, we examined the context in the corpus itself. In the main, resolution is possible examining the sentence containing the error. The example of *absord, which might equally resolve to *absurd* of *absorb*, without or within its one sentence context, shows wider context is sometimes needed.

> SINGAPORE : Asia Products Outlook - Shutdowns to *absord length . [...] The ample length in the Asian oil product markets is due to be *absorbed* by fresh demand arising from refinery shutdowns [...]

Context may also serve to resolve problems with domain specific terms, not likely to be found in standard dictionaries:

> Sri Lanka bought 50,000 tonnes of *opitional-origin soft wheat for September shipment and 50,000 of *optional-origin* hard wheat for October .

2.2.4 Error types discerned and how we counted

There are multiple ways of looking at a typo/correct word pair and of classifying the difference between them. We here detail what categories of errors we discern in the error list we culled from Reuters RCV1 and how we count them. The actual method employed for classifying lists of typo/correct word pairs we detail in Chapter 5 and Appendix A.

Deletion or insertion: point of view

Given the pair *acress/*actress*, when viewed from the correct form, the error is a deletion error. Viewed from the point of the typo, it would require an insertion to correct the error. We follow Kernighan et al. (1990) in naming the transformations from the point of view of the correction.

Substition and transposition

Substitutions are substitutions, whichever way they are looked at. Transpositions, however, can equally well be described as constituting two substitutions. Or, as Navarro (2001) (p. 38) points out: 'note that a transposition can be simulated with an insertion plus a deletion'. To get from *trasnport to *transport* one may substitute the *s* by an *n* and the *n* by an *s* or one may delete the *s* and insert another *s* after the *n*. Looked at this way, which in actual fact is what most implementations of LD do, transpositions have an edit cost of two instead of one.

Run-ons and splits, space to dash

Apart from the four basic Damerau (1964) categories we also discern between run-ons and splits. Note that these are special cases of deletions and insertions, respectively, both involving a specific character: the space. Nevertheless, we counted these separately, but their counts

might be added to their respective parent categories. The same is true for the space to dash substitution: the typo was written with a dash, it should have been two words divided by a space.

Capitalization

We also present counts for capitalization errors, e.g. *spain having been written for *Spain*. These involve mainly names, but the odd capitalized character(s) in the middle or at the end of a word were observed to occur (e.g. *orieNTED). Of course, these too might have been summed in with the other substitution errors.

Out-of-alphabet characters

Both categories insertion and substitution may contain items that either contain punctuation marks or digits, i.e. out-of-alphabet characters (e.g. *arou3nd, *atr=tributed). These were few and were counted with their parent category.

Initial character transformations:

Other special cases we were interested in, involve deletions or insertions of words' first character(s). While we present separate counts for these (in Subsection 2.2.7), these counts were not subtracted from their respective parent categories. We want to know their prevalence in our corpus, but further regard them for what they are: deletions or insertions of one or more characters. Note that first character capitalization errors might as well be seen as constituting first character substitutions too, so that those counts might be added to the ones reported for first character substitutions involving different characters and not mere capitalized versions of the same character. But they were not. These figures are retrievable from the table. First character transpositions involve the first two characters of a word that happen to have traded places. Their counts too are part of the overall transposition counts.

Multiple errors: single or multipoint

For more elaborate types of errors, necessarily involving edit distances larger than LD 1, we discern two types: single point or multiple point multiple errors. This is because the LD does not tell the full story. Consider: *apppointements. This typo has LD 2 with its correct form, but the two insertions do not occur at the same point within the word: we find an extra *p* at the beginning and an extra *e* between the stem and suffix. We refer to this type of error as a multi-point error, in contrast to the far more prevalent single-point errors. We called the single-point multiple errors more elaborate in that they involve a mix of at least two of the basic error types, e.g. a deletion followed immediately by a substitution. Multipoint (multiple) errors may well involve the same

Category	LD 1	LD 2	LD 3	LD 4	LD 5	Total	%
deletion	4,026	239	9	2		4,276	35.36
insertion	3,370	196	17	2		3,585	29.64
transposition		1,566		5		1,571	12.99
substitution	1,447	91	4	1	1	1,544	12.77
multiple		398	89	11		498	4.12
run-on	314					314	2.60
capitalization	168	2		1		171	1.41
multisingle		52	15	2		69	0.57
split word	51					51	0.42
dash to space	15					15	0.12
total	9,391	2,544	134	24	1	12,094	
%	77.65	21.04	1.11	0.20	0.01		100

TABLE 2.3 Statistics of the error categories in the 12,094 typo/correction list.

basic type, but these errors are not adjacent. Of the latter type, we present our favourite: *momopology, which has an LD of 3 (but 2 non-adjacent edits: *m*-*n* substitution and *og* insertion) with its correction: *monopoly*, as may be inferred from its context:

> The club , which holds the *momopology on gambling in Hong Kong , has retained the services of its other two fund managers , [...] , the newspaper reported .

2.2.5 Statistics of the error types

In Table 2.3 we present the statistics of our list in terms of the categories of errors encountered. Categories are ordered by diminishing frequency of the error category.

Prevalence of the error types

We see that deletion occurs most often, followed by insertion errors. Together, these two categories account for over 65% of all typos observed. This changes the order of prevalence reported by Damerau and Mays (1989) which we presented in Subsection 2.2.1. We see that transpositions and substitutions occur with the same likelihood, each accounting for about one in eight typos. Over 4% of the typos observed involved multipoint errors, accentuating the need for correction strategies capable of handling several edits within the same word. Run-ons appear to occur more often than split words. However, this may be an artefact of our sampling method: split words viewed within a word list may well appear to be either existent words or likely abbreviations. Single point

multiple errors (referred to as multisingle in the Table) are relatively rare in our sample. So is the space to dash category, which could probably be seen to represent bad tokenization rather than real typographical errors.

LD distribution of errors

Let us now take a look at the LD-variation of the 12,094 items list of paired errors and corrections in the bottom row of the Table.

LD 1 covers all single character deletions, insertions and substitutions. LD 2 covers all 2 character transpositions, deletions, insertions and substitutions. Of these, there were 1,566 transpositions. The LD 1 cases and LD 2 transpositions together account for 90.6% of the typos, which is more than Damerau found and confirms the findings of Peterson (1986), as well as Pollock and Zamora (1984). LD 1 + 2 together cover 98.7% of all non-word errors identified and provided with a correction in our list. Given a correction system that is capable of correction up to LD 5 would allow for correction of all the typos we observed in RCV1.

What we also observe, if we remember that transpositions should also have an associated cost of 1 and not 2 as we actually report them in the table, is that if we look at the distributions in terms of the LD, we see that LD 1 occurs by at least an order of magnitude more than typos involving LD 2, which in turn occurs by an order of magnitude more often than LD 3 errors. And so on, to LD 5. We think this is important: when building evaluation benchmark sets, for instance, their composition might well fruitfully be guided by this finding as a rule-of-thumb. In other words, if one were to build a benchmark data set consisting of 10,000 LD 1 typos, 1,000 LD 2 typos, 100 LD 3, 10 LD 4 and 1 LD 5 one would get very close to the actual distributions we have observed in the real world. We will return to these considerations in Chapter 5.

2.2.6 Errors versus hapaxes

Many researchers have repeated over and over, that 'typographical errors, if any, will appear in the hapax legomenon' in the wording of a recent paper on Zipf's Law (Ha et al., 2003). In what went before, we believe to have shown that the 'if any' is unduly optimistic. That they only appear among the hapax legomena is clearly not the case. Figure 2.1 shows graphically that typographical errors recur frequently. The log-log plot of their frequency versus the total number of types (top curve) and typos (bottom curve) shows that the distribution of typos closely follows the Zipfian curve displayed by the correct forms. A small

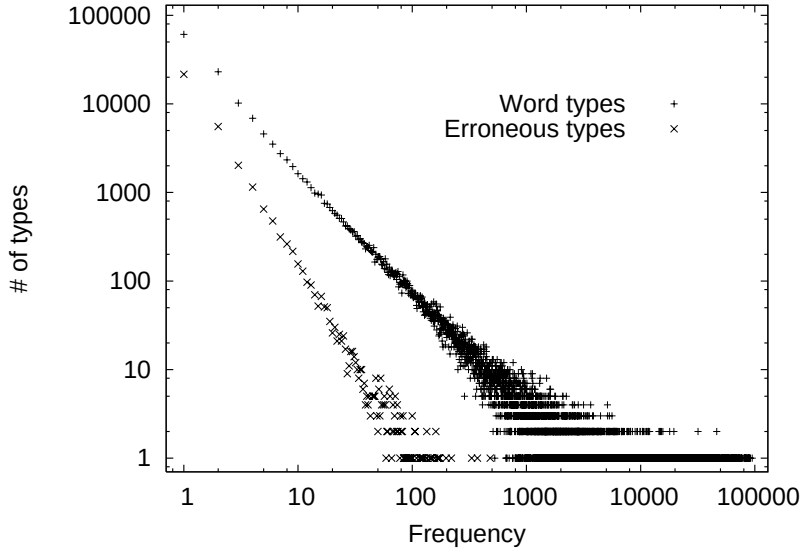

FIGURE 2.1 Log-log plot of total number of types versus number of erroneous types, per frequency.

note of warning is in order here: a log scale awards the same space on the scale to number 1 to 10 than it does to 11 to 100 or 101 to 1000, etc. So the curves appear much closer than they are in reality. Nevertheless, it can be observed that the most frequently occurring typos nearly join the distribution of the correct forms. Statistical linguists often try to circumvent the impact of typographical errors on language models by discarding the hapax legomena (Manning and Schütze, 1999). What we learn from our data is that the ratio of erroneous (21,576 items) versus correct (39,285 items) types for hapax legomena is 0.549, i.e. 55% of the hapaxes are correct word forms occurring only once in this particular corpus. The type list contains 122,153 correct types in all. Without the correct types occurring only once, the hapaxes, we would retain only 82,868 correct types. Of the 33,225 erroneous forms, removing only the hapaxes, we would still retain 11,649 typos. By discarding the hapaxes over 32% of the total amount of correct types are lost and 35% of the erroneous types retained.

At frequency 10, 156 of the 1,627 types in the RCV-1 list are typos, or 9.59%, accounting for 27.50% of the types up to that point and 20.39% of the total number of tokens occurring up to ten times. We find *currenty, *curreny, *currrently sharing the company of *curators*,

goverment [482] 1	governmenment [3] 3	governmnt [1] 1
governemnt [93] 2	governmeent [3] 1	governmnnt [1] 1
govenment [52] 1	govennment [3] 1	governmetn [1] 2
governement [42] 1	govrenment [2] 2	governmennt [1] 1
governent [39] 1	govnerment [2] 2	governmemnt [1] 1
governmment [32] 1	governmenet [2] 1	governmement [1] 2
governnment [24] 1	governmen [2] 1	governm+ent [1] 1
govenrment [19] 2	governmant [2] 1	govermnemnt [1] 3
governmnent [16] 1	govermnent [2] 2	goverbnment [1] 1
governmnet [14] 2	govermment [2] 1	govenrnment [1] 1
governmemt [8] 1	fovernment [2] 1	govenrement [1] 2
governemt [7] 2	bovernment [2] 1	gopvernment [1] 1
goverenment [6] 1	govrmment [1] 2	goovernment [1] 1
governmet [4] 1	govrernment [1] 1	givernment [1] 1
govermnment [4] 1	goversment [1] 1	giovernment [1] 1
governmenmt [3] 1	governvement [1] 2	gavernment [1] 1

TABLE 2.4 *government* variants, their frequency in Reuters RCV1 and their LD to the correct form *government*.

curtly and *curvy*, and *sharesholders sharing the rank of *share-holders*. At frequency 100, 3 of the 68 types are typos, or 4.41%, accounting for 23.47% of the types up to that point and 7.29% of the total number of tokens occurring up to a hundred times. We find *millon in among *long-ruling*, *manuals*, *non-traditional* and *nude*, as well as *signficant in between *shrewd* and *solicitor*. Up to frequency 482, where we found the most recurrent typo: *goverment, 22.14% of all the types were in error. These account for 2.85% of all the tokens up to that point. At this rank we find types such as: *globally*, *inappropriate*, *needing*, *non-bank* and *recipients*.

2.2.7 Variants for 'government'

If we take a closer look at the variants for *government* in the Reuters Corpus (Table 2.4), we see that this single type has spawned 48 variations. This is when we focus only on the singular, lower-cased form, disregarding the abbreviated and badly tokenized forms and whatever happens to the genitival, plural or compounded forms.

Effect on statistical estimates

Typos are statistical events seen, which should not have been seen. We do not know at this point how the amounts of non-words we observe in a large corpus would affect the predictions of Large Number of Rare Events or LNRE models, as described by Baayen (2001). What we can

conclude is that the LNRE distribution in a large corpus appears to be an Overly Large Number of Rare Events distribution, if over 20% percent of the types observed should not have been observed. What we do know is that Evert (2004) specifically mentions the use of data clean-up on the large English (BNC) and German corpora he uses to evaluate the two new LNRE models he proposes.

Typos detract from the frequency mass of the correct form. In terms of the frequency mass of the type *government*, its 48 variants account for 892 occurrences versus the corpus frequency of 163,697 for the correct form *government*, which means its variants account for 0.542% of what should have been its total frequency mass. Discarding the hapaxes from this list of variants still leaves us 28 non-word variants, which is still 0.530% of what should have been the frequency mass. Discarding the hapaxes, then, does little to redress the balance. Discarding the hapax dis legomena too, still leaves us 21 non-word variants and removes only a further 18 occurrences, so we are still short of 0.519%. Contrast this to the LD-distribution of these variants: 34 variants have LD 1, twelve variants LD 2 and only two LD 3, so nearly 96% of the variants fall within 2 edits. If one were able to fully automatically identify and correct these, this would leave only 4 occurrences out of 892 'displaced' corpus counts, or only 0.002% of what should have been the type's frequency mass. We think this would be conducive to a better unigram probability estimate.

Of course, it can be argued that for this particular word, which to all probability has an even distribution over the corpus, the loss in frequency mass due to variations is negligible and will not unduly affect probability estimates. But Curran and Osborne (2002) wrote:

> These large corpus experiments demonstrate the failure of simple Poisson models to account for the burstiness of words. The fact that words are not distributed by a simple Poisson model becomes even more apparent as corpus size increases, particularly as the effect of noise and sparseness on the language model is reduced, giving a clearer picture of how badly current language models fail. [...] Without better models all that training upon large corpora can achieve is better estimates of words which are approximately i.i.d. [i.e. independently and identically distributed].

The effect of burstiness is perhaps best illustrated by one of their own examples: *tightness*, which occurred 2,652 times in their corpus. Their corpus is not completely disjoint with ours: it comprised the RCV1. In the RCV1 the word has a corpus frequency of 1,024. But we also find: *tighteness [8], *tighness [5], *tightnes [2] and *tightnesss [1]. Which means that in the RCV1 alone 1.54% of the word's frequency

mass is distributed over variants. Another, more extreme example in our corpus would be the word *hygienic* [22]. This has a low frequency in the RCV1 and yet we find a variant: *hygenic [8]. This variant then accounts for 26.7% of the correct form's frequency mass. At the most extreme end, we observed cases where the correct form was lacking altogether, all that was observed was a variant. An example of this is *labrynths, although the list did have: *labyrinth* [8], *labyrinthine* [6] and even *labyrinthitis* [1].

We would like to argue that what emerges from this, is that given larger corpora more and more noise in the form of incorrect typographical variants of the orthographically correct word types is incorporated. We fail to see how the effect of more noise can then be said to be reduced by using larger corpora. As regards sparseness, Curran and Osborne (2002) compiled a homogeneous corpus of 1.145 billion words of newspaper and newswire text from three existing corpora. We think the best way to effectively combat sparseness would be to include text from diverse other sources as well, say: include all the books available in the Gutenberg archives[9], to start with. This would not help to alleviate sparseness as concerns neologisms and, to a certain extent: names. It might well help as concerns the 'rare but not necessarily unusual words' (Curran and Osborne, 2002) (p. 129).

Text Categorization

The RCV1 was made available by Reuters as a benchmark collection for Text Categorization research purposes. Lewis et al. (2004) describe in detail how the corpus was built up, how it was encoded for text categorization purposes and provide benchmark results obtained by three Machine Learning algorithms. They describe what the text representation used for benchmarking the collection looks like. Text is reduced to all lowercase characters. They define tokens to be maximal sequences of nonblank characters; sequences consisting purely of digits were discarded, as well as those occurring in a stop word list (i.e. a list of highly frequent words, which therefore have no discriminative value for text categorization). Using their own implementation of the Porter stemmer (Porter, 1980), the tokens were stripped of their suffixes and thus reduced to their stems. As no two implementations of the stemmer behave identically, they provide their list in online appendix 14 to Lewis et al. (2004)[10]. The final token list thus obtained finally contains 47,236 stemmed tokens (recall that the full frequency list we derived from the

[9] http://www.gutenberg.org/

[10] http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/~a14-term-dictionary/stem.termid.idf.map.txt

gov	16497	4.88602502858038
gover	16498	6.5907731208188
govern	16499	1.63033523058946
governemnt	16500	9.36336184305859
governmetn	16501	10.0565090236185
governmn	16502	8.67021466249864
governmnet	16503	10.0565090236185
governor	16504	4.16210618935368
govet	16505	6.62252181913338
govmt	16506	9.36336184305859
govt	16507	5.03262850277225

TABLE 2.5 Stems beginning on *gov* from online appendix 14 to Lewis et al. (2004). Shown are: the stem, the integer token identification number, inverse document frequency value (IDF). Note that the unstemmed typos have a low frequency and thus gain greater weight for Text Categorization by obtaining greater IDF values.

corpus contained 1,626,038 items).

We list the stemmed tokens beginning on *gov* in Table 2.5. It can be seen that this list contains 6 stemmed index items for the various word forms of *govern*, apart from 3 likely abbreviations. Note that the Porter stemmer (Porter, 1980) regards the *-ment* in *government* as a suffix to be stripped.[11] This is in contrast to the *-or* in *governor*, which is not stripped and the *-ness* in *governess* which is stripped, leaving the stem *gover*.

We cannot assess here what the possible impact of this is on Text Categorization results. At least in part, that will be dependent on the specific Machine Learning algorithm employed. What is clear, nevertheless, is that typos affect the index of terms used for this research. In five of these cases, the typographical variance precluded proper stemming, i.e. produce the stem *govern*, from a variant of the type *government*.

In order to show that just about any word may spawn an impressive range of variants, Table 2.6 lists the items for which we observed 20 or more variants in the subset of the RCV1 derived typo list we provided with corrections. The full typo list may well contain many more.

[11] The webpage at http://www.tartarus.org/∼martin/PorterStemmer/, maintained by the algorithm's author: Martin Porter, allows the reader to verify this. Compare the morphological variants of *govern* in the sample vocabulary provided there with the algorithm's output.

government (48)	opportunities (25)	communications (22)
percent (38)	against (25)	acquisition (22)
significant (32)	responsibility (24)	situation (21)
restructuring (30)	parliamentary (24)	parliament (21)
immediately (30)	opposition (24)	international (21)
newsroom (28)	negotiations (24)	agreement (21)
said (27)	business (24)	about (21)
million (27)	significantly (23)	possibility (20)
between (27)	previous (23)	pharmaceutical (20)
available (27)	operations (22)	activities (20)
company (26)	operating (22)	
particularly (25)	management (22)	

TABLE 2.6 Words with 20 or more variants in the corrected list, number of variants observed (between brackets).

Category	LD 1	LD 2	LD 3	LD 4	LD 5	Total	%
1st ch. delet.	122	11				133	1.10
1st ch. insert.	55	3				58	0.48
1st ch. transp.		23		1		24	0.20
1st ch. sub. lc.	106					106	0.88
1st ch. sub. uc.	138					138	1.14
% (uc.)							3.80
% (lc.)							2.65

TABLE 2.7 Statistics on first character transformations in the 12,094 typo/correction list.

2.2.8 More specific statistics

First character errors

Kukich (1992) (p. 388) writes that 'it is generally believed that few errors tend to occur in the first letter of a word'. She goes on to state that only a few studies actually document first-position statistics. Table 2.7 presents the statistics on first character transformations in the 12,094 typo/correction list. We distinguish between the counts where first character capitalization errors (1st ch. sub. uc.) are also counted as first character errors. Together with first character substitution errors not involving capitalization (1st ch. sub. lc.), they make up the category of first character substitution errors. One might wish to include capitalization errors, or choose not to, but we think that it is at least necessary to state which point of view is adopted, when a system is

L	#	L	#	L	#	L	#	L	#
2	21	7	1,451	12	967	17	44	23	1
3	85	8	1,802	13	581	18	12	24	4
4	305	9	1,846	14	357	19	16	25	1
5	503	10	1,684	15	134	20	18	26	2
6	932	11	1,255	16	64	21	8	27	1

TABLE 2.8 Length distribution of the corrected word forms in the 12,094 typo/correction list. Pairs: left: word length in characters (L); right: number of words (#).

described or evaluated.

Disregarding the capitalization substitutions, 2.65% of the errors involves the first character. We think this is not a negligible amount. It is actually slightly more than the run-ons we have observed. If one expects a spelling correction system to be able to cope with the latter, the former should certainly not be left out. This is what happens with e.g. the SOUNDEX similarity key (Odell and Russell, 1918/1922), which uses the first character to partition and thereby reduce the search space.

Breakdown by word length

Kukich (1992) (p. 388) further wrote:

> Unfortunately, little concrete data exists on the frequency of occurrence of errors by word length.

We present Table 2.8 to help remedy that. We counted the lengths of the correct word forms, not the lengths of the typos. Figure 2.2 shows graphically that the corrected typos show a distribution which is similar to the distribution of the correct types in the RCV1 frequency list.

In Table 2.9 we contrast the distributions of errors of the correct word forms of length less than five with those of word forms of length five or more. In order to facilitate comparison with the full table presented before, categories are ordered as they were there. Note that for the short words, insertion is now the most frequently observed error category, accounting for over 52% of the typos, followed by over 22% of substitution errors. Clearly different mechanisms are at work as regards short words. It will be readily understood that for words that are at most 4 characters long, greater LD variation is not likely to be observed. The table shows one case where we have an LD of 3, for the typo *olp, for which the correct form should be *to*, given the context:

> After severing its ties with Mobil last year the IAAF paid $ 2 million *olp support the circuit and will pay a similar amount next season .

It was clearly only by virtue of examining the context that the error

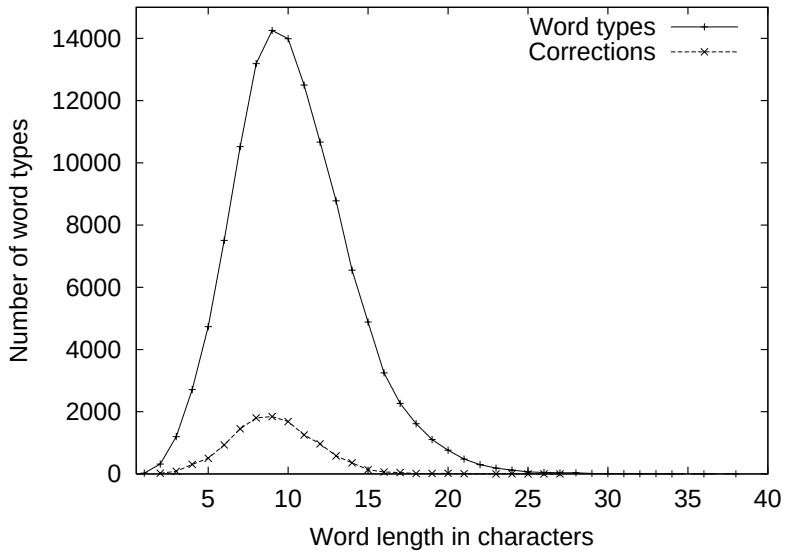

FIGURE 2.2 RCV1: Plot of the total number of word types per word length and the number of corrected typos per word length. Word length is measured in characters. The distributions observed are similar.

could be resolved to *to*. One lesson to be learned from this is that for short words, one should not let one's spelling correction system search too far.

The particular typo we just discussed happens to be a clear case of a keyboard proximity error, at least as regards the additional *lp*. This might prompt one to try to model this or other types of error. In this study we draw no such conclusions, nor do we try to devise specific spelling correction techniques on the basis of studying spelling error patterns. As shall become clear in Chapter 3, our algorithm effectively searches for all possible solutions, within bounds.

Inter-corpora statistics: recurrence of non-word errors

For the 33,488 errors culled from the Reuters RCV1, we also checked whether they recur in other corpora.

We observed:

± 14% overlap with NYT: 4,628 of the typos recur

± 13% overlap with BNC: 4,325 of the typos recur

Just over 28% of these recurrent typos appeared in all three corpora. These would be prime candidates for the **absolute** correction strategy as defined by Pollock and Zamora (1984), which they call limited but very cost-effective. These typos could be put in a misspelling dictionary, which when one of these is encountered, simply returns the correct form. In an isolated-word correction system, this requires that there is only a single unambiguous correction possible. A context-sensitive correction system should in principle be able to deal with recurrent typos having more than one resolution.

It will be clear no conclusions regarding the true error rate within the NYT or BNC can be drawn from these findings. What this does illustrate is that probably no corpus is free from Zipfian-distributed typos.

2.2.9 Final note on disregarded word strings

As we remarked before, we left foreign words, abbreviations and bad tokenization cases out of the accounting of the error types in the RCV1. For the sake of completeness, we here typify and discuss these in a little more detail.

Foreign words

Foreign words were marked as such and excluded from the typo list. The list contains quite a number of French words, to a lesser extent Dutch, German, Spanish and others. Unfortunately unknown to us when we started work on the RCV1 frequency list, Khmelev and Teahan (2003) had reported on about 400 foreign language news stories in the RCV1,

Category	LD 1	LD 2	LD 3	LD 4	LD 5	Total	%
Words of length < 5							
deletion	45	1				46	11.19
insertion	208	7				215	52.31
transposition		43				43	10.46
substitution	87	4				91	22.14
multiple							0.00
run-ons	2					2	0.49
capitalization	11					11	2.68
multisingle			1			1	0.24
split word	2					2	0.49
space to dash							0.00
1st ch. delet.	(5)					(5)	(1.22)
1st ch. insert.	(28)	(2)				(30)	(7.30)
1st ch. transp.		(4)				(4)	(0.97)
1st ch. sub. lc.	(24)					(24)	(5.84)
1st ch. sub. uc.	(11)					(1)	(2.68)
total	355	55	1			411	
%	86.38	13.38	0.24				100
Words of length ≥ 5							
deletion	3,981	238	9	2		4,230	36.21
insertion	3,162	189	17	2		3,370	28.85
transposition		2		5		1,528	13.08
substitution	1,360	87	4	1	1	1,453	12.44
multiple		398	89	11		498	4.26
run-on	312					312	2.67
capitalization	157	2		1		160	1.37
multisingle		52	14	2		68	0.58
split word	49					49	0.42
space to dash	15					15	0.13
1st ch. delet.	(117)	(11)				(128)	(1.10)
1st ch. insert.	(27)	(1)				(28)	(0.24)
1st ch. transp.		(19)		(1)		(20)	(0.17)
1st ch. sub. lc.	(82)					(82)	(0.70)
1st ch. sub. uc.	(127)					(127)	(1.09)
total	9,036	2,489	133	24	1	11,683	
%	77.34	21.30	1.14	0.21	0.01		100

TABLE 2.9 Breakdown of the statistics of the error categories for words less than 5 characters long (top half) versus words **5 characters or more in length** (bottom half) in the 12,094 typo/correction list. Counts between brackets are subsumed by the parent category.

which was to have been an exclusively English corpus. The authors recommend that these be removed if the corpus is used for Natural Language Processing purposes.

Abbreviations

The full list further contains countless and indeed uncounted abbreviations. When the context clearly showed these word forms to be abbreviations, we marked them as such and further left them out of consideration. If we have missed out on typographical errors in the list, it will be primarily in these shorter word forms.

Bad tokenization

We set aside 55,953 strings, primarily run-ons involving words concatenated to some sequence of digits (e.g. *item3,383 - *costs0.25 - *taxes692) which resulted from bad tokenization, in particular imperfect handling of tables. These strings were also not included in the statistics provided earlier. Bad tokenization should be solved at the level of tokenizing a text. Given the systematicity of this type of problem, this can be done: regular expressions could easily be written to extend the tokenizer to remedy this. Nevertheless, a spelling correction system should likely also be able to deal with this type of error.

2.2.10 Summary

We have studied a large sample of typos in English which should be representative of what one may encounter in a contemporary corpus. We think it is fair to conclude that our findings do not bear out Kukich's qualification that given the availability of spelling checkers less non-word errors occur than there used to before the advent of spelling checkers. Put quite simply: the non-word error problem has not gone away. It does not even show signs of being on the retreat. The extent to which a contemporary corpus has here been shown to contain non-word errors is to all intents and purposes the same as reported by Kukich (1992). Our findings are also in line with Pollock and Zamora (1983) (p. 53), who report an overall incidence of 0.20% of misspellings in the databases they studied and state that this is 'probably what one should expect in raw keyboarding by experienced operators'. We found an incidence of 1 in 400 tokens in the Reuters RCV1 corpus, or 0.25%. We take this figure to constitute the natural distribution of typos in keyboarded text throughout the rest of this work.

We have not tried to quantify the size of the real-word problem: real words cannot be detected by studying a frequency list. Neither have we tried to quantify the proportion of cognitive errors versus typographical errors. As we have explained in Chapter 1, Subsection 1.3.2.: it is hard

to see what caused the error from the output (Damerau, 1964). What we have quantified is that only very few typos have an LD larger than three. We observed only one single case where the LD was 5 in 12,072 typos. This was *seeked for *sought*, which is a grammatical error, but nevertheless results in a non-word.

We have seen that discarding the hapax legomena and hapax dis legomena from a corpus only allows for removing about 66% of the variation, at a cost of losing about 33% of the real-word types. We have detailed what the impact of this is on the frequency mass measured for one particular word, *government*. While it can be argued that for this particular word, which most likely has a very even distribution over the corpus, the loss in frequency mass due to variations is negligible and will not unduly affect probability estimates, we have argued that the situation is likely to be more dramatic for words more bursty in nature.

Given an automatic spelling correction system that achieves not only good recall, i.e. is capable of correcting the typos it finds, but also high precision in doing so, i.e. does not report real words to be non-words and replaces them by other real-words, up to 77,6% of the variants present in a corpus might be removed by correcting only those typos that are within LD 1. By correcting only the typos that are within LD 1 + 2, which given the computational resources available to date should be well within reach, up to 98.7% of the variation within a corpus might be removed. We think pursuing this goal has far better chances of alleviating the data sparseness problem and of improving statistical language models than the common practice of hapaxing has.

In the next chapter we describe our approach to detecting and correcting typos with good recall and great precision. In Chapter 4 we substantiate this with extensive evaluations.

3

Text-Induced Spelling Correction

3.1 Introduction

In Chapter 1 we sketched various approaches to spelling correction, most of which are based on performing string edits of some kind. In Chapter 2 we studied what kinds of spelling variation actually occur in a large corpus of contemporary English edited text. In what follows we present our approach to spelling correction, which in essence avoids the need for performing string edits by performing table look-up instead. The corpus-derived table containing the lexicon defines the language as well as the language's orthography. The alphabet provided delimits the major bounds of what can be regarded a spelling variant of another word form and allows for a systematic and complete search for these forms within the table. These features provide this algorithm's strongest characteristics: its unsupervised nature and its language independence. The fact that it should work with any alphabet, and by extension, language, makes it to a large extent language-independent. Taken together, these features provide a cheap solution to otherwise computationally expensive problems.

3.2 The correction algorithm

We develop the idea of using a corpus as the basis on which to build a spelling correction system. Given our findings in Chapter 2, we know that for most erroneously spelled word forms, the corpus contains far more counterexamples of the correct form.

3.2.1 Anagram hashing

We line up all those word forms present in the corpus that consist of the same set of characters. This alignment forms the basis for a corpus-derived lexicon and spelling correction system. A means to align only the word forms consisting of the same set of characters in a completely

unsupervised way was found in the theory of hashing, be it in the 'bad' part of it, in the normally avoided generation of collisions. Collisions occur when the mathematical function used to bin the information, puts more than one item of information in a single bin (Knuth, 1981). The mathematically simple function introduced and exploited here does precisely that, for all strings containing the precise same set of characters.

For each word type or word type combination (compound or word bigram) to be included in the TISC lexicon, we obtain a numerical value, which will serve as the hash key. The formula represents the mathematical function we devised to do this, where f is a particular numerical value assigned to each character in the alphabet and c_1 to $c_{|w|}$ the actual characters in the input string w.

$$Key(w) = \sum_{i=1}^{|w|} f(c_i)^n$$

In practice, we use the ISO Latin-1 code value of each character in the string raised to a power n. We currently use 5 as the value for n. This was empirically derived: lower values do not produce collisions between anagrams only. The rather large natural number produced by this function in effect raises the numerical distance between any two characters to such a degree, that only the strings containing the same set of characters are assigned the same natural number. This means that all anagrams, words consisting of a particular set of characters and present in the lexicon, will be identified through their common numerical value. As the collisions produced by this function identify anagrams, we refer to this as an **anagram hash** and to the numerical values obtained as the **anagram values**, further abbreviated as AVs, and **anagram keys**, when we discuss these in relation to the hash. Table 3.1 shows an extract from the lexicon derived from the combined NYT-BNC bigram list from which the bigram hapaxes were discarded. All lines contain the string *whale* as well as the anagrams observed, if any, for the particular word combinations formed on the basis of the string. We further discuss the lexicon and provide a rationale for applying frequency cut-offs to the bigram list in Section 3.3.1.

In the implementation we use chaining for collision resolution, as the anagram keys and their associated word forms are there stored in a regular hash. So the anagrams colliding to the same AV are associated to it in a linked list. The anagram key will enable us to look up immediately whether any string consisting of the same character set as the input string was encountered in the corpus. As will be explained

Anagram key	anagrams
69820787149	elhwa, elwah, elwha, wahle, whael, whale, wheal, wleah
70060304557	. whale, whale .
70199366832	3 whale, whale 3
78441681838	a whale, whale a
97745731164	of whale, owe half, whale of
98722257206	in whale, whale in, while an
99445098414	a wealth, wealth a, whale at
102637158206	whale ice
102730729081	as while, he wails, she wail, whale is, while as
102810023132	an whole, heal now, how lean, lane who, lean how, no whale, on whale, whale on, who lean, whole an
103249105917	bag while, big whale
103620573782	ale with, at while, la white, the wail, the wali, wail the, whale it, what lie, while at, with ale
104056682337	each lawn, whale can
104546781838	and whale, whale and
105959068956	a howler, earl who, how real, or whale, whale or, who real
106704923132	add whole, odd whale
106735035630	whale bar
106818495007	as whole, hole saw, hole was, sale who, seal who, so whale, whale so, who seal, whole as, whose al
107708339708	at whole, heat low, how late, late how, late who, low heat, tale who, to whale, whale to, who late, whole at
109763065238	fail when, fan while, fin whale, half wine
109955198915	eat whale, wheat ale
109992590371	deaf whale, flew ahead
110982753180	beachwalk, whackable, whaleback
113320123633	one whale
113685671137	whale calf
113767373230	narwhale
113771537113	if whales, whales if
116818495007	al showed, deal show, deals who, do whales, heads low, how deals, howled as, lad whose, lead show, leads who, low heads, owls head, show deal, show lead, was holed, whales do, who deals, who leads, whole sad
170274311781	white whales
193505358311	Melville's whale
204875577601	exploding whale
218430814371	important whale, white patrolman
235790947870	underwear should, whale surrounded
246647271323	bottlenose whales
338099502351	whale-tracking technology
340344255445	whale-tracking hydrophone
342030600028	whale-hunting forefathers
357657717136	whale-watching excursions
358755368115	whale-watching expeditions
364085024665	boy-meets-killer-whale movie
391720180929	3-million-year-old proto-whales

TABLE 3.1 Extract from a TISC lexicon with the anagram keys and associated, chained anagrams. The lexicon is based on the NYT-BNC bigram list with bigram hapaxes removed. Note that the first line shows that of the 120 theoretically possible words containing a particular set of 5 characters (here : a, e, h, l, w) only 8 were actually observed. Note too that longer word strings rarely have anagrams.

below, a short sequential search will then allow for the retrieval of the anagram which best matches the input string. When the actual AV is not found in the lexicon, close numerical neighbours might very well be present, and simple arithmetic will allow us to identify and retrieve these potential correction candidates or CCs.

This novel representation makes the implementation computationally efficient. The net effect of obtaining anagram hash key values is that it provides a cheap abstraction from the surface sequence of characters. This furthermore allows, through simple addition, subtraction or both, for moving from one particular combination of characters to another. The numerical difference obtained by means of this abstraction will in all cases be exactly the same for e.g. the difference between 'randomise' and 'randomize' or any other verb possibly ending in '-ise' or '-ize'. The same goes for all systematic differences between e.g. American and British English (think of single or double 'l' or 'ou' versus 'o').

Anagram key based spelling correction is an inexpensive solution to the string correction problem as it limits expensive searching: it relies primarily on the non-search strategy implied in hashing. In hashing search is limited in that it is known, through the hash key, what is available and where a particular item is to be found.

Number of elements per hash key

The maximum number of elements per anagram hash key is necessarily limited to the number of possible permutations defined by the number of characters $c_{|w|}$ in the string(s) assigned to a particular hash key. This number of permutations is the factorial $c_{|w|}!$. For a set of three characters, the number is $3 \times 2 \times 1 = 6$, for a set of five characters it is 120, while a set of 10 characters allows for 3,628,800 words to be formed. For longer words it is very rare for even a very small fraction of this theoretically possible number to be observed within a given language. Storing all the anagrams in a chained list linked to the anagram key, thereby solving the collision problem, is also quite efficient. Knuth (1981) (p. 521) states:

> Chaining is quite fast, because the lists are short. [...] In general, if there are N keys and M lists, the average list size is N / M; thus hashing decreases the average amount of work needed for sequential searching by roughly a factor of M.

In our case, taking for example the English lexicon obtained with bigram list cut-off frequency 2, we retain 16,329,811 unigrams and bigrams, the keys. Linked to their anagram keys, we have 13,445,472 lists. The average chained list size is 1.215, which shows that indeed

only a very small fraction of possible word forms given specific subsets of characters actually occur. As a consequence, the necessary sequential search through the list of chained anagrams for the anagram matching best with the input form will be short.

3.2.2 Anagram key based correction

Based on a word form's anagram key it thus becomes possible to systematically query the lexicon for any variants present, be they morphological, typographical or orthographical.

The list of anagram values for the character(s) collected from the input type we further refer to as the **Type-derived Anagram Values**, abbreviated: TAVs. Given e.g. the type *lolita* we collect the AVs for the single characters. Then we add a space front and back (we clarify the reason for this later, the space is represented as an underscore, here): _*lolita*_, derive the AVs for the character bigrams: _l, lo, ol, li, it, ta, a_ and store these. If the type is longer than 5 characters, we also derive the AVs for the character trigrams: _lo, lol, oli, lit, ita, ta_ and store these, too. So, in all, we derive n unigram values, n+1 bigram values and n trigram values. Given the number of characters $c_{|w|}$ in the string w we get $c_{|w|}+(c_{|w|}+1)$ if $c_{|w|} < 5$ and $(2 \times c_{|w|})+(c_{|w|}+1)$, otherwise.

What actually constitutes the alphabet we explain in Subsection 3.3.2. Let it suffice for now that the alphabet contains AVs representing characters and character combinations, not actual characters. We further refer to the AVs **in the alphabet** as the AAVs.

Correction candidate retrieval Figure 3.1 shows a schematic representation of the core correction mechanism. We use the AV for the input string and the list of TAVs and the longer list of AAVs to query the lexicon for variants to the string we need to find correction candidates for. These variants can all be seen as variations of the usual error type taxonomy due to Damerau (1964):

transpositions These we get for free: they have the same anagram key value, so when queried for the input word AV, the lexicon returns the correct form and its anagrams (if any).

deletions We iterate over the alphabet and query the lexicon for the input word anagram value plus each AAV.

insertions We iterate over the TAVs and query the lexicon for the input word anagram value minus each TAV.

substitutions We iterate over both TAV and AAV lists adding each value from the AAVs and subtracting each value of the TAVs to the input word anagram value and repeatedly query the lexicon.

By systematically querying the lexicon hash we retrieve all possible CCs that fall within reach and apply standard string matching tech-

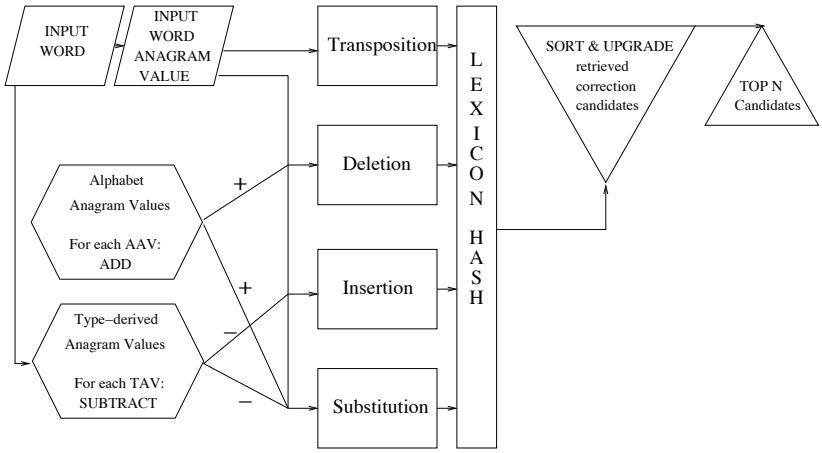

FIGURE 3.1 The core correction mechanism. The lexicon is queried for transpositions on the basis of the input word anagram value (IAV) alone. Each AAV is added to the IAV to query for deletions. Each TAV is subtracted to query for insertions. Each AAV is added and each TAV subtracted to query for substitutions.

niques to retain those that either in front or back match the input type for a specific number of characters, depending on the input type's length. We then iterate over the list of CCs retained and apply string matching rules to rank the list according to the plausibility that a given CC is the correct form for a particular typo. This step we call **upgrading**. We list the actual upgrading rules applied in the next paragraph. We finally retain only those CCs whose Levenshtein distance (LD) (Levenshtein, 1965) does not exceed a specified limit, as the algorithm is not in itself limited to a particular LD, which we will further discuss in Subsection 3.2.3. The CCs have then been ranked and the top n are proposed as correction candidates. Ranking is thus in part an automatic side effect of the algorithm which produces more hits on the actual most plausible CCs.

Upgrading is making it appear as if some CC were retrieved more often than it actually was because one or more string matching rules apply. Upgrading is achieved by adding a specific amount to a particular CC's count of times it was retrieved. The amount is specified as a parameter at run-time. Throughout this work we used 3 as the upgrading amount, prior experiments during development having shown that adding nothing resulted in poorer best-first ranking, adding one helped a little, adding three helped best.

We upgrade only those items whose 'cooccurrence frequency counts' (to be discussed in full in 3.3.3) are higher than a particular threshold (i.e. the Zipf Filter amount set, to be discussed in full in 3.4.1) and whose LD is lower than the LD limit set (the subject of 3.2.3), when one or more of the following string matching rules apply:

- if the lowercased CC matches with the specified number of initial characters of the input word and with the specified number of final characters
- if the lowercased CC matches with either the larger specified number of initial characters of the input word or with the larger specified number of final characters
- if the CC matches fully with the input word stripped of its first character
- if the input word matches fully with the CC stripped of its first character
- if the CC matches fully with the input word stripped of its last character
- if the input word matches fully with the CC stripped of its last character
- if the CC matches with the input word or the input word matches with the CC, regardless of the match being complete or not
- if the lowercased CC matches fully with the lowercased input word
- if the CC's first character is uppercased and the lowercased versions of CC and input word fully match

We further refer to the second rule as to the 'first-or-last-characters-matching' rule and to the rules 3 to 7 as the 'the-one-fits-in-the-other' rules.

The first two rules are the most important. Note that if the first applies, the second will apply too. The same set of upgrading rules is applied whenever the core correction mechanism is invoked. In Section 3.2.4. we show that this happens more than once, for a particular typo. We first detail by means of an example how the CCs retrieved by the core correction mechanism are upgraded and ranked.

Example We see what happens when the core correction mechanism is applied to *goverment, running TISC with a trusted dictionary. This produces 29 hits in all, retrieving the words in Table 3.2 with their frequency of retrieval. The examples show that the actual placing of the characters within the retrieved words is irrelevant. The correct word *government* was retrieved once on the basis of addition of the value for the missing character *n*, the other twelve times on the basis of the

CC	LD	FR	UR	URF
government	1	13	government	98
provement	4	3	overgarment	4
overgarment	3	3	provement	3
revetment	2	1	governments	2
regiment	4	1	emergent	1
ravelment	6	1	goniometer	1
ravagement	5	1	regiment	1
governments	3	1	ravelment	1
goniometer	3	1	convergent	1
emergent	3	1	revetment	1
convergent	5	1	ravagement	1
averment	2	1	averment	1
agreement	4	1	agreement	1

TABLE 3.2 *goverment: correction candidates retrieved (CC), their Levenshtein distance (LD) to the typo and their frequency of retrieval (FR). Column 4: upgraded ranking (UR) based on upgraded retrieval frequencies (URF).

substitution of the anagram values for each of its constituent characters for the character bigram value of each of these characters with the value for n. These character bigram values are contained in the alphabet. This repeated retrieval introduces desirable redundancy: it is an artefact of abstracting away from the actual character sequence through the anagram key values, but should be seen as a desirable side-effect as it helps to converge on what is usually the best correction candidate, besides the actual input string itself, if present in the lexicon.

After upgrading, this list looks as in column UR: shown are the words, their upgraded retrieval count and the LD to *goverment. It can be seen that only the first ranked correction candidate falls within LD 1. The far less plausible correction candidate *goniometer* was retrieved on the basis of subtracting the value 22,877,577,568, which represents the character v and adding the value 29,613,397,176, which represents the character bigrams oi and io = 6,735,819,608.

The LD forms the basis for the third ranking step, which filters out those that exceed the LD limit which was set, in this case LD 3. This leaves the words in Table 3.3, only the top 5 ranked of which are further passed on to the post-correction evaluation step. We only retain the top 5 to cut down on processing in comparing the outputs from the various times the core correction mechanism is invoked. We briefly experimented with retaining the top 10, but saw no performance gains.

Correction candidate	Retained?
government	yes
provement	yes
governments	yes
ravelment	yes
convergent	yes
revetment	no
averment	no

TABLE 3.3 *goverment: final ranking of correction candidates on the unigram level: top 5 only are retained.

It is possible that if the top 10 or more were passed on for shorter words only, performance would improve. We clarify this in Chapter 4.

Number of possible hits per error type In what follows we will discuss per error type, how many hits on the correct form may occur. We discuss this in terms of *x/y* replacements, meaning that the character(s) *x* are replaced by the character(s) *y*. Bear in mind no actual replacement of characters takes place in our algorithm, that anagram values are added or/and subtracted from the input string's anagram value, allowing for a hash key to match and a CC retrieval to occur. The reason we add a space front and back to the input string when collecting the TAVs is to allow for first or last character changes to be retrieved as many times as in-word changes, i.e. to also reap the benefit of character bigram to character unigram substitutions on these cases, which would otherwise not be retrieved as often.

The example of *goverment has shown us that in the case of a single character deletion error, the correct form may be retrieved as many times as the input word has characters, plus two for the spaces added, plus one for the single character *n*, plus one for substitution of the null-value by the AV for *n*.

The picture is different for a two-character deletion. For *govement both *movement* and *government* are retrieved, among 27 CCs in all. The correct form, using an alphabet containing only character unigram and bigram values is only retrieved twice, on the basis of adding the appropriate character bigram value. The other CC, *movement*, is actually retrieved three times, on the basis of g/m replacement, space-g/m replacement and go/mo replacement. So the upgrading has to ensure *government* is ranked first. On the basis of its matching both in front and back with the typo, *government* is still ranked first.

For transpositions, one would think that these would be retrieved

just once, on the basis of their identical anagram key value. However, the correct form is retrieved as many times as the TAVs match one of the AAVs, which is equal to the number of TAVs.

Single insertion errors produce four matches: one unigram character value subtraction, one character unigram value (the null-value) to character unigram substitution and two bigram character value to unigram character substitutions. Two-character insertion errors produce one match on the basis of one character bigram subtraction.

The substitution error *governmant produces 14 retrieved CCs, but only three times the correct form, on the basis of the replacements *e/a*, *me/ma* and *en/an*. It follows that two character substitutions will only produce one match.

Total cost in number of queries Our error model is in its essence simple and complete: within the limits imposed by the alphabet and the length of the input string, it allows for anything to happen and is still capable of retrieving the intended string.

This comes at the following cost: let $n(\text{AAV})$ be the number of values in the alphabet, $n(\text{TAV})$ the number of typo-derived anagram values, then

- transpositions are found in 1 query
- deletions in $n(\text{AAV})$ queries
- insertions in $n(\text{TAV})$ queries
- substitutions in $n(\text{TAV}) \times n(\text{AAV})$ queries

which gives us the total number of queries required:

- $1 + n(\text{AAV}) + n(\text{TAV}) + (n(\text{TAV}) \times n(\text{AAV}))$

This then also defines the maximal number of chained lists of correction candidates that may be retrieved. Follows the sequential search through these chained lists for the CCs matching best with the input string. We have seen that for the lexicon with frequency cut-off at 2, this multiplies this maximal number by 1.215 on average. The actual sequential search involves nothing more than looking at each item of the chained list in turn whether it matches the input string either in front or back. The items that do are retained and sent on for upgrading.

We have now given an in-depth description of how the core correction mechanism works. This mechanism is in actual fact applied in several levels or tiers in TISC. Before we describe how this works, we discuss in the next subsection how we in practice limit the LD covered by the core correction mechanism.

3.2.3 Levenshtein distance

The LD that can be covered by our system can be set or limited in two ways.

First, the anagram values that make up the alphabet could include not only the values for character unigrams and bigrams, but also for character trigrams. The values included define the scope of the correction mechanism. We will further discuss this in Subsection 3.3.2.

Secondly, for each correction candidate retrieved, we use a separate subroutine which calculates the LD between the string to be corrected and the candidate. This is required because even with the LD limit imposed by the alphabet, CCs of great LD are retrieved, e.g. *goniometer* for *goverment, with an LD of 6. By discarding the CCs retrieved that have an LD larger than the limit we set at run-time, less plausible candidates are removed and ranking is further improved. The Levenshtein Distance implementation we used throughout this work was provided by Eli Bendersky[1]. This has a 'classical' complexity: $O(m \times n)$ (Navarro, 2001) (p. 47) where n is the length of string 1 and m the length of string 2. As a matter of fact this is wasteful: the implementation tells us what the actual LD is. We only need to know whether or not the LD exceeds the limit we set or not. This can be obtained by Ukkonen's 'diagonal transition algorithm' (Navarro, 2001) (p. 48) which can check in time $O(k^2)$ whether the distance is $\leq k$ or not (Ukkonen, 1985). Future implementations of TISC will use this algorithm instead.

3.2.4 Tiered correction

The core correction mechanism is invoked several times for a particular typo. It is actually applied on three tiers: on the tier of the isolated words, on the tier of the four bigrams formed by the 2-1-2 word window in the input context and, if necessary, on the tier of the input word's two compounding parts. These three tiers of correction together form the full correction mechanism. On the first and third tiers we not only retrieve CCs from the lexicon, but also from the list of word types derived from the input text. This latter step is based on the simple observation that the correct form may well be present elsewhere in the input text, due to the bursty nature of words. This may actually allow TISC to correct words for which the correct form is not even present in the lexicon, as we shall see in Subsection 3.4.4.

The unigram tier consists of two levels. On the first level, we effect unigram correction on the basis of the corpus-derived lexicon, which is equivalent to isolated word correction as performed by most spelling

[1] Available from http://www.merriampark.com/ld.htm

Unigram Tier: Items returned			
Error Type	Typo	Input AV - TAV + alphabet AV = lexicon key AV Actual character(s) represented by the AV are shown between brackets	CCs
Deletion	pro-goverment	198002675145 + 16105100000 (n) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 33554432 ( ) + 16138654432 () = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 17623416832 (p) + 33728516832 (pn) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 19254145824 (r) + 35359245824 (rn) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 16850581551 (o) + 32955681551 (on) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 184528125 (-) + 16105100000 (n) = 213923247020	progovernment
Substitution	pro-goverment	198002675145 - 11592740743 (g) + 27697840743 (ng) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 16850581551 (o) + 32955681551 (on) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 22877577568 (v) + 38982677568 (vn) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 10510100501 (e) + 26615200501 (ne) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 19254145824 (r) + 35359245824 (rn) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 15386239549 (m) + 31491339549 (nm) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 10510100501 (e) + 26615200501 (ne) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 16105100000 (n) + 32210200000 (nn) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 21003416576 (t) + 37108516576 (tn) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 33554432 ( ) + 16138654432 () = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 33554432 ( ) + 16138654432 () = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - () + 16105100000 (n) = 214107775145	pro-government
Substitution	pro-goverment	198002675145 - 17035109676 (o-) + 32955681551 (on) = 213923247020	progovernment
Substitution	pro-goverment	198002675145 - 11777268868 (g-) + 10543654933 () = 196769061210	never promote, promote nerve
Substitution	pro-goverment	198002675145 - 11777268868 (g-) + 27697840743 (ng) = 213923247020	progovernment
Substitution	pro-goverment	198002675145 - 39728159119 (vo) + 19097440758 (ea) = 177371956784	pro-agreement
Unigram Tier: Upgraded ranking			
Typo	CCs	Upgraded count	LD
pro-goverment	pro-government	119	1
pro-goverment	progovernment	3	2
pro-goverment	pro-agreement	1	4
pro-goverment	promote nerve	1	8
Unigram Tier: LD filtered ranking			
Typo	CCs		
pro-goverment	pro-government		
pro-goverment	progovernment		

TABLE 3.4 Correction on the unigram tier. For this particular example, actually only one item retrieved from the lexicon forms a chained list of two anagrams. The first of these, *never promote* does not make it to upgrading: it matches neither front nor back with the input word. The second, *promote nerve*, is not upgraded because its LD is too high, as is the case for *pro-agreement*. Both are discarded in the last step, the LD filtering. In this case, only two CCs are finally retained.

Bigram Tier: Items returned			
Error Type	Typo	Input AV - TAV + alphabet AV = lexicon key AV Actual character(s) represented by the AV are shown between brackets	CCs
Deletion	pro-goverment coalition	327162062029 + 16105100000 (n) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 33554432 ( ) + 16138654432 () = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 17623416832 (p) + 33728516832 (pn) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 19254145824 (r) + 35359245824 (rn) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 16850581551 (o) + 32955681551 (on) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 11592740743 (g) + 27697840743 (ng) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 16850581551 (o) + 32955681551 (on) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 22877577568 (v) + 38982677568 (vn) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 10510100501 (e) + 26615200501 (ne) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 19254145824 (r) + 35359245824 (rn) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 15386239549 (m) + 31491339549 (nm) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 10510100501 (e) + 26615200501 (ne) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 16105100000 (n) + 32210200000 (nn) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 21003416576 (t) + 37108516576 (tn) = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 33554432 ( ) + 16138654432 () = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 33554432 ( ) + 16138654432 () = 343267162029	pro-government coalition
Substitution	pro-goverment coalition	327162062029 - 0 + 16105100000 (n) = 343267162029	pro-government coalition
Substitution	by pro-goverment	233012862146 - 15386239549 (m) + 32016961449 (ri) = 249643584046	over-reporting by
Substitution	the pro-goverment	241716275678 - 22877577568 (v) + 31513517077 (te) = 250352215187	the mentor-protege
Substitution	the pro-goverment	241716275678 - 39728159119 (vo) + 19097440758 (ea) = 221085557317	the pro-agreement
Bigram Tier: Upgraded ranking			
Typo	CCs	Upgraded count	LD
pro-goverment	pro-government	476	1
pro-goverment	pro-agreement	1	4
Bigram Tier: LD filtered ranking			
Typo	CCs		
pro-goverment	pro-government		

TABLE 3.5 Correction on the bigram tier. Given that the four correct bigrams are actually in the lexicon, the list of CCs returned that converge on the correct form, is in actual fact four times the list shown: once for each bigram. After upgrading and LD filtering only the single correct form remains as the CC. This will be compared to the list obtained on the unigram tier. As both tiers are in agreement on this CC, no further correction, i.e. on the compound tier, is done.

error correction systems available. On the second level, we perform unigram correction on the basis of the list of input context derived types and compounding parts, in which the types exceeding a particular frequency threshold are taken to be correct and are made available to the system as an input text derived lexicon. The frequency threshold is set very low: all non-hapaxes are included.

On the bigram tier, TISC performs context-dependent error correction, to some extent. It examines the 4 bigrams contained within a 2-1-2 window around the type in the input text (e.g. the green **bottel* was empty → the **bottel*, green **bottel*, **bottel* was, **bottel* empty). For the 4 bigrams sent consecutively through the correction loop, all the correction candidates retrieved are stored in the same list, reinforcing the evidence found that a particular CC is more likely given the context than the other CCs retrieved. This produces more reliable counts after upgrading.

After correction on the unigram and bigram tiers, the output candidates are compared and if both tiers concur, i.e. the same candidate(s) were returned, they are accepted if they differ from the input type, or rejected (and 'let go') if not. This is illustrated in detail in Table 3.4 for the unigram tier and in Table 3.5 for the bigram tier on the basis of the typo *pro-goverment as in the context: 'by the *pro-goverment political coalition'. All four bigrams derivable from this 2-1-2 window happen to be in the lexicon used. To produce the results in this table, the alphabet contained AVs for character unigrams and bigrams only. The LD limit was set at 3. In the next paragraph we discuss what happens when none of the four bigrams happen to be in the lexicon. As the unigram tier table shows, only two CCs are finally returned. The less desirable variant *progovernment* is not actually returned on the bigram tier. Nevertheless, less plausible CCs are retrieved. The CCs *over-reporting* and *mentor-protege* are not retained: they match neither front or back with the input word. The CC *pro-agreement* is passed on to the upgrading step, but is not actually upgraded because its LD exceeds the limit and is therefore discarded in the final step. The best-first ranked CCs on both unigram and bigram tiers are identical: the output is in agreement and no further processing on the compound tier is necessary.

When no output is returned by the unigram and bigram correction tiers, or the results of these do not concur, the type is further checked on the third tier, that of its substrings, i.e. the compounding parts returned by the compound splitter, which we discuss further in 3.4.2. The compound correction tier treats both LPC and RPC as words in their own right, queries the system for correction candidates in the same way

as on the unigram tier for both parts, i.e. making use of both the lexicon and the input text derived list of word types and frequency information. Finally, the top candidates returned are concatenated and proposed as correction candidates. In the case of *pro-goverment without any of the context bigrams being present in the lexicon, all is as in the unigram correction table presented above. On the bigram tier, however, only the CC *pro-agreement* is retrieved from the lexicon and discarded because of its LD. The bigram tier then produces no CCs, agreement between the unigram level and the bigram level is not possible and the correction continues on the compound tier. This produces the single CC *pro-government*: on the compound tier we retain only the best-ranked CC found. The CC returned by the compound tier is in full agreement with the best-first ranked CC returned on the unigram tier. What happens when no CC is returned on the unigram tier, because the actual correct form to which the typo should be resolved is not at all in the lexicon, we exemplify in 3.4.4.

Resolving ambiguous typos We next consider how the bigram tier helps to resolve the correct ranking of the CCs in the case of ambiguous typos, such as for instance *onjections which might resolve to either *objections* or *injections*. Again, output on the unigram tier is necessarily the same for the typo *onjections, regardless of whether it appears in the context 'most vehement *onjections lodged against' or 'painful intramuscular *onjections received daily'. The contexts are fabricated: we made sure all four bigrams are actually in the lexicon we used. Let it suffice for the unigram tier that the 5 CCs retained in the end are, in that order: *injections, objections, projections, rejections, ejections*. In Table 3.6 we give the results obtained on the bigram tier. Given that for the *injections* context the best-first unigram and bigram CC are identical, the unigram list of CCs is output by the post-correction evaluation module. Given that for the *objections* context the bigram tier's best-first CC is also present in the unigram tier's list of CCs, the bigram tier's best-first CC is output first, followed by the rest of the unigram tier's CCs. We see that given the perfect contexts, both typos are resolved to their proper correct form, achieving perfect best-first ranking.

Handling segmentation errors Due to the fact that the space character is included in the alphabet the outlier CCs *never promote* and its anagram were retrieved on the unigram tier in the above. We handle run-on errors, where two words happen to form a typo because the space dividing them has somehow been lost, on this basis. During processing on the unigram tier we gather CCs returned, if any, on the basis

Typo	CCs	Upgraded count	LD
'painful intramuscular *onjections received daily'			
Bigram Tier: Upgraded ranking			
onjections	injections	210	1
onjections	objections	48	1
onjections	connections	4	2
onjections	conditions	1	4
Bigram Tier: LD filtered ranking			
onjections	injections		
onjections	objections		
onjections	connections		
'most vehement *onjections lodged against'			
Bigram Tier: Upgraded ranking			
onjections	objections	144	1
onjections	projections	16	3
onjections	connections	12	2
onjections	conventions	12	3
onjections	conditions	4	4
Bigram Tier: LD filtered ranking			
onjections	objections		
onjections	projections		
onjections	connections		
onjections	conventions		

TABLE 3.6 Ranking performed by the bigram tier. Same typo, different context.

Corpus	NYT-BNC	ILK-TWC
language	English	Dutch
tokens	1,219,542,274	681,686,340
bigrams (frq.>2)	21,626,223	9,927,378
derived unigrams	797,532	861,604
keys/anagrams	13,421,017	9,000,131

TABLE 3.7 Statistics of NYT and ILK-TWENTE corpus and lexicon.

of the addition of the AV representing the space character in a separate list. Those returned are returned because the actual bigram that can be formed by adding a space character somewhere to the erroneous string is present in the lexicon. This implies that solving run-ons is dependent on the fact of the bigram being available. We do not return all bigrams for all words that happen to constitute two other words present in the dictionary, as by default ISPELL does. The cost of that strategy we demonstrate in Chapter 4.

The other type of segmentation error, split words, we can correct on the basis of watching out for CCs returned on the bigram tier on the basis of subtraction of the AV for the space.

Having word bigrams in the lexicon represents a sizeable overhead, but this is off-set, in part, by the fact that correction of segmentation errors without an exponential explosion of possible edits to consider becomes possible.

3.3 TISC corpus-derived components

3.3.1 The lexicon

A TISC lexicon is derived from a large corpus of tokenized, but otherwise raw text, from which all XML or other mark-up tags have been discarded. We normalise the corpus by replacing all word-external punctuation by a single unique mark, as well as all digits and numbers by another. We apply the rule-based tokenizer we described in Subsection 2.1.3. and use the CMU Statistical Toolkit for deriving a bigram frequency list from the corpus (Clarkson and Rosenfeld, 1997). We discard the tail of the bigram list below a threshold frequency, which we further refer to as the cut-off. This ensures we do not incorporate the bulk of erroneous types present in the corpus. Next the frequency information is discarded and a unigram list is derived from the retained part of the bigram list. We lowercase the unigram list and concatenate the bigram list, the unigram list and the lowercased unigram list, removing any doubles. Adding the lowercased list removes the need for having

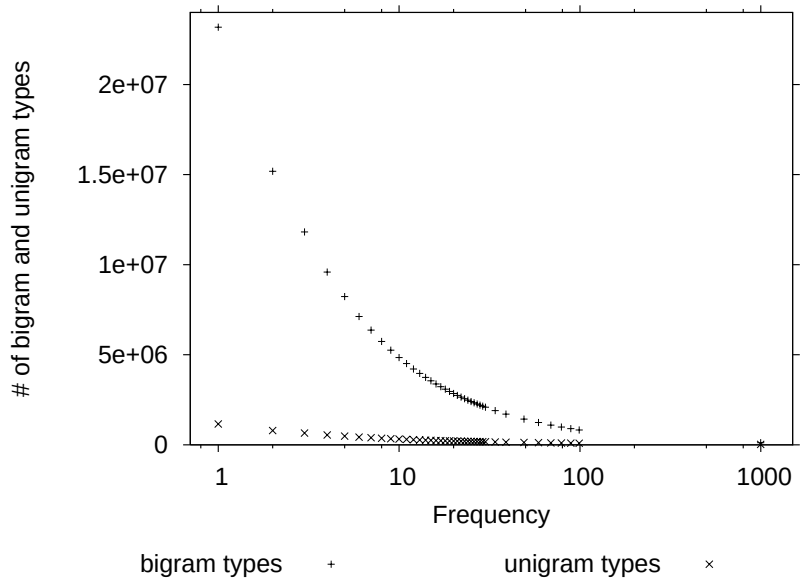

FIGURE 3.2 Plot of the number of bigram and unigram types in the combined NYT-BNC lexicons per frequency cut-off.

character bigram values for lower/uppercase character combinations in the alphabet, which reduces the total number of queries needed. If for the first ranked CC the list contains a matching CC differing only in that its first character is capitalised, the upgrading process should push the latter forwards as the first ranked CC, if it has higher cooccurrence frequency counts.

Finally, we compute the anagram key values for the unigram/bigram list. Together, the anagram keys and their lined-up unigrams or bigrams constitute the lexicon. As the space is regarded as a character in its own right no unigrams in the lexicon can line up with bigrams. Note that the lexicon will contain names and recurrent typos whose frequency exceeds the cut-off. Table 3.7 provides some statistics for an English and a Dutch lexicon, both at cut-off frequency 2, i.e with hapax legomena and dis legomena removed.

In the evaluations we present in Chapter 4, we used the combined NYT and BNC corpora for English. For Dutch we used the combined ILK and TWC corpora.

For English, we combined the NYT and BNC corpora in order to emulate an international English dictionary, which contains the regular spelling variations between British and American English. We did this

because the Reuters RCV1 corpus contains stories sent in from all over the world, in both orthographies. An overview of the bigram list cut-off points and the resulting size of the lexicons obtained in terms of the unigrams and bigrams present is given in Figure 3.2. It can be seen that the lexicon obtained at cut-off 1 (containing 23,182,623 bigram and 1,156,351 unigram types), where only those bigrams occurring just once in the whole corpus are discarded, is huge compared to regular spelling correction systems' dictionaries. Note that discarding word bigram hapaxes is not the same thing as discarding the unigram hapaxes. Both words contained in a bigram hapax may well be present with far greater frequency in other bigrams. However, discarding the bigram hapaxes effectively removes the unigram hapaxes. The fact that unigram hapaxes are also removed may seem undesirable. Nevertheless, it should become clear when we discuss our spelling checking strategy in Section 3.4.4. that they would not have a useful role to play. We have also seen in Section 2.2.6. that in the RCV1 45% of the unigram hapaxes are in fact typos. One does not want to incorporate these if one's goal is to remove as many typos as possible from a large corpus. Only if the system were equipped with an additional trusted dictionary would we be able to fruitfully employ word unigrams such as *crayfishes*, *hyphenations*, *unifications* or *whaleburgers*, all of which are hapaxes in the NYT. This is certainly an option, but one which is not explored in the present work.

The lexicon obtained at the highest cut-off shown here, at frequency 999 (containing 118,272 bigram and 22,099 unigram types), is but small. It represents only a basic vocabulary, though it should be trustworthy to a high degree.

3.3.2 The alphabet

Transformations on the word type to be evaluated are necessary in order to identify correction candidates. These transformations occur on the anagram key of the word type under consideration on the basis of the AVs for the alphabet which is made available to the system. The alphabet used throughout this work, consists of the anagram key values for all character unigrams and character bigrams we want to work with. We here experimented with both a larger and smaller alphabet: with and without character trigram values added to the alphabet. The smaller alphabet contains the unigram values for the apostrophe, dash, *A-Z* and the special characters: à, á, ä, è, é, ë, í, ï, ó, ô, ö, ü, as well as the unigram and bigram values for the regular alphabet *a-z* and the space. Added to this, for the larger alphabet, are the trigram values for *a-z*, i.e. *aaa* to *zzz*. The smaller alphabet contains 498 AVs, the larger

Combination	Count	Corpus examples
LPC count	37	bottled, bottlenose, bottleneck, bottles, ...
RPC count	3	Harbottle, bluebottle, milk-bottle.
LPB count	110	aspirin bottle, beer bottle, big bottle, blue bottle, ...
RPB count	82	bottle and, bottle bank, bottle containing, bottle feed, ...

TABLE 3.8 The cooccurrence counts are obtained by counting the combinations observed in the lexicon which contain a particular word type. Example: 'bottle'.

3,774. The special characters were derived from the Dutch corpus.

Replacing the actual character bigrams and trigrams by their corresponding AVs represents a sizeable reduction of the search space: every possible combination of two (two combinations) or three characters (six combinations) is represented by a single numerical value. As we have seen, this furthermore represents an abstraction away from the actual character string sequence.

3.3.3 The cooccurrence information

From the word bigram and unigram lists we derive cooccurrence information for all the word types present. For each word type we count the number of times it forms the:

- left part of a compound (LPC)
- right part of a compound (RPC)
- left part of a bigram (LPB)
- right part of a bigram (RPB)

The COOC table contains only the counts per word-type, not the actual cooccurring word types. It is derived from the same word bigram list as is used to build the lexicon, i.e. the bigram list retained after a particular frequency cut-off was applied. These four type frequency counts are the only frequency information employed by the system in its current implementation. Table 3.8 gives an impression of the cooccurrence counts, further abbreviated as COOCs, obtained for *bottle*.

3.4 TISC: the implementation

A graphical overview of TISC's architecture is given in Figure 3.3. We have so far discussed blocks A and B: the preprocessing done on a corpus in order to derive the lexicon and the cooccurrence information. We have also in depth discussed block 3, i.e. how correction is effected. We will now walk through blocks 1, 2 and 3 and illustrate the working of the various modules and more specific aspects of TISC on the basis of a single typo in a newspaper article taken from our development set for Dutch. The development set contains an article about the threats posed by legionella, the bacterium responsible for legionnaires'

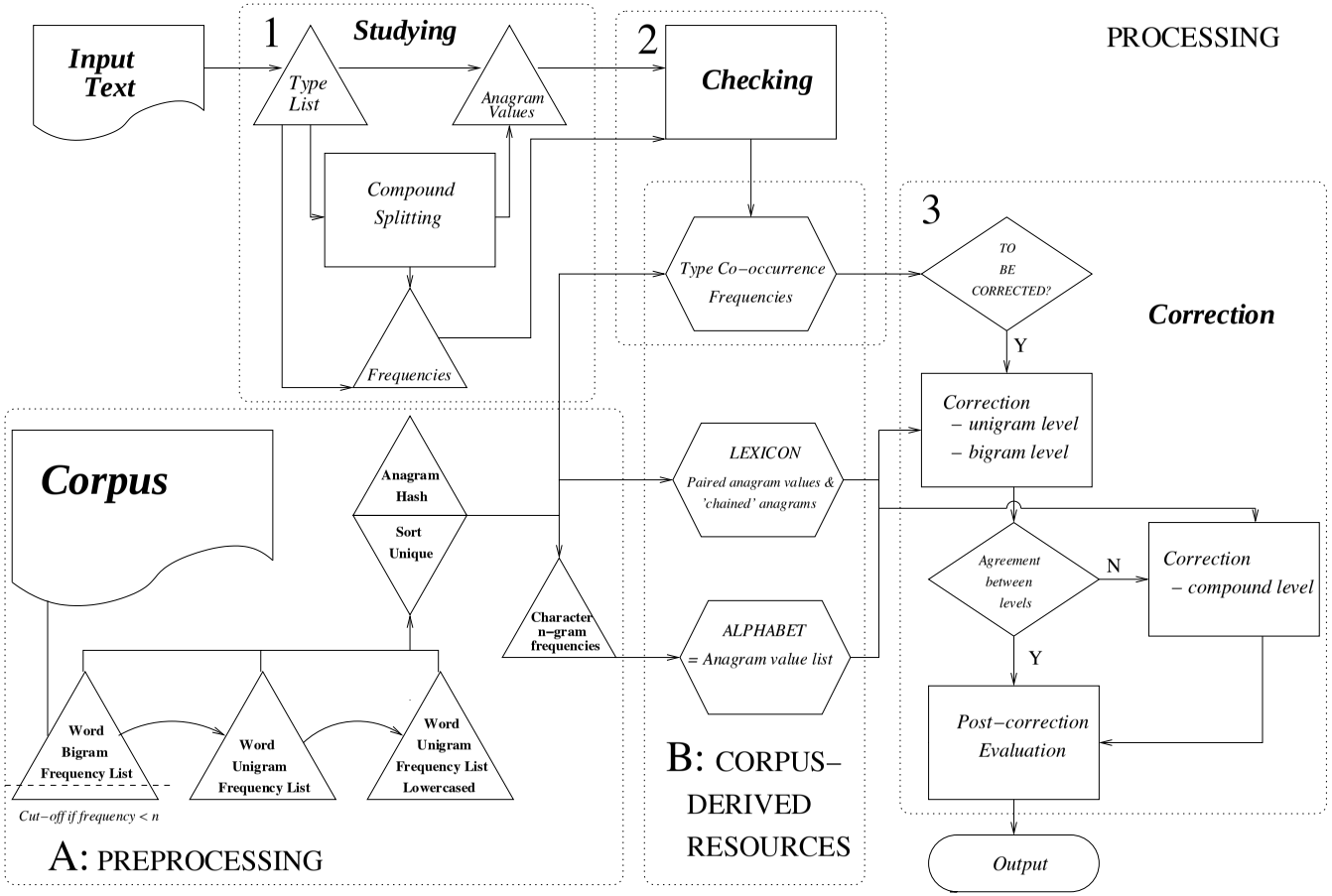

FIGURE 3.3 TISC's architecture. Block A gives an overview of the pre-processing phase which produces TISC's corpus-derived components, shown in block B. To spelling check and correct an input text, TISC first (Block 1) studies the text, i.e. gathers all frequency information about word types in the text and effects compound splitting. In the checking phase (block 2), it uses the cooccurrence information to validate a word or to send it on to the correction phase (Block 3) where correction is effected on the isolated word level and the level of the word in context. In case of non-agreement between these two levels, correction is attempted on the sub-word level on the basis of the compounding information.

disease. The typo is *egionellawaakhonden, which should have read *legionellawaakhonden* [legionella guard dogs - official bodies that guard over legionella outbreak prevention].

This is a rather special case, as the lexicon used during development contains only the types *legionella-bacterie* and two bigrams: '*legionella-bacterie .*' and '*de legionella-bacterie*'. The compound is spelled with a dash, which is not strictly necessary in Dutch, but is allowed. The first bigram is the compound followed by a dot, which denotes any kind of word external punctuation mark. The second has the Dutch definite article *de* (the). Based on this the COOC-table offers only this: 'legionella-bacterie RPB: 1 LPB: 1', which means there are in all 2 bigrams containing the word. Note that the unsupervised method used to build the COOC-file was not able to split the compound and count the LPCs and RPCs because the stand-alone type 'legionella' does not appear in the particular lexicon used.

Before we can walk through the system to see how a text is first preprocessed, then spelling checked and finally corrected, we first need to introduce Zipf Filters, an integral part of the spelling error detection mechanism we employ, and explain how TISC effects compound splitting.

3.4.1 Zipf Filters

Zipf (1935) stated that the frequency of a word is inversely proportional to its length. This implies that we should expect to see more combinations of any given short word, be it in bigrams or as part of compounds, than of longer words. A long compound, e.g. one composed of three or more shorter words, cannot be expected to further compound with very many more words. Short words can be expected to combine in a myriad of ways, be it as part of compounds or of numerous bigrams. We exploit this in what we would like to call the **Zipf Filters** implemented in our prototype. We make the number of expected cooccurrences of a word dependent on the length of the word form. This then allows to detect anomalies in the COOCs for particular word types. We posit a particular amount of times a string or substring is seen as sufficient to conclude the string is likely well-formed as it is highly productive. To this end we take a constant, which is higher for the shorter strings and lower beyond a particular number of characters, divided by the number of characters in the string, or the string's length. We compare the COOCs of a string to be evaluated with the outcome of this calculation and accept the string as being well-formed when the COOCs are higher, reject and thus send on to the correction module, when lower.

While the use of a particular, heuristically set, constant works, it

is rather unsatisfactory in that it is not particularly well founded and in that one intuitively feels this will not work in the same way if one uses a very large rather than just a large corpus. We will revisit this in Subsection 3.5 and there propose a corpus-derived and finer-grained solution.

3.4.2 Compound splitting

Given that a language such as Dutch freely allows for compounding, any text may contain quite a number of previously unseen compounds. Any spelling checking system will therefore require ways of dealing with this. Correct compound splitting is a hard problem, even if the compound is correctly spelled. Many compounds allow for several readings, depending on the dictionary used. We have, e.g., observed the Dutch ISPELL splitting *internetwinkel* (internet shop) into *interne twinkel* (internal twinkle), which is somewhat unlikely, though perfectly valid. Its dictionary was built in 1995 and lacks the word *internet*.

TISC currently proposes a single particular split to be further provided to the checking and correction modules. This was primarily based on reasons of parsimony and may well not be the best possible solution. Furthermore, the implementation currently allows for only a split in a left and right part, even though we are well aware of Oflazer's pertinent criticism of this approach: this cannot work for agglutinative languages whose millions of possible word forms can never be available in a static dictionary (Oflazer, 1993). However, an example of where decomposition in more than two parts would validate an unacceptable compound would be *vluchtvaartmaatschappij for *luchtvaartmaatschappij* (airline company)(Metro, Dutch Edition, 2 March 2005, frontpage). Decomposing the compound into *vlucht* (flight) + *vaart* (navigation) *maatschappij* (company) would validate the three distinct parts and thereby the whole. We follow Daelemans (1987) (p. 61) in assuming that in a single composition, a maximum of two word forms is combined. So our example would necessarily decompose first in *luchtvaart* and *maatschappij* and then in *lucht* and *vaart*. The incorrect *vluchtvaartmaatschappij cannot thus be decomposed, neither *vluchtvaart or *vaartmaatschappij being acceptable.

Our decompounding mechanism operates as follows: While iterating over the input word string to compute its anagram value, TISC repeatedly queries the lexicon to check for the presence of the substring handled so far. If this is successful for the whole string, the substrings, if any, which show the best balance between length and COOCs are stored with their anagram values. If no full parse was possible, the process is repeated from right to left. A decision is then made over both the

left-right and right-left parses and the split deemed most usable on the basis of the compounding parts' COOCs and length are stored for use by the checking and correction modules.

Example We illustrate this with the typo *egionellawaakhonden. The erroneous LPC *egionella does not appear in the lexicon. The trace of the decompounding module gives:
egionellawaakhonden → e
egionellawaakhonden → eg
egionellawaakhonden → egi
egionellawaakhonden → egion

Moving from left to right 'egion' was the longest substring found in the lexicon. As the remainder of the string was not found in the lexicon, the type was re-evaluated by the splitting module from right to left, providing the trace:
egionellawaakhonden → n
egionellawaakhonden → en [and]
egionellawaakhonden → den [pine, fir]
egionellawaakhonden → honden [dogs]
egionellawaakhonden → waakhonden [guard dogs]

The longest potential RPC 'waakhonden' has a COOC of 5, whereas the longest LPC 'egion', just 1 in lowercased form and just 1 in uppercased form (which may indicate it is a rather rare name). We choose the longer, more frequent substring and store this as the RPC, together with its anagram key value. The remainder and associated anagram key value is stored as the LPC.

3.4.3 Studying the input text

Studying the input text amounts to collecting all useful information from the text to be spelling checked. TISC is currently batch-oriented and so has the full text at its service. The input text is first fully analysed: anagram values are added to the type list derived from the text, frequencies of types and their compounding parts are tallied, track is kept of how many times the type was capitalised and recurrent LPC's not in the lexicon are stored for future reference. Table 3.9 exemplifies this process for the single example word *legionella*. The legionella-article contains 33 token occurrences of the type, of which 13 in compounds and 6 in capitalised form. Given the freely occurring form, TISC tallies how often this form occurs in the input text, whether free or as the LPC of compounds, even if it does not occur in the lexicon.

This information will be used by the decompounding, checking and correction modules. A spelling checking system which does not take the input text into account and which does not have the type legionella in

type	frequency
legionella	10
Legionella	5
legionellamasterplan	2
legionellabesmetting	2
legionellabacterie	2
legionellavrij	1
legionellaschoon	1
legionellarisico's	1
legionellaregeling	1
legionellapreventie	1
legionellamaatregelen	1
legionelladoden	1
legionelladiagnose	1
legionella-uitbraak	1
legionella-onveilig	1
legionella-notitie	1
Legionellapreventie	1

TABLE 3.9 The type 'legionella' and its compounds in the development set article.

its dictionary (as is the case for both the Dutch ISPELL and MPT), will on this article produce 33 precision errors (token count), due to this single missing word. TISC will produce none, given that in default of too low COOCs, it accepts the 33 input text occurrences, provided the COOCs or input text occurrences of the second part of all 'legionella'-compounds provide enough evidence to accept them.

EXAMPLES	bizarre	bizzare
COMPOUND - LEFT	5	0
COMPOUND - RIGHT	0	0
BIGRAM - LEFT	712	5
BIGRAM - RIGHT	252	4
COOC	969	9
ZIPF FILTER THRESHOLD	14	14
DECISION	let go	sent on

TABLE 3.10 Overview of COOCs for *bizarre* and *bizzare, Zipf Filter threshold and decision taken accordingly.

3.4.4 Checking

We employ a simple but effective heuristic as the first step in deciding whether a word type is erroneous or not, which is its frequency in the input text. Word types occurring n times or more, are simply regarded as being correct and not further evaluated. The threshold can be set at run time and is set according to the length of the input text, higher the longer the text.

All the types whose input text frequency is lower than the threshold are sent to the spelling checking module. We do not have a trusted dictionary, so we cannot content ourselves with simply checking whether a type is present in the dictionary or not. Instead, we query the cooccurrence information table to see whether the particular type's COOCs conform to our expectation of how many times a type of the given length should have been incorporated in the lexicon, i.e. the expectancy level or threshold set by the Zipf Filter. If the type's COOCs conform, the type is not further evaluated, which we will refer to as 'let go'. An example is given in Table 3.10. If the type's COOCs do not conform, the COOCs for its LPC and the RPC are evaluated against the Zipf Filter thresholds. We do not, at this stage, want to risk to lose too many of the erroneous types, so the level of expectancy is set rather high. We simultaneously check whether perhaps the lexicon contains possible bigrams based on the type's anagram key value with the value for a space added. All the types which did not conform to the expected levels or were found to be present with an additional space, are further evaluated.

Further checking is a dual process. On the one hand we employ the set of rules we present next. On the other hand we invoke the unigram correction tier and see if that returns likely CCs. The additional rules are:

- extra-space cases: If it turns out the lexicon contains only the inverted form with the added space (e.g. 'koffiebekertje' [coffee cup]: not in the lexicon, but 'bekertje koffie' [cup of coffee] is present), we accept the form as being correct, the rest are further evaluated. This happens regardless of whether both forms actually mean the same thing or not: it is assumed the orthography remains identical, even if the semantics differ.
- we check whether perhaps the LPC was seen in various other input text compounds or whether the RPC was perhaps seen as a word in its own right with a given frequency in the input text, the other part's COOCs conforming. Again those passing this test are let go. See the example of *legionella*, above.
- we check whether perhaps the COOCs for the LPC with first or all

characters upper-cased conform to expectance.

- if the input type contains a dash, we check whether the COOCs for the type without the dash conform. Or perhaps whether the type without the dash but with an extra space is present in the lexicon and its COOCs conform.
- finally we check those forms for which the cooccurrence table contains no information at all. This would typically be the case for previously unseen compounds, e.g. neologisms. If the COOCs for their LPC and RPC exceed a high expectancy threshold, these are let go too.

For the types that were not let go by one of these rules, we evaluate the CCs returned by the unigram correction tier. The most likely situation here is that the input word is returned as the best-ranked CC as it occurs in the lexicon. We therefore check whether or not its COOCs are higher than the COOCs of the second-best ranked CC. If they are, we let the type go. Else we check if perhaps the second-best ranked CC is perhaps a more frequent morphological variant of an otherwise correct input type. We therefore re-employ the 'the-one-fits-in-the-other' upgrading rules. If one of these applies, we check the COOCs for the shorter word form and if these conform to the Zipf Filter threshold, we let the type go. An example would be a rarer adverb such as *tempestuously* with COOCs: 2 in the NYT-BNC lexicon at cut-off frequency 2, where the adjective *tempestuous* has COOCs: 82. It will be clear that language-specific morphological production rules would be in order here. Nevertheless, the implementation has none.

In the cases where the unigram tier correction returned no CCs, we take a close look at the input text derived type list to see if perhaps enough evidence can be found there to validate the input type. This would typically be the case for text specific compounds. Examples here would be the correctly spelled *legionellaschoon* [free of legionella], *legionellamasterplan* [legionella masterplan], *legionella-notitie*[2] [white paper on legionella] from our development article.

All the types that could not be validated and let go, are sent on to the full correction module.

To return to our development article about *legionella*: The type *egionellawaakhonden has no COOCs, which causes the type to be evaluated on the basis of the COOCs for the LPC **egionella* and RPC *waakhon-*

[2] Three more Googlewhackblatts (on 22-06-2005), all referring to one document on the web (URL: http://www.belproject.nl/default.asp?mid=2&nid=17), which identifies our Metro-article as its source: Metro, Tuesday 3 june 2003, p 1, 12-13. Our typo *egionellawaakhonden has there been corrected and constitutes a fourth Googlewhackblatt.

den, after we have ascertained the type is not present as a bigram in the lexicon (unigram AV + AV for a single space). The Zipf Filter for an RPC-type of length greater than 4 was set at 100 / length of string (LPC: 100 / 9 = 11, RPC: 100 / 10 = 10), which for the RPC means that at least 10 occurrences were required, in this phase. Neither part fulfils this requirement. No other reasons to let go the type are found in further checking. It is therefore sent on to the full correction module.

3.4.5 Correction

A sufficiently high frequency in the input text of the correct form for an incorrect compounding part may enable the system to correct the error even if the correct form is not present in the lexicon. So, for our example *egionellawaakhonden no CCs are found on the unigram tier, nor on the bigram tier. Neither does the input text contain the correct compound form. The type is therefore also sent through the correction module at the tier of the compounding parts. We present the trace produced for the LPC, after upgrading.

egionella → legionella 55
egionella → sigonella 15 [Sigonella: place in Italy]
egionella → algenbloei 11 [algae bloom]

We see that the CC *legionella*, which is derived from the input text as it is not available in the lexicon, is ranked before *sigonella*, which is in the lexicon and not in the input text. The upgraded retrieval count of 55 for *legionella* is due to the fact that this input text derived CC fulfils three upgrading rules compared with the LPC. The second CC only fulfils the first-or-last-characters-matching rule. The third CC, *algenbloei*, looks wide off the mark, but is actually a near anagram, having only an extra *b*.

The RPC trace is then:

waakhonden → waakhonden 115 [guard dogs]
waakhonden → waakhond 10 [guard dog]
waakhonden → onwaarheden 6 [falsehoods]

The RPC is in the lexicon and is returned as the best-first ranked CC, followed by its singular form, followed by another implausible candidate (a substitution of 'k' by 're' in terms of the anagram key value).

Finally, both best-first ranked compounding parts are concatenated to 'legionellawaakhonden', which is then proposed as the correction candidate for the compounding tier.

Post-correction evaluation After correction on the tiers that were actually invoked, the post-correction evaluation module decides on what is to be output and in what order. We have seen that if the unigram and bigram tiers concur, the best-first ranked bigram tier CC is

output first, followed by the rest of the unigram tier CCs. If there are no CCs on the bigram tier, it is first evaluated if perhaps a run-on was detected. If so, the bigram CC returned for the unigram which was sent through correction is output first, followed by the rest of the unigram tier CCs. Else the output of the compounding parts' tier is compared to the unigram tier CCs and if in agreement, output first. In default of CCs on the unigram tier, as was the case for *egionellawaakhonden, the single CC obtained on the tier of the compounding parts is output.

We also employ two final resort back-off rules, which are specific for English and can be invoked at run time. The first rule says that if the input type was a compound written with a dash and if the unigram tier CCs contain the same form written as a single word **and** with a space instead of the dash, then nothing is output. The second rule says that if the input type differs from the single CC by only the anagram value for the difference between *s* and *z* (i.e. *-ise/-ize*, *-isation/ization* variation) no output is generated either. These rules preclude a great number of precision errors. We will say more on language-specific allowable variation at the beginning of Chapter 4.

3.5 Refinements

In this section we address the two main problems affecting TISC as it was described above.

In the next chapter we see that TISC's performance in terms of recall degrades when the lexicon cut-off is decreased. On the other hand, we see that this boosts precision. The decline in recall is due to more and more noise in the form of recurrent typos being incorporated in the lexicon if we build it the way we described in Subsection 3.3.1. In Subsection 3.5.1. we present an unsupervised method for reducing this level of noise.

The second problem is related to our Zipf Filter settings. So far, these were set manually, by positing a constant from which the threshold is derived. In Subsection 3.5.3, we propose ways of setting the threshold on the basis of corpus-derived values.

3.5.1 Corpus-Induced Corpus Clean-up

A next logical step is to apply our algorithm to the lexicon and derived cooccurrences file themselves in order to overcome the vexing problem of not being able to correct very common spelling errors. The way we have up to now dealt with things, these errors quite simply occur too frequently in the corpora employed, ending up in the lexicons and eluding our Zipf Filters.

We built a simple lexicon correction module we will further refer to

as CICCL for Corpus-Induced Corpus Clean-up. It uses as its input a particular lexicon, its associated cooccurrences file and the alphabet. While reading in the cooccurrences file we tally the right parts of compounds and store the top 30 for the RPCs. This gives us in effect a corpus-derived list of the most common suffixes in English.

Arbitrarily limiting ourselves to the 10,000 most frequently used words in the lexicon we then run the TISC unigram correction module to retrieve all CCs present in the lexicon itself. This process too was restricted; we allowed retrieval only of variants within LD 1 and 2, but LD 2 restricted to transpositions: if the anagram key value of two character strings is the same and the LD is two, the less frequent string contains a transposition. By allowing for LD 2 deletions, insertions or substitutions we found we retrieved too many valid words and these, of course, we aim not to lose.

CICCL is not fully language-independent. We found we had to model a number of specifically English constraints for ensuring we did not retrieve too many valid words when we applied CICCL to the English lexicon. After retrieval we apply the upgrading process as we do in TISC. Here, however, the process is modified to up- or downgrade. Downgrading is subtracting from the number of times an item was seen. It reduces the count to zero or less. Only word forms obtaining a count higher than zero are eventually returned. Word forms retrieved that differ only in their last characters by a value present in the RPC top-30 value list are probably simple morphological variants and downgraded. Also, when a variant is encountered which differs only in the value for a space, but the input word's list of CCs also contains a variant differing only in a dash, we downgrade. Variants differing only in first character capitalisation are likewise downgraded. The *s-z* variants are downgraded. Finally, all variants that amassed a count greater than zero and whose COOCs are lower than the Zipf Filter threshold set at 10, are passed through to the following filters: we modelled the end *e-y* variation in e.g. *reasonable-reasonably* and the end *s-d-r* variation in e.g. *changes-changed-changer* as well as allowing for any final *s* to appear as *'s* too as in *dogs-dog's*. All variants that pass through these filters are taken to be unacceptable and output, paired to their correct form.

The system is lossy: we lost *blooping* through *blooming* and *notionally* through *nationally*. Table 3.11 shows the 21 variants for *government* that were retained by CICCL. Just 12 of these also figure in the list of Reuters RCV1 variants we presented in Chapter 1.

The total list retrieved on the basis of the 10,000 high-frequency words we let CICCL examine, contained 10,424 items. This means that we found just over one variant per word inspected, on average. In order

CICCL-collected variants			
fovernment	govenment	govern ment	governnment
gGovernment	govenrment	governement	govternment
geovernment	goverment	governent	gvernment
goernment	goverments	government'	gvoernment
gopvernment	govermment	governmetn	
gov ernment	govermnent	governmment	

TABLE 3.11 Variants for *government* retrieved by CICCL from a NYT-BNC lexicon.

CICCL-list and Ispell dictionary overlap			
economias	employeres	examinee	expecially
economizes	employess	exampled	expedience
edication	encouragd	exchangers	expensed
eduction	endorsee	excising	explictly
efficency	enduing	exected	exployed
electroic	enraging	exection	expresssion
electronis	enterprisse	exercise'	extention
eliminate-	entourages	exigible	extraordinarly
emphases	equites	exorcises	

TABLE 3.12 Overlap between CICCL list and ISPELL expanded dictionaries: excerpt: words beginning on *e*.

to check the accuracy of CICCL, we looked at the overlap between our list and the expanded ISPELL dictionary[3]. We indeed found that 809 words appear in this ISPELL dictionary and so are supposedly correct words we lose using CICCL. However, the ISPELL dictionaries are not without their own problems, as was already pointed out by Zobel and Dart (1995). Table 3.12 lists the overlap in words beginning on 'e'. We see the list contains correct but rather more infrequently used words, as well as unacceptable forms, typos derived from a corpus, probably. We can only issue a note of warning to users of the ISPELL extra large dictionaries.

This leaves us to conclude that CICCL has procured a reasonably accurate list of recurrent errors present in the lexicon. We further refer to the CICCL retained variants as the *acquired* errors. We might have built new lexicons, removing these acquired errors from the initial bigram list. This would enhance performance. Another option was to

[3] Available from http://wordlist.sourceforge.net/

simply let TISC have access to the list and to use these acquired errors to perform absolute correction (Pollock and Zamora, 1984). We made the necessary changes to TISC so that the list can be read in at runtime and put to direct use, while retaining the uncleaned prior lexicons. The idea behind this is that by giving TISC access to a list of known errors, unnecessary replication of work can be avoided. The algorithm has identified errors, so these are retained for further use: if a known error is encountered in a new text to be spelling checked, appropriate steps are taken: either the spelling detection system is bypassed or the correction module is bypassed. The former happens when CICCL returned only a single possible correction candidate, in which case the error is replaced by this CICCL candidate without further ado. When CICCL returned more than 1 correction candidate, we perform context-sensitive absolute correction: the error is passed on to the correction module, where the bigram correction level will propose its specific ranking on the basis of the input context.

Note that to all intents and purposes the Reuters RCV1 error list would be an ideal substitute for or supplement to the acquired errors list obtained by CICCL. As the Reuters RCV1 error list forms the basis for our English evaluation sets this option was not open to us in the present work.

3.5.2 Zipf Filters revisited

As we have seen in Subsection 3.4.1., manually setting the constant by which the Zipf Filter threshold is set does not provide an intuitively satisfactory solution in that it does not adapt to the size of the initial corpus used. So, for a relatively smaller corpus, as the one used for Dutch, the level set may be too high, with consequent deterioration of performance.

Corpus-Induced Zipf Filter thresholds A solution to this presented itself when we studied the levels we had set. We had always used a higher constant for smaller words, less than five characters long, than for longer words. This is fully in line with Zipf's law, which states that shorter words will occur more often. We reasoned that our constant must be akin to some sort of average level of use, or productivity of a word. If this were calculated on the actual figures of productivity as present in the COOC-list, we would have a more informed, corpus-induced level of expectancy. This would also dynamically adapt itself to the levels as observed in a smaller or larger corpus. And this we would be able to do in a finer-grained way than the rather arbitrary boundary between less characters than 5, or more. This can be calculated and set per actual word-length.

We changed the implementation in such a way that at start-up time, for all words in the COOC-list, per word-length, it would be possible to calculate the average, the median or a percentile of the number of occurrences actually observed. These values are then used to set the Zipf Filters. This introduces a new parameter, with which we can now experiment to see which is the most performant.

In Table 3.13 we contrast the manually set values with those obtained from the corpus, for the NYT-BNC lexicon at cut-off frequency 2. The values for 'manual' are our constant values: the integer value of 1,000 divided by the word length for words smaller than 5 characters and 100 for words 5 character or more long.

We see that the mean values degrade more gracefully than the percentiles, which very quickly diminish to very low values. With these very low values more word forms will meet the COOC threshold and be validated. From this we can predict that with lower percentiles, we will obtain lower recall and precision scores. The mean scores show a mix of both higher and lower values than our manually set threshold levels. We will therefore also see what happens when we either raise or lower these thresholds, by doubling or halving or otherwise manipulating the mean values.

3.6 Summary

We have in the above outlined our spelling detection and correction mechanisms. We have introduced the notion of anagram-key based hashing, which allows for quick retrieval of correction candidates within the limits specified by both the alphabet and the Levenshtein distance limit imposed. We have shown how having word unigrams as well as word bigrams in the lexicon allows for the correction process to be applied not only to isolated words but also to words within their context. We have shown how this facilitates best-first ranking. We have illustrated how we handle compounds and when spelling correction on the tier of the compounding parts is effected. The resources used by TISC are derived from corpora in a completely unsupervised manner. This necessitates more informed typo detection strategies than simple dictionary look-up as performed by most spelling error detection systems. We have introduced the notion of Zipf Filters, thresholds set manually or derived from the corpus-derived lexicon and we have described how these help us to distinguish between correct and erroneous word forms. We have extensively used examples to illustrate how the various modules act and interact. We have finally shown how the core correction mechanism can be applied to the TISC lexicons themselves to filter

Word length	Manual	Mean	90%	70%	Median	30%	10%
1	1,000	11,586	22,584	5,161	3,423	72	6
2	500	1,223	1,040	240	92	23	4
3	333	192	162	35	10	4	1
4	250	110	83	19	7	3	1
5	20	59	45	9	4	2	1
6	16	42	37	6	3	2	1
7	14	39	36	6	2	1	1
8	12	33	32	5	2	1	1
9	11	29	30	5	2	1	1
10	10	25	27	5	2	1	1
11	9	20	22	4	2	2	1
12	8	16	19	4	2	2	1
13	7	13	16	4	2	2	1
14	7	9	12	3	2	2	1
15	6	6	10	3	2	2	1
16	6	5	8	2	2	2	1
17	5	4	7	2	2	2	1
18	5	4	6	2	2	2	1
19	5	3	5	2	2	2	1
20	5	3	5	2	2	1	1
21	4	2	4	2	2	1	1
22	4	2	4	2	2	1	1
23	4	2	3	2	2	1	1
24	4	2	3	2	2	1	1
25	4	2	2	2	2	1	1
26	3	1	2	2	2	1	1
27	3	1	2	2	2	1	1
28	3	2	2	2	2	1	1
29	3	1	2	2	2	1	1
30	3	1	2	2	2	1	1

TABLE 3.13 Overview of manually set threshold and automatically derived mean and median values.

highly recurrent typos from them. The errors acquired in this way then form the basis for context-sensitive absolute spelling correction.

In the next chapter, we extensively test TISC's performance first on its own, then in comparison with the state-of-the-art systems available today. We test performance for two languages, English and Dutch. We test the systems' correction capabilities on lists containing only real-world typos. The systems' error detection and correction capabilities are then tested on real texts, containing typos to greater and lesser extent, the latter arguably reflecting typos' natural distribution.

4

Evaluation

In the previous chapter we have described our proposal for a spelling error detection and correction system. In this Chapter we evaluate the system's performance, both on its own terms and in comparison to other spelling error detection and correction systems. We first introduce the metrics we use in these evaluations. Next we discuss some preliminaries such as the actual types of errors we do and do not evaluate on, how we score results and the terminology we employ. We describe the evaluation sets we derived from the Reuters RCV1 corpus and present a range of evaluations on English in which we gauge TISC's performance under varying conditions. We then compare the performance results obtained by ISPELL, ASPELL, the Microsoft Proofing Tools (MPT) and TISC. This is followed by qualitatively different evaluations on Dutch, where the focus is more on the systems' performances in light of whether the typos are presented within more or less context. To this aim we compiled an evaluation benchmark set of full newspaper articles containing one or more typos as they occurred in the printed edition of the Dutch Metro. We again compare TISC's performance on the Dutch benchmark set to that of ISPELL and MPT.

4.1 Evaluation metrics

Figure 4.1 gives a graphic representation of the tasks involved in spelling error detection and correction. The large box represents the set of word strings in a text or language. The typically much larger, left portion depicts the correct or acceptable word forms, the smaller portion to the right the incorrect or unacceptable word forms. These are split by a dashed line, representing the fact that the boundary between these two categories is not always razor-sharp, what is and what is not correct depending on the definition used. Words to the left of the boundary are non-target items for spelling error correction, the

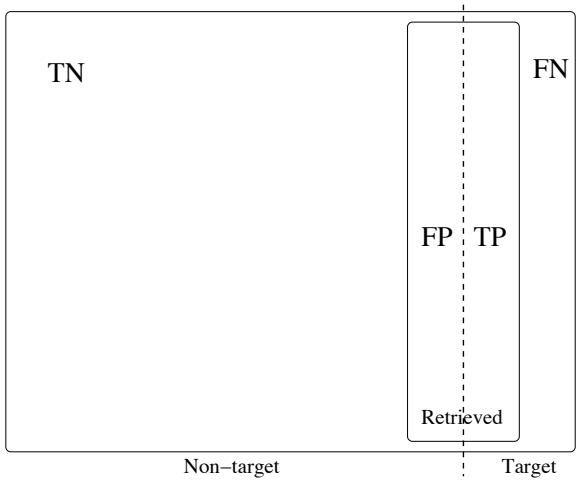

FIGURE 4.1 Schematic representation of the task faced by a spelling error detection and correction system.

	Target	Non-target	
Selected	TP	FP	
Not selected	FN	TN	
Totals	P	N	TOTAL

TABLE 4.1 Confusion matrix. P = positive, N = negative T = true, F = false.

words to the right form the target. The diagram therefore describes the problem of distinguishing between correct words (true negatives or TN) and incorrect words (the target). The system selects a set of words of which it assumes that they are incorrect (the selected set). The false positives or FP are those retrieved that are in fact correct word forms. The part of the target which was not retrieved by the system forms the false negative or FN set. The intersection between words retrieved and the target set defines the set of incorrect words correctly identified as such and corrected (true positives or TP). The aim of any spelling error detection and correction system will be to maximize the overlap between the target and retrieved sets, achieving perfection when this is 100%.

The interrelations between true and false positives and negatives are conventionally represented in a confusion matrix (often referred to as a contingency table). This is shown in Table 4.1. From the confusion matrix many metrics can be derived.

From the TP, FN and FP we can derive recall and precision (van Rijsbergen, 1975), as follows (Manning and Schütze, 1999) (p. 268-269):

- Recall = $\boldsymbol{R} = \frac{TP}{TP+FN}$
- Precision = $\boldsymbol{P} = \frac{TP}{TP+FP}$

The harmonic mean of $\boldsymbol{R}$ and $\boldsymbol{P}$, the simplified F measure, $\boldsymbol{F}$, is given by:

- $\boldsymbol{F} = \frac{\boldsymbol{2PR}}{\boldsymbol{R+P}}$

We fully motivate the choice for these metrics in Chapter 5.

4.2 Preliminaries

We evaluate TISC in various ways, using as the evaluation metrics recall on its own and recall and precision as combined in the F measure. We first evaluate TISC on its own. Both English and Dutch versions are then compared with three state-of-the-art-systems available today: ISPELL (version 3.2.06), its successor ASPELL (version 0.50.3) and the Microsoft Proofing tools (Microsoft Office Word 2003 (11.5604.5606), Microsoft Office Professional Edition 2003), henceforth abbreviated as MPT.

In spelling correction there is an intricate interplay between various factors. First, there is the error model available to the system: what types of errors is the system equipped to deal with? Next, there is its dictionary: what does it contain and how many items does it contain? Third, is it or is it not equipped with a language model? In what follows we try to dissect this interplay by presenting experimental results where as far as possible these factors are analysed, first in isolation, then in combination.

4.2.1 Error correction evaluated on error lists

The first series of tests is run on error lists and allows us to present results on the limited task of correcting typos under varying conditions:

- results of experiments with only a dummy dictionary, i.e. the lexicon contains only the correct forms for the incorrect ones presented in the test material
- results obtained when TISC's lexicon can be trusted, i.e. when it is given ISPELL's expanded dictionary
- results obtained when TISC's lexicon is built in a supervised way, i.e. when the corpus-derived bigram list is filtered on the basis of ISPELL's expanded dictionary

- results obtained with TISC's unsupervised, corpus-derived, noisy lexicons
- results after adding the list of acquired errors produced in an unsupervised manner by Corpus-Induced Corpus Clean-up (CICCL).

Performance on the error lists is measured as correction recall: in these tests, a correction candidate is considered correct if the correction as defined in the evaluation files is found among the correction candidates or CCs returned. What constitutes the correct form for a particular typo correction is determined by the typo's context as observed in the Reuters RCV1 corpus. We also report the best-first ranking scores. Since we aim at ultimately obtaining automatic correction in batch mode, we deem other n-best scores largely irrelevant: an automatic system would only use the best-first ranked correction candidate for replacing an error. However, since TISC's output is limited to the 5-best and some of the scores reported in the literature are also based on the 5-best, we list these too when necessary. Scores for these tests are obtained on **types**, not on tokens: in a list of errors, each error occurs only once.

4.2.2 Error detection and correction evaluated on typos in context

The second series of experiments allows us to present results on the full task of not only correcting typos, but first detecting them: to this end the typos are presented within their original real-world contexts. We again vary the conditions:

- we vary the Levenshtein distance (LD) limit we set in TISC
- we vary the amount of context surrounding the typos
- we determine what the contribution is to performance of the refinements we discussed in Chapter 3, Section 3.5, i.e. the contribution of CICCL and the automatic Zipf Filter threshold settings.

Performance for the full task is measured as **recall** and **precision**, resulting in their harmonic mean, the **F-score**. (van Rijsbergen, 1975). Scores are on **tokens**, not on types: in text, errors may recur and this needs to be measured.

4.2.3 Evaluation terminology

In order to avoid confusion, we specify more closely what is meant when we talk about recall.

Some of the tests we run involve adding all the correct word forms for the typos to be corrected to the dictionaries or lexicons used. This allows for measuring the upper bound on correction attainable by a

particular system to be measured; it effectively removes the effect of dictionary shortcomings on recall. We therefore refer to this as the **upper bound recall**, henceforth abbreviated as UBR.

The recall achieved without ensuring that all the correct forms are in the dictionary or lexicon, we then refer to as **true recall**, abbreviated as TR. By extension, we also talk about **true scores**.

When we measure recall without taking into account the ranking of the CCs we call this **overall recall** or OR. Overall recall may be reported for the upper bounds, i.e. when it is ensured that the dictionary contains all the correct words for the typos to be corrected, or for the true scores, when this is not the case. We need to distinguish between overall recall obtained on lists of typos and overall recall obtained when typos are presented within context. In tests on lists of typos, each typo represents one word type. The score for true positives is therefore increased by one: we measure on types. In tests on text containing typos, the same typo may recur. The score for true positives is therefore increased by the typo's token frequency in the input text: we measure on tokens.

When we focus on best-first ranking of the candidates and measure those that effectively are ranked with the desired candidate given the context presented first, we measure **best-first ranking recall**. On occasion we also report scores on rank 5, however. So we abbreviate best-first ranking recall as RR1 for recall on rank 1 and RR5 for recall on rank 5. The same distinction as in overall recall applies for tests on lists versus tests on running text.

4.2.4 Adverse effect on precision by allowed variation

Both English and Dutch allow for a certain amount of variation in their orthography. Non-exhaustively, for English, types of allowable variation would be: harbour - harbor, center - centre, defenceman - defenseman and even: feminity - femininity. For English a lot of this variation is due to the divergence between British and American English. No codification has been imposed. For Dutch, up to 1995, examples would have been: chronisch - kronisch [chronic], theorie - teorie [theory], examen - eksamen [exam], cultuur - kultuur [culture]. For Dutch, what is and what is not allowed has been codified, by law, and is officially published as the '*Woordenlijst Nederlandse Taal*' (Word list of the Dutch Language), better known as '*het Groene Boekje*' (the Green Book) (Woordenlijst Nederlandse Taal, 1995). Incidentally, *het Groene Boekje* forms the basis for both the Dutch ISPELL[1] and MPT (van den Heuvel, 2003) dictionaries. The spelling change adopted in 1995 disallowed most of the

[1] http://fmg-www.cs.ucla.edu/fmg-members/geoff/ispell-dictionaries.html

variations we listed above and which were permitted before. A revision of the 1995 spelling changes is currently underway: decisions taken ten years ago about the spelling rules for linking consonants in compounds are again under review.

As we are concerned only with word variants that really may obstruct communication, we have largely ignored these matters. Wishing to keep TISC as language-independent as possible, we have striven to incorporate as few language specific rules as possible. However, the *s-z* variation and compound variation in English we found we could not ignore as they had a heavy toll on precision. We implemented a switch that tells TISC to not pass on the results of the correction module in those cases where the only correction candidates returned by the correction module differ from the input word only by the anagram value difference between *s* and *z*. We did likewise for the space/dash/nospace variation. This switch was turned on for the English evaluations, off for Dutch.

In Dutch, diacritics are also often used to put extra emphasis on words, so *ongeveer [about, as in: 'She is about 20']* appears as *ongevéér* in the Dutch benchmark evaluation set and TISC returns the normal orthography as the correction candidate, which is then counted as a precision error. To remedy this, this should be modelled in the Dutch version. This we have not done either, since we wished to find out what can be achieved using next to no language specific knowledge. Adding more language specific filters would be relatively easy to do, would to a high degree be learnable from the specific language's lexicon and is probably desirable seeing the adverse effect not doing it has on the precision of the system. Nonetheless, we did not add more.

We do not evaluate on split words, e.g. 'ingredients of *alcoho *lic beverages' for *ingredients of alcoholic beverages*. Our implementation currently does not deal with split words, although TISC in principle is able to and we have experimented with it in the past. However, split words present a difficulty in accounting: how should they be counted: as 2 errors constituting 2 problems to be solved or as one correction effected? We do not know and have not made up our minds about this.

4.3 Evaluation on English

4.3.1 Evaluation: the test sets

Error lists

The Reuters RCV1 error list we selected randomly and manually provided with corrections was 12,225 items long. The error type breakdown of this list was presented in Chapter 2, Table 2.3. From this list we re-

Reuters RCV1 benchmark	
sentences	2,736
tokens	95,104
types	16,000
type/token ratio	16.82%
errors	3,022
error/type ratio	18.89%
error/token ratio	3.18%

TABLE 4.2 Statistics of Reuters RCV1 benchmark evaluation file.

moved the 53 split words and 134 typos for which we could not decide on what their correction should be. This left us 12,038 typo/correction pairs for evaluation purposes. We randomly chose 2,000 items from this list, 10 times. Overlap between the sets was allowed. This set we refer to as **10x2K**. We first report results obtained on this set. Next we present results on the full 12,038 error list. To this list we further refer as **12K**.

For all sets, we present results obtained on the errors in isolation and on the errors within a 2-1-2 context window. The input contexts employed in these tests were the first context for a particular error encountered in the Reuters corpus.

Typos in context: benchmark evaluation set

Apart from the 12,038 typos, selected randomly from the 33,488 Reuters RCV1 typo list and provided with their correction(s), we selected another 3,300. This too was done randomly, but this time we allowed for the same error to be selected several times: more highly recurrent errors had a higher probability of being reselected. For these errors we extracted the context from the tokenized corpus, the context in principle being limited to the sentence the error occurred in. Also, as regards context, the selection was made randomly from all available contexts. The 3,300 contexts were proofread manually and all further errors encountered marked as such. Over 300 contexts had to be discarded: they did not provide running text, but rather captions from tables and suchlike. The greater part of these 300 were abbreviations rather than typographical errors proper. For evaluation purposes, we finally provided all the errors identified with the proper correction as dictated by the context. In this way, we obtained a benchmark set of English spelling errors of all types, in all 3,022 errors within 2,736 context sentences (or short paragraphs, when the initial sentence tokenization had failed). Further statistics are provided in Table 4.2. Table 4.3 lists

Not evaluated on:	
Category	Total
repeated words	13
missing words	2
confusables	66
split words	22

TABLE 4.3 Error type breakdown of the Reuters RCV1 benchmark evaluation set: error types not evaluated on.

the error type breakdown of the benchmark set for the types of errors present in the Reuters RCV1 benchmark set we do not evaluate on, in all 103 errors. As can be seen, the benchmark set contains confusables and split words, too, error types not dealt with in the present work and further left out of the accounting. Table 4.4 lists the error type breakdown of the benchmark set for the types of errors present in the Reuters RCV1 benchmark set we do evaluate on, in all 2,919 typos. It can be seen that the error type distribution in the benchmark set is in line with the statistics of the full error list we presented in Chapter 2, Table 2.3.

4.3.2 Evaluation: Recall on error lists

Test 1: 10x2K: The lexicon contains only the correct words

The first test allows us to determine the upper bound on what the spelling correction mechanism can possibly achieve: given the range of spelling errors encountered in real data and given that there is no (or only very limited) confusion possible with other non-target words in its dictionary, how many can the system correctly resolve to their correct form? In this test, we only activated TISC's correction mechanism on the unigram tier; the bigram and compound tiers were not activated. We ran this test with the larger alphabet, containing 3,774 anagram key values.

In Table 4.5 we present the results for TISC when it is given no real lexicon, only the correct word forms for the erroneous ones it is presented with. TISC manages to correct 99.6% on average of the errors on ten sets of 2,000 randomly chosen real-world errors. The standard deviation in upper bound recall between the sets is very low at 0.0012. The table further shows how many typos were corrected within what LD to the correct form, thereby giving an idea of the error type distribution within the sets. It can be seen from the standard deviation per LD that the number of items per LD has an even distribution over the sets. No errors exceeding LD 4 happened to be included in any of the sets by

Evaluated on:						
Category	LD 1	LD 2	LD 3	LD 4	Total	%
deletion	1,333	51	3		1,387	47.516
insertion	647	19	4		670	22.953
transposition		332			332	11.374
substitution	309	20			329	11.271
multiple		57	14	1	72	2.467
run-on	58				58	1.987
capitalisation	24				24	0.822
multisingle		5	6	1	12	0.411
space to dash	35				35	1.199
1st ch. delet.	(21)				(21)	(0.719)
1st ch. insert.	(5)				(5)	(0.171)
1st ch. transp.		(7)			(7)	(0.240)
1st ch. sub. lc.	(21)				(21)	(0.719)
1st ch. sub. uc.	(18)				(18)	(0.617)
total	2,406	484	27	2	2,919	
%	82.425	16.581	0.925	0.069		100

TABLE 4.4 Error type breakdown of the Reuters RCV1 benchmark evaluation set: error types evaluated on.

10x2K	Return	Correct	LD1	LD2	LD3	LD4	UBR
Mean	1993.1	1992.5	1546	421.5	22.6	2.4	**0.9963**
Std.dev.	2.47	2.42	10.64	11.42	2.94	1.36	0.0012

TABLE 4.5 Test results averaged over the ten sets of 2,000 randomly chosen typos, TISC LD restriction on CCs returned set at LD 4: We show the average number of typos for which a correction was returned, the average number for which the correct one was present, average number of typos corrected per LD and upper bound recall. Standard deviations per result are also given.

the random selection procedure. The results presented were obtained with LD restriction on CCs returned set at LD 4. We ran tests with the LD set at 2, 3, 4 and 9. We found that at LD 2 recall is lowest: this is only to be expected, because this prevents the system from correcting errors that require greater edits. The highest recall is reached with the LD restriction set at 4, with no further gain when set higher. At LD 9 just about anything could happen, most words could effectively be transformed in just about any other word, within the limits imposed by the alphabet. The fact that the same recall is reached at LD 9 as at LD 4 might lead one to conclude that apparently the system can do without an LD check, but this would be a mistake. We return to this matter later.

For this test, we do not report on best-first ranking scores. There being no lexicon to speak of, TISC's ranking mechanism has next to no words to work with and so cannot do its job. The system here has access to all correct word forms, and only these. Many are closely resembling word forms and fall well within the limits set by the LD, e.g. for *openion TISC here returned: option, opinion. The second CC is what is wanted given the context: 'world public *openion by pretending'.

We believe the score on overall recall firmly establishes that what we have is a powerful correction mechanism. We next discuss its few residual errors and the causes for no correction being made or the correct form not having been retrieved.

Residual errors

Typos for which no correction candidates were returned:

- *report-Kremlin, *yields-Rothfos, but also: *struc-tures: LD1: This type represents the majority of residual errors, 35 of 89 cases (39% of the residual errors). These fell victim to our space-dash-nothing variation modelling. The unigram tier did produce the corrections, but no output was produced because the only correction candidate retrieved differed from the input only by the AV for a dash. The filter should not have been activated in this test, it is meant to prevent precision errors due to allowable variation. Under normal working conditions it is unlikely only one CC would be returned. Note that these cases could be seen as examples of bad tokenization, rather than as typos. They are prevalent as an error type, as we have seen in Chapter 2, Table 2.3.
- *added.if and *announcement.in: LD 1 : All 17 cases (19% of the residual errors) are bad-tokenisation examples of two existing words joined by a dot: this is an oversight. The implementation contains a filter which prevents the dot being sent to the correction module,

whether on its own or embedded in a character string. Only the former should not be sent on.

- *participatipation for *participation* and *willingfully for *wilfully*: both are LD 4. It concerns four consecutive characters in both cases. We do not have 4-character anagram values in the typo-derived anagram value list, we have modelled for insertions only up to LD 3. For a while we considered using the powerset of input word characters for deriving the TAVs, but this proved prohibitively expensive.
- *momopology for *monopoly*: LD 3: This should have been within reach, but the trigram character value for *m + og* was not present in the TAV: we only calculate the values for consecutive characters.
- *parashooter for *parachuter*: LD 3: Same as the above. Allowing for 4 to 3 character substitions would give the correction mechanism too much scope.
- *partizan for *partisan* and *pazteurization for *pasteurization*: both are LD 1: these fell victim to the *s-z* variation modeling, which should have been modelled as *is-iz* variation. This would still have lost us the former. The correction candidate was proposed by the correction module but filtered out before the final output was produced.

Typos for which correction candidates were returned, but where the correct one was missing from the 5-best list:

- *concensut for *consensus*, correction candidate returned: coconut. This two-point error reveals a weakness of our system for LD 2 cases. A substitution needs to be made, but the type-derived anagram value for the character combination *c + t* was lacking, as only the TAVs for adjacent character bigrams are calculated. The LD between *concensut and *coconut* is actually only 3.
- *skilless for *skill-less*, CCs returned: smokeless, sickness. No anagram values for dash+character bigrams are available in the alphabet.
- *seeked for *sought*, CCs returned: second, secede, shied, signed, passed : LD 5: Out of reach of the TAVs and AAVs.
- *lae for *led*, CCs returned: later, sale, table, lauds, value. LD 2: The correct form did not make it into the 5-best list. The same typo in another of the ten test sets was ranked fifth and was therefore 'corrected'. This suggests there is room for improvement in the ranking mechanism on the unigram tier. Ties in the upgraded retrieval counts are not currently resolved.
- *olp for *to*, CCs returned: only, step, one, on. LD 3: The correct form *to* was retrieved, but not upgraded: neither front nor back does it match the typo. This problem occurs only with very short words.

10x2K	mean UBR	std. dev.	mean RR1	std. dev.
UNI-A1	0.9780	0.0028	0.9184	0.0051
UNI-A2	0.9785	0.0020	0.9216	0.0057

TABLE 4.6 Test results at LD 4 - average scores over 10 random sets: **10x2K**: ISPELL dictionaries, smaller (A1) and larger (A2) alphabets. Tests run on unigram tier only. Upper bound overall recall and best-first-ranking recall.

- *wouldn for *wouldn't*, returned: would, wounded. LD 2: No anagram values for apostrophe+character bigrams are available in the alphabet.

The examples of the correction candidates are illustrative: it may seem strange that *step* is suggested for *olp. It should be borne in mind that this entails the word *step* appeared in the correct word list for this test set, thus was present in the lexicon and was retrieved and ranked among the five best because it

- was within reach of the restrictions imposed on TISC, i.e. two to three character substitutions were allowed by the character trigram values in the large alphabet used here
- this suggestion made it into the 5-best list because no other words which have a greater 'resemblance' were available in the correct-word list lexicon, which would be the case when a normal lexicon is used.

Test 2: 10x2K experiments with the ISPELL dictionaries. Unigram tier only.

In this test we determine what TISC can achieve given its correction mechanism and given the collated dictionaries that come with ISPELL. This word list contains 144,106 word types from both the standard and expanded US and UK ISPELL dictionaries. All correct forms are again made available to the system. This test necessarily runs only on the unigram tier, there being no bigrams in the dictionary. We run this test, 10 times, with both the larger alphabet, which contains 3,774 anagram key values, and with the smaller, containing 498 items.

As can be seen from Table 4.6, results given the larger alphabet, which allows for richer edits to be made and potentially more errors to be corrected, are only slightly better. Given its order of magnitude higher processing cost we will not further conduct tests with the larger alphabet. This will mean that TISC cannot and will not be able to correct three character deletions in the further tests. We think this is a small price to pay, given the above results. Three character insertions remain within its reach as these values are derived from the input word,

10x2K	mean UBR	std. dev.	mean RR1	std. dev.
BI-A1	0.9819	0.0016	0.9135	0.0045

TABLE 4.7 Averaged test results at LD 4 for **10x2K** given a semi-supervised lexicon, smaller (A1) alphabet. Upper bound overall recall and best-first-ranking recall. Results obtained with all three tiers invoked and context available.

given that this is longer than five characters. Three to two character substitutions are also still within reach, but two to three character substitutions are not.

Test 3: 10x2K experiments with the ISPELL dictionaries. Bigram and compounding tiers too.

We next study what happens when we supervise the building of TISC's lexicons: we use the words available in the expanded ISPELL dictionary to filter the bigram list underlying TISC's own, fully unsupervised, lexicons. The bigram list is filtered as follows: we accept only those bigrams of which both words are in the ISPELL dictionaries. The bigram list thus obtained amounts to 21,812,660 bigrams. We apply no further filtering on the basis of some frequency cut-off to these. In effect, the lexicon derived from this bigram list should allow us to determine the contribution a simple bigram language model has to TISC's performance. The aim of this test is to see what the benefit is of also activating TISC's bigram and compound correction tiers. This test was run using only the smaller alphabet. Table 4.7 shows that including the word bigrams to the TISC lexicon has only a slight positive effect on TISC's overall recall in these upper bound tests. Standard deviation between the sets, however, is lower than in the unigram tier only results.

In comparison with the first test, both the second and third tests establish that in upper bound recall experiments the effect of the dictionary is only slight. Practically without interference of the words in the dictionary as in the first test or with possible interference of over 20 million unigram and bigram types as in this third, when all the correct forms are available, we see a loss in overall upper bound recall of only 0.015 or 1.5%. In the discussion of the fourth test, we analyse this interference in more depth.

This third test also appears to show that having these millions of bigrams in the lexicon has practically no effect on performance. That this is not in fact the case, we show in Test 4. That this is in fact an artefact of performing upper bound recall experiments, we show in Test 6.

Test 4: 10x2K: Regular TISC lexicons derived from NYT-BNC corpus bigram lists

Next we measure the performance of TISC given its proper, corpus-derived and noisy, lexicons. These were derived from bigram lists obtained from the combined NYT and BNC corpora, in order to emulate an international English dictionary, as we explained in Chapter 3. We present the results obtained when the bigram lists from which the lexicons are derived are cut off at various frequencies. The overall recall presented for TISC is obtained by taking the 5 best correction candidates into account. Remember that the input contexts employed in these tests were the first context for a particular error encountered in the Reuters corpus. In these tests, best-first ranking recall is therefore relative to the real-world input context.

In this upper bound evaluation experiment on **10x2K** we again ensure that all the correct word forms for the typos are present in the lexicon. In this way we can measure the upper bound recall obtainable by TISC given NYT-BNC lexicons and associated COOC-tables obtained from a particular corpus bigram list cut-off. We again compare the results obtained when correction is performed only at the unigram tier, i.e. the input context window of the two words preceding the error and the two words following is not presented to the system, versus those obtained when the context is presented and correction also takes place on the tier of the word bigrams and possibly on the compound tier.

Figure 4.2 shows that adding the bigram correction tier does not add a lot to the overall upper bound recall obtained. At lower cut-offs, we observe some gain. We also see that we lose nearly 3% upper bound recall using the largest lexicon versus using the smallest. All the words necessary to effect correction were available, so that cannot be the cause. What happens here is that the system was run with a wide reach in LD: it could return CCs differing from the typo by up to 4 edits. With ever larger lexicons more and more words within that reach are available, irrespective of the question whether or not they are correct words. This is best illustrated by an example: Table 4.8 lists the unigram tier ranked CCs per frequency cut-off for *fiture, which should be corrected as *future* given its original context: 'the bank has a policy of no comment on speculation about what it might or might not do in the *fiture.'.

At cut-off frequency 1 we see two French (loan-)words and what is another typo for *future* (context: '... that makes it incumbent upon the industry to start focusing on the *futre and doing it now.'). These gradually disappear with the larger cut-offs allowing for the actual correct

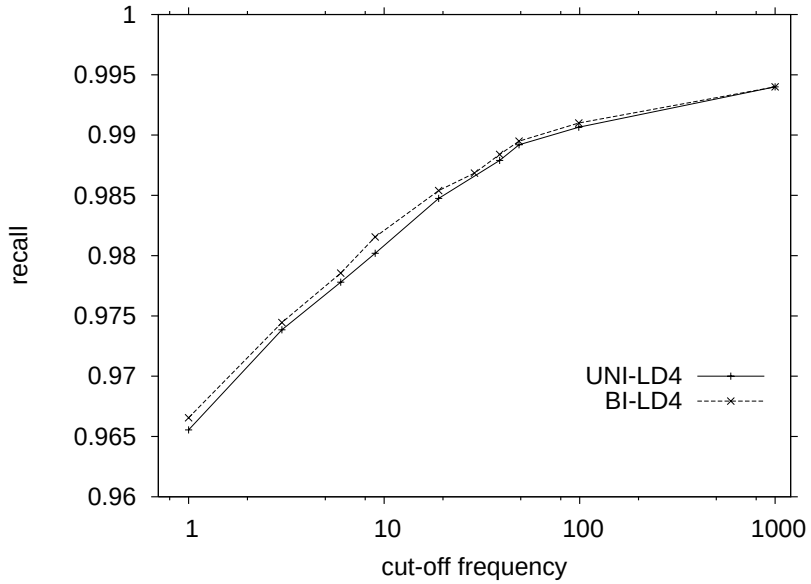

FIGURE 4.2 Evaluation results: average overall upper bound recall over 10 random sets (**10x2K**): LD 4. Tests run with proper NYT-BNC corpus-derived TISC lexicons with increased cut-off frequency.

Frq.	rank 1	rank 2	rank 3	rank 4	rank 5
1	friture	fixture	furtive	futre	fuire
3	friture	fixture	furtive	futre	fire
6	friture	fixture	furtive	fire	ture
9	friture	fixture	furtive	fire	ture
19	fixture	furtive	ture	fire	future
49	fixture	furtive	ture	fire	future
999	fixture	fire	future	figure	fixtures

TABLE 4.8 Unigram-tier ranked CCs for *fiture per lexicon cut-off frequency.

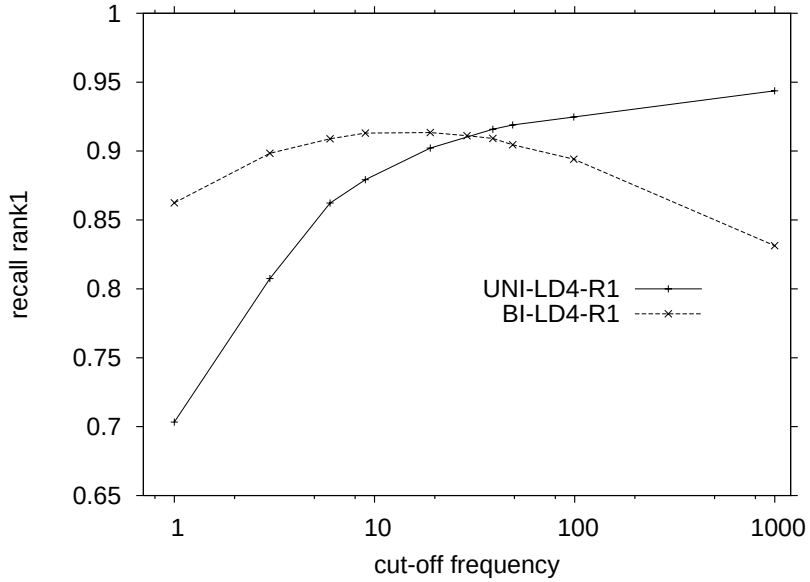

FIGURE 4.3 Evaluation results: average best-first ranking upper bound recall over 10 random sets (**10x2K**): LD 4. Tests run with proper NYT-BNC corpus-derived TISC lexicons with increased cut-off frequency.

Frq.	U1	U2	U3	U4	U5	B1	B2	B3	B4	B5
1	zid	zida	zidi	zide	zind	zid	zida	zidi	zide	zind
9	zidi	zide	zaid	zind	ziad	zidi	zide	zaid	zind	zeid
19	zaid	zind	ziad	**did**	bid	**did**	zaid	zind	ziad	bid
49	ziad	did	bid	id	cid	did	ziad	bid	id	cid

TABLE 4.9 Ranked CCs for *zid per lexicon cut-off frequency. To the left: U1-5: CCs ranked 1 to 5 proposed by the unigram tier, right B1-5: CCs ranked 1 to 5, reranked by the bigram tier.

candidate to make its way into the 5-best ranked list. So, allowing for greater LD CCs to be retrieved (furtive is at LD 4 from *fiture) and for more near-neighbours to be available in the lexicon, may crowd out the actual correct form expected. We therefore call this phenomenon **crowding**. Crowding is in a way an artefact of the fact that we only retain the five-best ranked candidates at the unigram tier, as well as not employing other ranking mechanisms there. It is also partly an artefact of the way we tested here, invoking only the unigram correction tier.

We next discuss what the contribution of the bigram correction tier is to the proper ranking of the CCs retrieved. This effect can be seen in Figure 4.3. We see that on the unigram tier only we lose nearly 25% in best-first ranking going from the lean and clean lexicon at cut-off frequency 999 to the big, noisy one at frequency 1. The contribution of the bigram tier at 999 is negative: there simply are not enough bigrams available to help in the ranking. At frequency 9 this is another story: we achieve best-first ranking in 91.3% of the cases versus 87.9% on the unigram tier only. This we illustrate in Table 4.9 by the short typo *zid for *did* (2-1-2 context: '. why *zid it take'). The bigram tier in all cases returned *did* as its best-first ranked CC. We see that only at cut-off frequency 19 the correct candidate *did* is no longer outcrowded, it has entered the list of 5-best CCs: this makes it concur with the best-first bigram-tier CC, whereupon the list is reranked accordingly. All the CCs one does not recognise as English words turned out to be lowercased names. This shows the hidden cost of lowercasing the frequency lists when we build the lexicon. This problem is most acute for shorter words and at the lower cut-offs, as can be seen by the drop in best-first ranking there.

Apart from invoking the bigram tier correction, we have other strategies to outdo the effects of crowding. The contribution of one of these is the subject of the next test.

Test 5: 10x2K: CICCL's contribution to performance

In this paragraph we present the scores obtained by TISC when we rerun the experiments on English as presented in the previous test. All else being equal, we now give TISC access to the CICCL-derived list of acquired typos culled from the NYT-BNC word bigram list.

Figure 4.4 shows that nothing much happens on the level of overall recall, when we give TISC access to the list of acquired errors and let TISC effect absolute correction. On the level of best-first ranking, as can be seen from Figure 4.5, we see a very noticeable effect, particularly with the smaller frequency cut-offs. It is striking that the effect of CICCL when correction takes place only on the unigram tier, with the larger lexicons, is more marked than the effect when the bigram tier is invoked too. This was to be expected, the lower the cut-off the more recurrent erroneous forms are incorporated in the lexicon. CICCL effectively manages to remove part of this noise. This is less markedly apparent when correction on the bigram tier is invoked too: there the bigrams by themselves steer the solution away from the noise, as we have seen in the previous paragraph. In the case of *zid, this did not help: remember we set a threshold on CICCL: no variants were sought for words less than 7 characters long.

Test 6: 12K: Results on the full 12,038 error list

In this paragraph we present the results obtained on the full 12,038 Reuters RCV1 error list. In the previous paragraph we showed that CICCL contributes to performance. Results shown here were obtained by using the list of acquired errors. What we contrast with here, is with the results TISC obtains when the correct words are not added to the lexicon. We therefore determine here what the system's **true recall** is, given a particular lexicon and the benefit of absolute correction.

Figure 4.6 shows the true recall obtained by TISC (restricted to LD 3) on the full 12,038 error list, with the higher score curves representing upper bound recall, obtained when all the words to be corrected were added to the lexicon, and the lower score curves the overall true recall: the actual results obtained with the real lexicons. The difference between the best true recall and its associated upper-bound recall, a drop from 0.989 to 0.918 at frequency 19, is: 7%.

Figure 4.7 shows performance at rank 1. Again, we see the contribution by the bigram tier is substantial: while we witness a drop from upper bound recall 0.910 to true best-first ranking recall at 0.834 on the bigram tier, this is 6.1% better than the true best-first ranking recall at 0.773 when only the unigram tier is invoked.

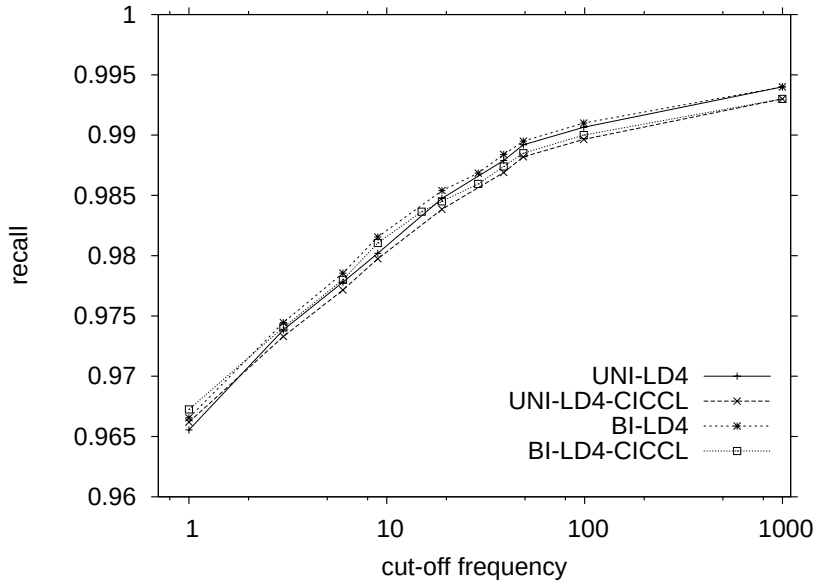

FIGURE 4.4 Evaluation results: average overall upper bound recall over 10 random sets (**10x2K**): LD 4. Tests run with proper NYT-BNC corpus-derived TISC lexicons with increased cut-off frequency, as well as with CICCL list of acquired errors.

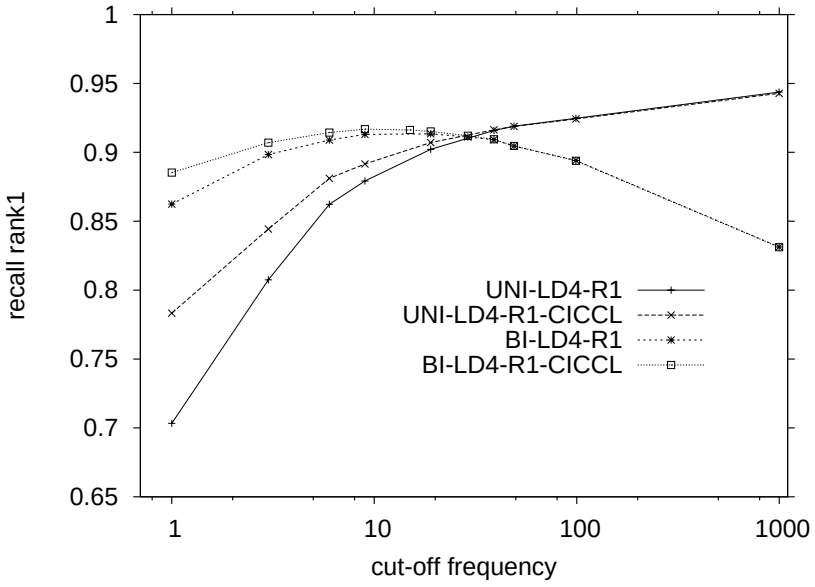

FIGURE 4.5 Evaluation results: average best-first ranking upper bound recall over 10 random sets (**10x2K**): LD 4. Tests run with proper NYT-BNC corpus-derived TISC lexicons with increased cut-off frequency, as well as with CICCL list of acquired errors.

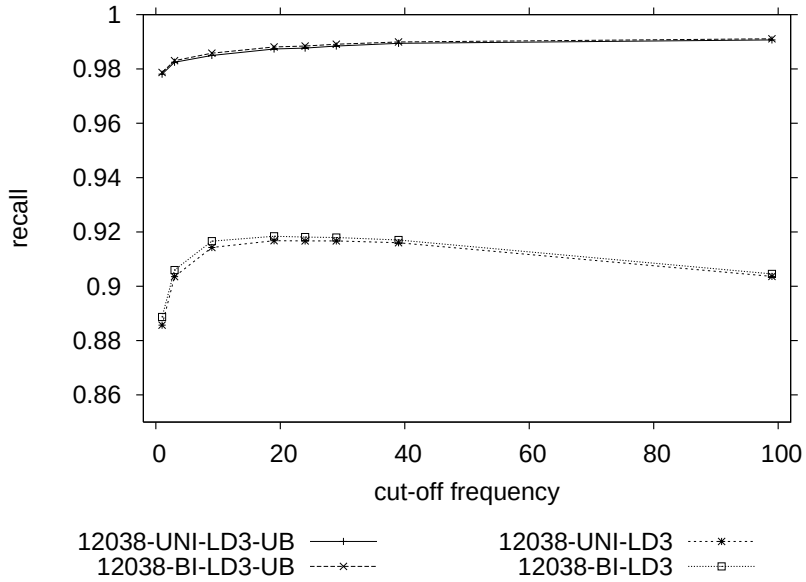

FIGURE 4.6 Evaluation results: Overall recall on 12,038 errors (**12K**) with LD 3 and increased cut-off frequency. Upper score curves represent the upper bound scores, lower score curves the overall true recall scores.

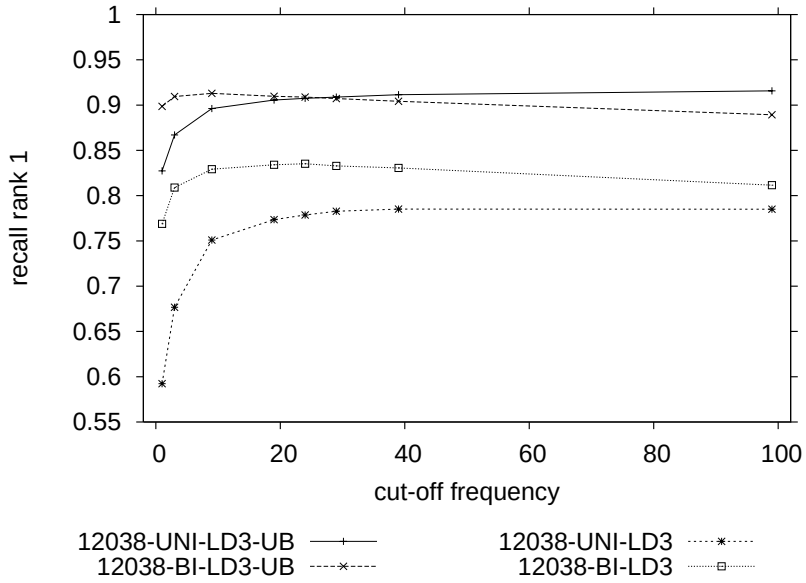

FIGURE 4.7 Evaluation results with increased cut-off frequency: Recall on rank 1 on 12,038 errors (**12K**) with LD 3 and increased cut-off frequency. Upper score curves represent the upper bound best-first ranking recall scores, lower score curves the overall true best-first ranking recall scores.

Intermediate conclusions

This concludes the tests on recall obtained on error lists. We have now established that we have a powerful correction mechanism and that TISC obtains far better results by performing correction on the three tiers than when only the isolated word tier is invoked. Further, that given its own corpus-derived lexicons, the system benefits from employing absolute correction on the basis of the CICCL-derived list of acquired errors. Finally, that there is a substantial difference in performance if true recall is measured rather than upper bound recall. In this we have again seen a substantial contribution of the bigram tier to performance expressed in best-first ranking recall.

In the next subsection we address how the system performs in the full task of first detecting, then correcting typos in their natural environment: the context.

4.3.3 Typo-in-context evaluation: F-score on full task

In this subsection we address what we consider to be the full task. Rather than merely measuring correction recall as we did before, we here present the system with typos in their original context: the sentence containing the typo which was chosen randomly from the Reuters RCV1 corpus. We measure performance in terms of the F-score. Given that the system is presented with errors in a context, we do not solely measure its ability to correct incorrect forms, but also to discern between correct and incorrect input forms. The fact that the context is limited to just the sentence means that the distribution of errors versus non-errors is skewed compared to their actual distribution in the full Reuters RCV1 corpus. We later demonstrate what happens when more and more context is present. Of the word forms for which correction candidates are returned, we check if the output contains the correct form. If so, the score for successful correction, i.e. overall recall, is calculated on the basis of its input context frequency, i.e. the typos' token count, no account being taken of the ranking of the correction candidates. The score for best-first ranking is likewise incremented if the first correction candidate is actually the correct form as specified in the evaluation file. The score for false positives, i.e. precision errors, is incremented in the same manner by the type's token frequency for those types for which the system returns correction candidates, but where the correct one is missing as well as for those where CCs are returned when the input word was not erroneous.

An issue not addressed in any other study we know of, is the effect that modelling greater edits has on precision. What gain in recall there can be made, may very well be undone by the loss in precision

caused if this is evaluated in the real world task of identifying the errors first and then correcting them. Further to be considered is the possible drop in best-first ranking of the correction candidates. The experiments performed were designed to shed light on these questions.

Test settings and evaluation results

For TISC we report score curves obtained by varying the threshold at which the corpora's bigram lists were truncated (cut-off frequencies: 1–19, 29, 39 49 and 99 for NYT-BNC). The implementation used was the same as for Dutch as it contains no provisions specific to either, apart from the allowed-variation switch mentioned earlier, which was here turned on. The input text frequency threshold, which prevents words appearing frequently from being sent on to the correction module, was set at 20, for all tests. The system was here run with the CICCL-derived list of acquired errors made available to it. We present the score curves for tests run with the LD limitation on correction candidates returned set at LD 2, 3 and 4.

Results presented here were obtained on **tokens**. Figure 4.8 shows the recall and precision obtained on the benchmark data. The figure introduces a new feature. The curved, dotted lines are **isometrics**: collections of points with the same value for the metric (Flach, 2003). Here, the isometrics depict the F-score, the top right point being perfect performance, i.e. an F-score of 1.0 and so the isometrics from right-up to left-down represent the F-scores at 0.9, 0.8 and 0.7, in this figure. Our results thus fall in between the 0.8 an 0.9 F-score isometrics.

Figure 4.9 shows the results obtained on best-first ranking.

We see that TISC's lexicons based on the lower frequency cut-offs produce the highest precision. Recall rises to a particular ceiling as the cut-off frequency is set higher, precision drops, with less and less information being available. The same pattern is repeated in the best-first ranking scores, though at consistently lower recall and precision levels. As concerns overall F-score, we only see divergence between the three LD-levels score curves at the lower frequency cut-offs. This is even more apparent in the best-first ranking scores. What is happening is that with lower frequency cut-offs more of the types' variants in the corpus are present in the lexicon. More of these are retrieved as correction candidates and competition for the best-first ranked place is tougher, whereby the correct form does not always make it to its rightful first position through crowding. Limiting the LD to 2 reduces crowding: fewer correction candidates are returned, allowing for better best-first ranking and thus somewhat higher precision and recall on rank 1. While higher LD is necessary to be able to correct typos which

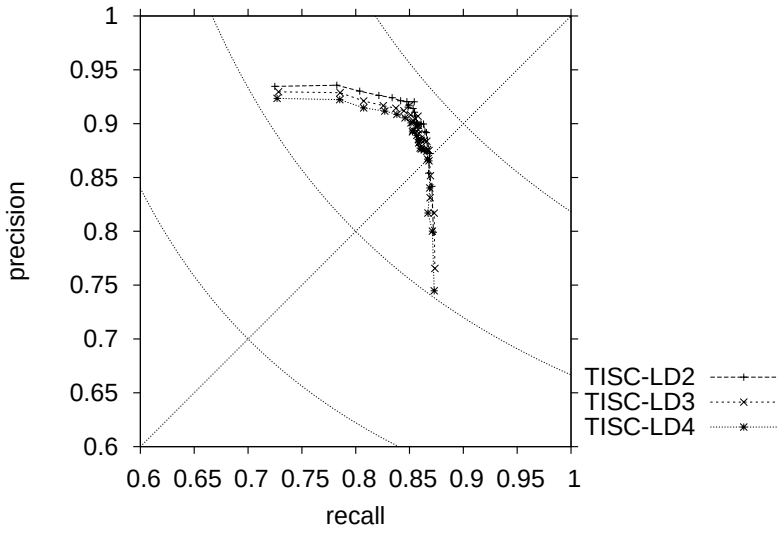

FIGURE 4.8 Evaluation results: English - Reuters RCV1 benchmark data: Precision and overall true recall. Shown are results obtained with LD 2, 3 and 4. Leftmost point: cut-off frequency 1. In sequence left to right: cut-offs 2–19, 29, 39 49 and lowest point: 99. Our results fall in between the 0.8 and 0.9 F-score isometrics.

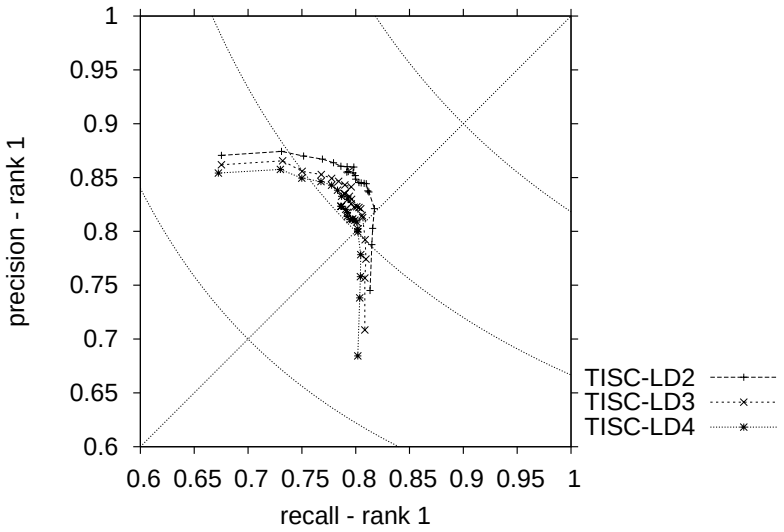

FIGURE 4.9 Evaluation results: English - Reuters RCV1 benchmark data: Precision and best-first ranking recall. Shown are results on best-first ranking obtained with LD 2, 3 and 4. Leftmost point: cut-off frequency 1. In sequence left to right: cut-offs 2–19, 29, 39 49 and lowest point: 99.

require more edits to be resolved to their correct form, this is actually counterproductive: there are far fewer of these higher LD typos in a natural distribution of error types and striving to resolve these implies retrieving more correction candidates and lower precision overall.

4.3.4 Evaluation of mean-median-percentiles parameter

We ran experiments to measure the effect of the mean-median-percentile parameter on the Reuters RCV1 benchmark set. We compared the performance of the hand-tuned Zipf Filter constant with results obtained with the mean setting and with the percentiles at 10, 20, 30, 40, 50 (the median), 60, 70, 80 and 90%.

The corpus-derived thresholds do not improve on the manual settings, at least not on overall F-scores. The mean settings consistently produce somewhat lower scores than the manual Zipf Filter settings. Given the percentiles based settings, at the very lowest frequency cut-offs, there is gain in precision, with attendant loss in recall. This is also reflected in the best-first ranking scores, where we also observe a small gain in precision. We observed the following pattern: higher percentiles settings make the system perform on a par with the mean settings.

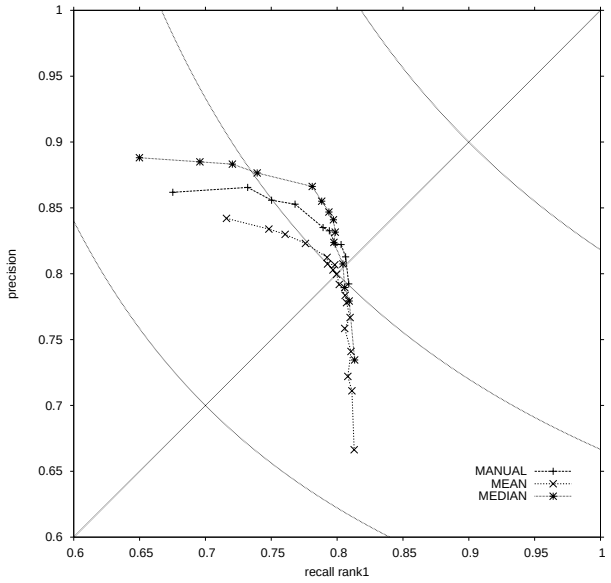

FIGURE 4.10 Best-first ranking scores obtained with the manual Zipf Filter settings, the corpus-derived mean and median (percentile: 50) settings. Tests run on Reuters RCV1 benchmark set with LD 3 and lexicon cut-off frequencies: 1–19, 29, 39 and 49 (top left to bottom right).

factor	R	P	F	RR1	PR1	FR1
mean / 0.5	**0.896**	0.759	0.822	**0.843**	0.714	0.774
mean / 1	0.886	0.817	0.850	0.840	0.774	0.806
mean / 1.5	0.876	0.845	0.860	0.834	0.804	0.818
mean / 2	0.867	0.862	**0.865**	0.826	0.820	**0.823**
mean / 2.5	0.848	0.870	0.859	0.807	0.828	0.817
mean / 3	0.835	0.878	0.856	0.797	0.837	0.816
mean / 4	0.812	0.899	0.853	0.775	0.859	0.815
mean / 5	0.791	**0.904**	0.844	0.755	**0.863**	0.805

TABLE 4.10 Dividing the mean value obtained from the corpus by a small factor enhances recall, dividing by larger factors enhances precision.

The lower the percentiles are set, the higher precision, outranking the manual settings, but with attendant loss in recall, resulting in slightly lower F-scores. This can be explained by the fact that lower percentiles put the threshold very low, validating too many types and correcting too few. Since the ratio of incorrect versus correct types is very skewed in favour of the correct types, this explains the higher precision obtained. The fact that this does not cause recall to drop completely is explained by the interaction between the Zipf Filter thresholds set and the unigram correction tier which is also invoked during the spelling checking phase. In Figure 4.10 we contrast the results obtained with the manual, mean and median (percentiles: 50) settings on best-first ranking on the Reuters RCV1 benchmark set.

We further experimented with manipulating the mean settings. In Table 4.10 we list results obtained by dividing the mean values by a number of factors. We see this acts as a moveable threshold, which can be set to let the system focus on achieving higher recall or higher precision, as warranted by the specific application one has in mind.

4.3.5 Discussion of performance in comparison with ISPELL, ASPELL and MPT

In this subsection we compare three isolated-word spelling error detection and correction systems, each of which has trusted dictionaries, with TISC, a context-sensitive system equipped with lexicons derived in an unsupervised way from corpora, supplemented with a list of typos unsupervisedly derived from the lexicons. The research questions we try to answer here are, first, whether it is feasible to perform correction up to the levels attained by the state-of-the-art systems given the novel correction algorithm we propose and the corpus-derived data we employ. Second, given that the typos are presented to the systems

in their real-world context, to what extent TISC's word bigram based language model and the cooccurrence information derived from the bigram list help the system to avoid reporting false positives compared to the isolated-word systems.

How we tested

Both ISPELL and ASPELL were run with the same dictionaries. These were the concatenated standard ISPELL US and UK dictionaries, which are by default supplemented with a medical dictionary. As a separate test, we further augmented these concatenated dictionaries with the US and UK extra large dictionaries.

ISPELL was run in its native batch-mode. However, the fact that it evaluates compounds written with a dash as two separate words and returns them on separate lines, obliged us to rewrite these cases as separate words, in both input and evaluation files used with ISPELL. So the original pair 'rollar-coaster#roller-coaster' was rewritten as 'rollar#roller', the RPCs here being omitted by virtue of being identical and therefore correct words. This might have a slight impact on performance, the pair 'sea-sawed#see-sawed' turning in 'sea#see', both of which are correct words. In any case, these compound-confusables are rare. In the same vein, we let ISPELL have the apostrophe as an extra valid word character. This enables it to suggest *Glencore's* for *Glencore'w. By default ISPELL sorts its suggestions alphabetically. We used the option of not sorting the CCs, which produces the output in the order in which the suggestions were arrived at:

- unsorted: & mnister 4 0: minister, minster, mister, meister
- sorted: & mnister 4 0: meister, minister, minster, mister

The unsorted items show that ISPELL first searches for deletions, then transpositions, insertions and substitutions. ISPELL does not perform any other ranking of its suggestions. We nevertheless report its score on best-first ranking, so as to get an idea of what is achieved without an explicit ranking mechanism. Bear in mind that for the majority of typos only one correction candidate is possible and that this one suggestion is then perforce perfectly ranked.

ASPELL was run in ISPELL compatibility mode, which means that apart from the correction engine, all else was equal to the ISPELL evaluations.

MPT was run with both its UK and US dictionaries separately. We saw no way of providing it with a hybrid American/British English dictionary. We do not know how large MPT's dictionaries are. We added the correct forms for the typos evaluated as the user's dictionary: 'cus-

System	Dictionary	# Words	OR	RR1	RR5
upper bounds: **correct word forms added to dictionary**					
ISPELL	UK-US-MED-COR	40,934	0.918	0.831	0.901
ISPELL	UK-US-MED-XLG-COR	94,601	0.910	0.814	0.891
ASPELL	UK-US-MED-COR	40,934	**0.992**	0.840	0.967
ASPELL	UK-US-MED-XLG-COR	94,601	0.991	0.811	0.960
MPT	UK-COR	unknown	0.942	0.873	0.938
MPT	US-COR	unknown	0.941	0.873	0.937
TISC	NYT-BNC/23-COR	196,883+	0.988	**0.920**	**0.988**
true scores: **proper dictionaries**					
ISPELL	UK-US-MED	36,472	0.875	0.789	0.858
ISPELL	UK-US-MED-XLG	90,277	0.886	0.792	0.867
ASPELL	UK-US-MED	36,472	0.945	0.800	0.920
ASPELL	UK-US-MED-XLG	90,277	**0.965**	0.791	**0.935**
MPT	UK	unknown	0.923	**0.856**	0.919
MPT	US	unknown	0.913	0.846	0.909
TISC	NYT-BNC/23	196,883	0.926	0.844	0.926

TABLE 4.11 System evaluated, dictionaries used, number of words in dictionary (lemmata for ISPELL and ASPELL, expanded word forms for TISC), overall recall and recall in ranks 1 and 5, upper bound and true scores.

tom.dic'. MPT only offers an interactive mode, which was not appealing given the amount of text we intended to check. We had a macro developed in Visual Basic [2], to be run in Microsoft Excel, which simply collects the correction candidates returned and pastes them in successive columns, the first column after the typo corresponding to the correction candidate ranked first, the second to the second, and so forth. Paired with the input text token frequency info per type and the evaluation file containing, per typo, the desired correction, this nevertheless automated the evaluations of MPT.

For TISC we list the scores obtained with lexicon cut-off frequency at 23, i.e. with just under 200,000 expanded word types available to it. TISC had access to the CICCL-list of acquired errors and was run with the LD limit set at 2.

Evaluation results for the full error list: 12K

In Table 4.11 we present the upper bound and true recall scores obtained on **12K** by the four systems evaluated.

[2]We are indebted to Jozef Vancoillie, autodidact hacker/problem-solver, for providing us with this macro.

The upper bound recall scores show us that given a 'perfect' dictionary, i.e. one in which all the correct forms for the typos to be corrected are present, aspell and tisc perform practically on a par in overall recall. mpt performs 4% worse and ispell, through its limitation to Damerau-edits, 7%. On best-first ranking and 5-best ranking, tisc outperforms all other systems. mpt performs markedly better in best-first ranking than aspell, which in that respect is outclassed even by ispell.

As regards the true scores, aspell outperforms the other systems in overall recall and 5-best ranking recall, but only attains the level of ispell in best-first ranking. tisc is runner-up to mpt in best-first ranking and to aspell in 5-best ranking, both by about 1%. The greater difference between aspell and tisc on true overall recall is the result of tisc here being limited to ld 2. However, tisc's score on true overall recall and best-first ranking recall dropped by about 1% when run with ld limits set at 3 and 4.

We think these scores allow us to conclude that spelling error correction on the basis of unsupervised corpus-derived lexicons nearly on par with the level of performance obtained by state-of-the-art systems is feasible and has here been achieved.

Scoring and evaluation results for typos within a limited context

We measure performance for all four systems in the way we set out in 4.3.3. For all the correct types marked by ispell, aspell or mpt as 'not in the dictionary', or for which ccs are returned by ispell, aspell, mpt or tisc, the score for false positives (precision errors) is incremented by the type's token frequency in the input text. The results presented in Table 4.12 were thus again obtained on **tokens**, for all four systems. For ispell, aspell and mpt we used the dictionaries as in the test on the **12K** error list. Here we did not test with all the correct forms added. For tisc we report three scores: the one which achieved best recall, the one which achieved best precision and the one where the best balance between the two was achieved, resulting in best F-score.

aspell obtains the best overall recall. It tries hardest and reports most ccs. This achievement is offset, and in part explained, by its very poor precision. This translates into the lowest F-scores of the systems tested. It is striking that it performs so badly, in perspective, on best-first ranking. We see its best-first recall scores are 3 to 4% lower than ispell's, which does not even 'do' ranking.

tisc's best performance as regards overall recall on the English benchmark set was obtained with lexicon cut-off frequency 3 and the

System	Dictionary	R	P	F
true scores: **overall**				
ISPELL	UK-US-MED	0.885	0.523	0.657
ISPELL	UK-US-MED-XLG	0.881	0.543	0.672
ASPELL	UK-US-MED	**0.938**	0.415	0.575
ASPELL	UK-US-MED-XLG	0.935	0.468	0.624
MPT	UK	0.925	0.543	0.684
MPT	US	0.917	0.511	0.656
TISC-R	NYT-BNC/3	0.915	0.662	0.768
TISC-P	NYT-BNC/2	0.782	**0.934**	0.852
TISC-F	NYT-BNC/8	0.854	0.920	**0.886**
true scores: **rank 1**				
ISPELL	UK-US-MED	0.817	0.483	0.607
ISPELL	UK-US-MED-XLG	0.805	0.496	0.614
ASPELL	UK-US-MED	0.785	0.348	0.482
ASPELL	UK-US-MED-XLG	0.764	0.382	0.509
MPT	UK	0.855	0.502	0.632
MPT	US	0.848	0.472	0.607
TISC-R	NYT-BNC/3	**0.859**	0.621	0.721
TISC-P	NYT-BNC/2	0.731	**0.874**	0.796
TISC-F	NYT-BNC/8	0.798	0.860	**0.828**

TABLE 4.12 System evaluated for typos within a limited context, dictionary used, recall (R), precision (P) and F-score (F). Overall and best-first ranking scores.

the Zipf filter threshold: mean values times four. Best precision was obtained with cut-off frequency 2 and the Zipf filter threshold set manually. Best F-score was obtained with cut-off frequency 8 and the Zipf filter threshold set manually. Results for TISC reported at rank 1 were those obtained with the same settings as the overall scores reported. Only as concerns precision did we observe a higher score on rank 1 with different parameter settings: with percentiles set at 40% and lexicon cut-off at frequency 1: recall: 0.649, precision: 0.891 and F-score: 0.751.

It may come as a surprise that TISC's manual Zipf filter settings produced the highest performance as concerns precision and F-score. This may be due to the fact that as a consequence of fine-tuning on the development set, we eventually had slightly different settings in the checking module than in the correction module. With the automatic settings these are the same throughout the program. However, the runners-up with the automatic settings are close. For precision we see that cut-off frequency 1 and percentiles at 40% gives: recall: 0.680, precision: 0.932 and F-score: 0.786. For F-score at cut-off 11 and percentiles at 40% , we get: recall: 0.842, precision: 0.898 and F-score: 0.870. Based on this we think it is very likely that had we covered the full gamut of cut-off frequencies and threshold settings in the automatic tests, we would have encountered the optimal settings that would outperform the manual ones. As there were about 20 automatic threshold settings to explore, we did not think this was called for. We are also confident that these optimal settings would prove to be found in manipulating the mean threshold values.

We see that in overall recall TISC lags behind the performance of MPT by 1% and of ASPELL by 2%. In best-first ranking, TISC manages to slightly outperform MPT, but greatly outdo ASPELL. We further see that TISC can outperform both ISPELL and MPT by as much as 40% and ASPELL by 50% in overall precision. Precision reduced by the pursuit of the highest possible level of recall is still 12 to 15% above ISPELL and MPT, both of which attain the same level, and 20 to 25% above the level reached by ASPELL. The balance in recall and precision which is obtained by TISC translates into an F-score which lies 20% higher than what MPT achieves. This is true on the best-first ranking level as well.

We believe the results shown here prove sufficiently that context-sensitivity as provided by the simple word bigram model we employ greatly reduces the false positive rate incurred by isolated-word systems.

Obtaining balanced precision and recall

We think we may conclude that in order for a spelling checking and correction system to obtain a balance between recall and precision

- it should be provided with far more lexical information than has been the norm to date;
- it is beneficial to restrict the scope in LD of the correction mechanism;
- it should be provided with context information.

As we have seen, ASPELL achieves a high overall recall, but only a low precision. The more information, i.e. the larger the lexicon available to TISC, the higher precision it reaches. At the very top, i.e. with all the information in the corpus save the word bigram hapaxes, precision still is better than with more information discarded, admittedly at the cost of recall. Recall and precision are in balance at cut-off frequency 8, for this particular hybrid British and American English lexicon, tested on this particular benchmark set.

This concludes our evaluations of the English version of TISC. We have shown that Text-Induced Spelling Correction is a viable and competitive spelling error detection and correction solution for English. In the next section we investigate whether the same goes for Dutch.

4.4 Evaluation on Dutch

Dutch is a language related to English, both being members of the family of North-Sea Germanic languages, but has a richer morphology and its highly productive compounding produces more word types because compounds are written as single words. In this section we investigate whether the same Text-Induced Spelling Correction methodology we have demonstrated to work for English, works for Dutch as well.

In our evaluations on Dutch we concentrate on the full task of detecting and correcting typos within their context. We here employ a realistic context size for spelling checking: the full newspaper article the particular typos appeared in. We study the effect of having more or less context surrounding the typos, the effect of using a greater or smaller LD restriction on the correction candidates returned, the effect of the automatically set Zipf Filter thresholds, and finally, the effect of having a larger corpus to derive the lexicons from. For Dutch we did not apply CICCL, note therefore that all results reported here were obtained without employing a list of acquired errors. In the last subsection, we compare TISC to the Dutch versions of ISPELL and MPT.

4.4.1 Composition of the evaluation files

For evaluation purposes, we proofread the Dutch version of the newspaper *Metro* during April-May 2003 and collected the non-word errors encountered. This amounts to 129 non-word errors, which were extracted from the online version with the full article they appeared in. We used this first batch (Metro1) for development purposes. A second bout of proofreading yielded a second, similar batch, which we reserved for testing purposes only. We report the scores on the benchmark evaluation set: Metro2, which contains 126 typos. Further statistics on development and benchmark sets are provided in Table 4.13.

While still a far cry from the total amount of word tokens published in the full editions of the Metro, we believe the benchmark test set represents a more realistic sample of text to be spelling checked. The error ratio observed here, i.e. 1 in 200 words represents a typo, is about twice that made by competent typists (Grudin, 1983) or that observed in the Reuters RCV1 corpus. These news stories were published and one may therefore assume they had been subjected to proofreading and/or automatic spelling checking prior to publication.

Collecting the development and evaluation sets was no sinecure. We proofread the paper version and then used the portable document format version which is daily posted on Metro's website to extract the electronic version of the articles in which we had found errors. This we did by copying the text from Adobe Acrobat Reader to an editor. Unfortunately, this is a lossy process: due apparently to 'ligatures' this loses all occurrences of the letter combinations *fl* and *fi*. Apostrophes are also lost. So the Dutch word *financieel* becomes *nancieel, which is a non-word, but the word *fiets (bike)* becomes *ets (etching)*, which is a real word. So after extraction, another round of proofreading was required for text running up to 50,000 words, about the size of a small novel. We also tried using optical character reading (OCR) to circumvent this problem, but that proved futile: it introduced even more OCR-induced errors.

4.4.2 Test settings

For TISC we report score curves obtained by varying the threshold at which the corpora's bigram lists were truncated (frequencies: 2–10 and 15 for ILK-TWENTE). The implementation used was the same as for the English evaluations and so was the input text frequency threshold, set at 20, above which words are never sent on to correction. However, the switch telling TISC to accept the specifically English admissible variants detailed above, was here turned off. We also had to lower the constant set for the Zipf Filters. When we first ran preliminary, development

Set Status	Metro1 Development	Metro2 Evaluation
context	article	article
tokens	21,919	25,750
types	5,747	6,441
type/token ratio	26.22%	25.01%
errors	129	126
error/type ratio	2.25%	1.96%
error/token ratio	0.59%	0.49%

TABLE 4.13 Statistics of Metro development and evaluation sets.

Category	LD 1	LD 2	LD 3	Total	%
deletion	54	3		57	45.238
insertion	34	1	1	36	28.571
substitution	16			16	12.698
transposition		3		3	2.381
multiple		2		2	1.587
space deletion	8			8	6.349
space to dash	2			2	1.587
capitalisation	1			1	0.794
multisingle		1		1	0.794
1st ch. delet.	(1)			(1)	(0.794)
1st ch. insert.	(1)			(1)	(0.794)
1st ch. sub. uc.	(1)			(1)	(0.794)
total	115	10	1	126	
%	91.270	7.937	0.794		100

TABLE 4.14 Error type breakdown of Metro2 benchmark evaluation set.

set, tests on Dutch evaluating on tokens instead of on types we obtained poor precision results, in the 40% range. The results reported in Reynaert (2004) were on types. We had then used the same settings for Dutch as for English and obtained reasonably satisfactory results. False positives are often correct words which occur more than once in the input text and thus have a heavier toll on precision when tokens rather than types are counted. The Dutch corpus is smaller than the English one. This was further compounded by the greater sparseness of Dutch due to its richer morphology. Consequently, the COOCs observed at a particular bigram list cut-off are relatively higher for English than for Dutch. This observation formed the basis for the automatically set Zipf Filter thresholds we discussed in Chapter 3. Results reported below were obtained with the Zipf Filter constant still set at 1,000 for words shorter than 5 characters, but lowered from 100 to 50 for those of length 5 or more. This results in more types being sent to the correction module, more erroneous ones being corrected, hence a rise in recall and accompanying rise in precision.We also report on tests with the automatically set Zipf Filter thresholds.

4.4.3 Error list: Gauging the level of difficulty of the Dutch benchmark set

Before we move on to the evaluation on the full task, we present an estimate of the level of difficulty presented by Metro2. A breakdown of the actual error types in the Dutch benchmark set is given in Table 4.14.

As we have seen above, MPT scores very well on recall for English. We therefore take MPT's true recall score on **types** on the list of typos in the Dutch benchmark set to be indicative of the benchmark's level of difficulty. The overall true recall score on types for MPT on Metro2 is 0.675. Best-first ranking recall is 0.627. When we compare the level of overall recall reached on Dutch with that reached on English, we see the score on Dutch is far lower. Apparently, Dutch presents a greater challenge to spelling correction systems.

4.4.4 Typo-in-context evaluation: F-score on full task

TEST 1: Effects of context size

For Dutch, we report scores obtained with the full evaluation files as well as with the errors within a 3-sentence context, extracted from the full articles. This was designed to reveal the difference in precision which results from more context. The more context given to the errors, the more skewed the distribution of errors versus non-errors becomes, which is more in line with the distribution naturally seen in reality. Remember we have observed 1 error per 400 words of running text

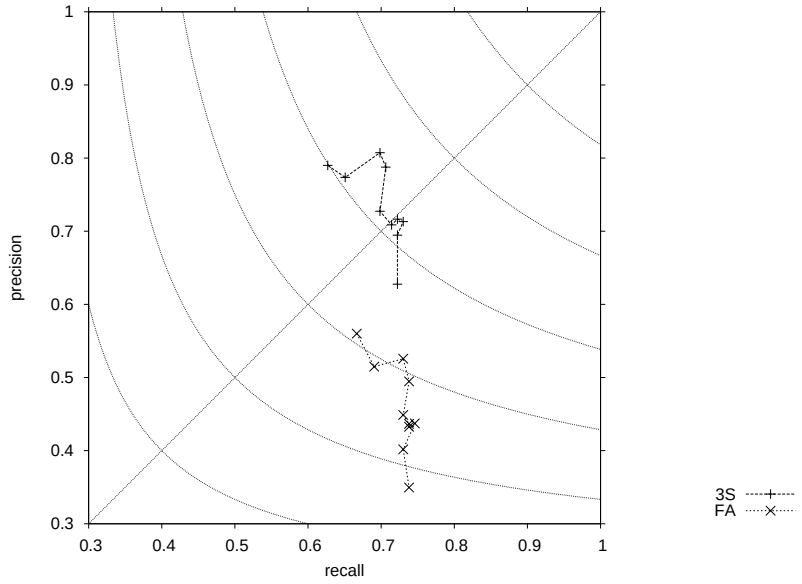

FIGURE 4.11 Evaluation results: Dutch - Metro2 benchmark set: effects of more context. LD limit at 3. Lexicon cut-offs: 2 to 10 and 15: top left of curves to bottom left. Remark the loss in precision due to more context, but also the shift to the right due to greater recall of the full article (FA) curve compared to the 3 sentences (3S) curve.

in the Reuters RCV1. Given the longer contexts here, we are at about 1 in 200. More context gives more word types, and more scope for the system to report false positives. This is reflected in the scores on precision obtained.

Discussion of context effects

The score curves for Dutch are more erratic than they were for English. This can be explained by the facts that

- the corpus contained only half the number of tokens as the English one: this results in lower frequencies for the types incorporated in the lexicon and less reliable COOCs overall.
- the error to type ratio was much lower than that for English: this is due to our employing more input context, but also to the richer morphology of Dutch, which results in a higher type to token ratio than one would observe in English.

The effect of more context is particularly striking as regards precision. Nicely observable is the fact that given more context, TISC may reach higher levels of recall due to input-text derived information that aids correction. All else but the context being equal, at frequency cut-off 4 which gives us the best F-score for the full articles, i.e. 0.611, we see a drop in precision of 28%, from 0.807 to 0.526. The gain in overall true recall is then: 3.17%, from 0.698 up to 0.730.

Having established the importance of studying the import of error to token ratio in spelling error detection and correction, we now move on to seeing what the effect is of variation in the scope of the correction mechanism on performance.

TEST 2: Effects of LD restriction

Another factor which has an impact on performance is the scope, expressible in LD, given to a system. Allowed a larger LD to search for correction candidates, a system selects or retrieves more correction candidates and ranking becomes harder to perform correctly. We study that effect here.

Table 4.15 lists the scores obtained at cut-off frequencies 2 and 15 with LD limit set at 3, 4, 5, 10 and 99.

Discussion of LD restriction effects

We see that varying the LD at cut-off frequency 2, affects recall and precision. The lower level of recall at frequency 2 is the effect of crowding. At LD 4 we get the highest recall. Nevertheless, the higher precision at LD 3 still makes for the best F-score, though with a small margin.

At cut-off frequency 15 we see no variation in recall at all anymore: widening the search space does not allow for more typos to be corrected.

LD	OR	P	F
FRQ 2			
LD 3	0.659	0.568	0.610
LD 4	0.667	0.560	0.609
LD 5	0.659	0.516	0.578
LD 10	0.651	0.474	0.548
LD 99	0.651	0.456	0.536
FRQ 15			
LD 3	0.738	0.372	0.495
LD 4	0.738	0.350	0.474
LD 5	0.738	0.329	0.455
LD 10	0.738	0.292	0.418
LD 99	0.738	0.284	0.411

TABLE 4.15 Metro2 benchmark set: effect of varying the LD on overall recall (OR), precision (P) and F-score (F).

It does still allow for more correction candidates to be retrieved, which explains the attendant loss in precision. Note that at LD 99 the only restriction on retrieval is the restriction imposed by the alphabet. The lower precision reached at frequency 15 is the result of the smaller lexicon: less of the information needed is available, both lexical information and derived COOCs will be less complete and reliable.

TEST 3: Effects of using a larger corpus

This test allows us to study the effects of using more data. We built new Dutch lexicons and cooccurrence tables, adding the Twente Update to the Twente Corpus used so far. We present a range of scores, obtained with varying automatic Zipf Filter settings based on manipulated mean settings and percentiles.

Table 4.16 lists the scores obtained with the lexicons derived from the ILK-Twente corpus as well as from the concatenated ILK-Twente and Twente Update corpora. Lexicon cut-off frequency was 4. The LD was set at 3.

Discussion of the effects of using a larger corpus

Using a larger corpus, we are confronted with the limitations of our approach. We gain precision, but the gain is in part offset by a loss in recall. Simply looking at the number of typos which were not sent on to correction, 26 for the ILK-Twente and 33 for the ILK-Twente and Update lexicons, tells us that using a larger corpus means incorporating more and more of the errors to be corrected in the lexicon, with higher

	ILK-TWENTE			+ UPDATE		
SETTING	OR	P	F	OR	P	F
25%	0.595	**0.600**	0.598	0.571	**0.632**	0.600
50% - median	0.643	0.566	**0.602**	0.603	0.589	0.596
75%	0.706	0.509	0.591	0.683	0.558	**0.614**
mean / 0.25	**0.778**	0.142	0.240	**0.754**	0.210	0.328
mean / 0.5	**0.778**	0.196	0.313	0.746	0.288	0.416
mean	0.754	0.281	0.409	0.714	0.391	0.506
mean / 1.5	0.738	0.342	0.467	0.698	0.456	0.552
mean / 2	0.722	0.397	0.513	0.659	0.509	0.574
mean / 2.5	0.690	0.433	0.532	0.627	0.527	0.572

TABLE 4.16 Metro2 benchmark set: effect of varying the corpus size on on overall recall (OR), precision (P) and F-score (F). Cut-off frequency: 4. LD limit: 3.

COOCs, more likely to be validated. Looking at the actual typos missed, we see that most of these involve missing diacritics or other highly recurrent typos (e.g. *comissie for *commissie* (commission), *geweldadige for *gewelddadige* (violent)). These are prime candidates for systematic prior corpus normalisation and for CICCL, steps we will take in the future.

Nevertheless, given the larger corpus, the gain in precision is substantial: with just the Twente corpus, we had 124 false positives, 79 with the Update (TISC settings: cut-off frequency 4, LD at 3, average divided by 2). Examining the data, we see that rarer morphological forms: e.g. *definitiever* [more definitive], *haperendste* [most wavering], *wurmt* [wriggles], compounds: e.g. *nekkenpakkende* [neck grabbing], *nietsverhullende* [hiding nothing], names: *Hulk*, *Calypso* and finally foreign words: *talks*, *wish*, are now being properly validated.

This does not mean the same levels of recall as with the smaller corpus cannot be reached with the larger. It means we will have to use a larger cut-off. At that larger cut-off, we should find we attain similar recall, but higher precision.

4.4.5 Performance in comparison with ISPELL and MPT

How we tested

Both ISPELL and MPT were run with their standard Dutch (Netherlands) dictionaries, the first in its native batch mode, the second with the Excel macro. We ran ISPELL in two modes: one using the -C parameter which is provided for compounding languages and which tells the system to not further evaluate words it can split into two in-dictionary

System	Dictionary	TR	P	F
true scores: overall				
ISPELL (-c)	dutch	0.627	0.054	0.099
ISPELL (+c)	dutch	0.516	0.073	0.128
MPT	dutch (NL)	0.659	0.081	0.143
TISC-R	ILK-TWC/2	**0.794**	0.164	0.272
TISC-P	ILK-TWC/2	0.540	**0.660**	0.594
TISC-F	ILK-TWC/2	0.683	0.610	**0.644**
true scores: rank 1				
ISPELL (-c)	dutch	0.548	0.047	0.087
ISPELL (+c)	dutch	0.460	0.065	0.114
MPT	dutch (NL)	0.611	0.075	0.133
TISC-R	ILK-TWC/2	**0.675**	0.139	0.231
TISC-P	ILK-TWC/2	0.460	**0.563**	0.507
TISC-F	ILK-TWC/2	0.571	0.511	**0.539**

TABLE 4.17 System evaluated on Metro2 (typos within full newspaper articles), dictionary used (for TISC: /2 indicates lexicon cut-off at frequency 2), on true recall (TR), precision (P) and F-score (F), overall and best-first ranking. TISC-R is the version of TISC optimized to obtain the best recall, TISC-P the version of TISC optimized to obtain the best precision and TISC-F the version that obtained the best F-score.

words, the second mode without this switch, as it was run for English. For TISC we present the best scores obtained on recall, precision and F-score and discuss which settings these were obtained with.

Scores obtained by ISPELL, MPT and TISC are presented in Table 4.17.

Discussion

We see that TISC outperforms both the Dutch MPT and ISPELL on both recall and precision when evaluated on typos within the full texts they appeared in. This achievement is due to TISC's context-sensitivity. We have shown that given more context, TISC's recall rises. This is the effect of evidence found that a word is acceptably spelled within the wider context, especially in cases where e.g. part of a compound is not at all present in the lexicon, but amply in the context: remember the case of *egionellawaakhonden discussed in Chapter 3, Subsection 3.4.4. Note that these results were obtained without running CICCL on the lexicon, so TISC was run without a list of 'acquired errors'. This means that there is further scope for improvement, yet. What we think is even more encouraging, is the fact the performance on best-first ranking of the candidates is also markably better for TISC. However, the top score

is an F-score of 0.539, with balanced recall and precision, both over 50%. In everyday terms this means that given typos in full context, we correct over half of the typos and the list of items returned contains target and non-target items in about equal proportions: just over 1 in 2 items returned is a corrected typo. If one were to let the system work in a fully unsupervised, automatic mode, the resulting text would improve, but only very slightly. We return to this topic in the next Chapter.

The ISPELL results with and without using the -C parameter are particularly enlightening. We see that when ISPELL does not further evaluate compounds for which it finds the two compounding parts in its dictionary, it achieves a precision nearly on par with the precision achieved by MPT. When it does further evaluate, it actually corrects more typos, reflected in the higher recall. This causes a steep drop in precision, however. If we compare ISPELL's output with that of MPT, we see that actually MPT employs a strategy very similar to the one offered when ISPELL's -C parameter is activated. MPT's higher recall seems then simply due to the fact it can handle greater edits. For both systems, precision is quite low, a direct result of the dictionaries containing insufficient information. For MPT this seems also the result of allowing for the scope of the search in terms of LD to be too wide.

We looked into this more closely by varying the amount of context. We made three test sets: the original one with the typos within their full article, a subset of this with the typos within their sentence only (1S) and another with the sentence preceding and following added to this sentence (3S). We then ran MPT on these three sets. The results in Table 4.18 show clearly that while recall is not affected, precision drops steeply given more context. Remark that precision on the single sentence test, while lower than that we reported on the English single sentence benchmark set, can still be said to be at a comparable level. We have no doubt that had we had a full article English benchmark set, we would there too have seen a comparable drop in precision attained by the systems we evaluated.

In van den Heuvel (2003) an example was given of Dutch MPT output where the system suggests *olieproductie* (oil production) for the input word *liposuctie* (liposuction). These have an LD of 4. In order to find out what the LD permitted by the Dutch MPT is, we measured this for the correction candidates returned on Metro2. On this list the maximum LD observed was 7, as can be seen from the examples in Table 4.19. This explains the low precision attained by the system: its dictionary is too limited and it searches too far. The latter seems a consequence of applying too many point-edits. Turning *godfather* into *hoofdader* requires

System	Context	R	P	F
true scores: overall				
MPT	1 sentence	0.659	0.444	0.530
MPT	3 sentences	0.659	0.249	0.362
MPT	full article	0.659	0.081	0.143

TABLE 4.18 MPT: effect of context size on overall recall (R), precision (P) and F-score (F).

a substitution of the first character, an insertion after the second, a transposition of the third and fourth and substitution of sixth and seventh into a single character. This seems an overly powerful correction mechanism, most likely due to the application of phonetic rules, which van den Heuvel (2003) tells us the Dutch MPT applies. The resulting great scope in LD stands in contrast to what the permitted LD in the English version of MPT appears to be. We found that 4 is apparently the maximum allowed. As can be seen from the examples, even this is quite a distance for words as short as 4 characters (out-of-vocabulary foreign currencies, all), especially in combination with multiple point-edits.

We think the above amply demonstrates that in evaluating spelling error detection and correction systems measuring recall only will not do because it does not tell the full story. Only by measuring recall in combination with precision can particular aspects of particular systems be brought to light. We devote the best part of the next chapter to further investigating issues involved in evaluating spelling error detection and correction systems. To conclude this chapter on evaluation, we now summarize our findings.

4.5 Summary

In this chapter we have established that:

- the correction mechanism we propose can resolve virtually any type of error encountered in a real-world corpus. The few types of errors it cannot resolve typically involve higher LDs and multiple errors. These are very rare in keyboard-input text.
- using a smaller alphabet, i.e. an alphabet not including character trigram values, results only in a minimal loss of performance.
- performing correction not only on the isolated word tier, but also looking at the immediate context has only a slight effect on overall recall but improves the best-first ranking of the CCs.
- the corpus derived lexicons can to a certain extent be cleaned in an unsupervised way

Input	Output	LD
MPT: DUTCH		
Presley	Porselein (Porcelain)	5
Selassi	Easy	5
Sydney	Ziende (Seeing)	5
Viktor	VITO	5
godfather	hoofdader (mother lode)	5
Applegate	Kabelgaten (Cable holes)	6
Noord-Ierse (Northern-Irish)	Ordners (Files)	6
Zuid-Spaanse (Southern-Spanish)	Uitspansel (Firmament)	7
MPT: UK + US		
rupiah	repaid	4
ecus	issues	4
fpritns (forints)	fronts	4
hrivnia	hiving	4
hryvnia	triennia	4

TABLE 4.19 MPT output: some examples of overly high LD correction candidates for Dutch and English.

- for the full task of detecting and correcting typos within running text we have seen that precision keeps rising the more information is provided to the system, i.e. the lower the frequency cut-offs used
- with manually set Zipf Filter thresholds this results in loss of recall through crowding by incorrect variants included in the lexicon. This calls for more rigorous corpus normalisation during pre-processing and for applying absolute correction.
- using corpus-derived thresholds we can manipulate the levels of precision and recall and can force the system to focus on the one. This is to the detriment of the other, but a good balance can be achieved.
- given a realistic amount of context, i.e. typos within their full newspaper article, we reach a level where for every error removed, only one correct word would be replaced, if the system were run in a fully unsupervised, automatic fashion. This we have shown to be the case for Dutch, which we have also shown to pose greater challenges to spelling error detection and correction systems than English.
- for both languages English and Dutch, TISC outperforms the state-of-the-art systems available today. For English we showed that this is the case for ASPELL, ISPELL and MPT, for Dutch: ISPELL and MPT.

On the basis of these findings we conclude that Text-Induced Spelling Correction is a viable alternative to the approaches to spelling error

detection and correction as embedded in the state-of-the-art systems available today.

In the next chapter we discuss TISC in light of the state-of-the-art as proposed in the literature. We also discuss by what means we believe we can further enhance TISC's recall without affecting its level of precision. We finally examine the question what constitutes fully automatic spelling error detection and correction and whether we have achieved that goal. More than anything else, our focus is on how and why spelling error detection and correction systems should be evaluated and what the proper metrics to that end are.

5

Evaluation of the evaluations

In this chapter we first discuss TISC in comparison with a state-of-the-art spelling correction system as presented in the literature. We show that even if the same metrics are used in evaluation, real comparison between two systems remains problematical if the actual data used are not available, even if more is known about the data than a rather minimal description. We propose ways of describing the data so that more of the information necessary for eventual replication is made available. This leads to an exploration of the role we see for TISC and whether the system is capable of filling that role. We next discuss two preprocessing techniques we might fruitfully employ and which should further enhance TISC's performance. Then we examine a number of questions concerning evaluation methodology and reliability of evaluation sets and evaluation metrics. We consider the question when a system would be fit for performing fully automatic spelling error detection and correction and whether the system we propose has advanced the field towards fully automatic spelling error detection and correction. We conclude the chapter with an evaluation of whether another metric, the area under the curve, which has in recent years gained acceptance and popularity in Machine Learning, would serve as well as the F-score to evaluate spelling error detection and correction systems.

5.1 Related research: TISC in comparison with the state-of-the-art in the literature

We compare the English version of TISC as a spelling checking and correction system with state-of-the-art-systems proposed in the literature.

Brill and Moore (2000) report an excellent accuracy, 98.8%, on a test set of 2000 erroneous word/correction candidate pairs. Accuracy is another of the metrics derivable from the confusion matrix we described in Chapter 4 and is defined as follows:

$$Accuracy = \boldsymbol{A} = \frac{TP + TN}{P + N}$$

Now, by measuring performance on a list containing only erroneous word forms, the terms TN and N in the definition reduce to zero, which leaves $\frac{TP}{P}$, which shows that what was being measured is recall = $\boldsymbol{R}$. This is what we measured and reported on in the first series of evaluations, which should facilitate comparison between the systems. We shall here further refer to the scores obtained by Brill and Moore (2000) as recall as this removes any objections that have been raised in the literature against the use of accuracy as a performance metric. (Provost et al., 1998).

What is effectively measured by recall in these kinds of tests, is the ability of a correction system given a typo, to produce its correct form, regardless of the means used to arrive at the correct form. We believe this is a valid and necessary measurement. If a system cannot handle more than for instance single character Damerau edits, it will not be able than correct more than about 90% of the natural distribution of the typos we have observed in the Reuters RCV1.

The score attained by the noisy-channel based approach proposed by Brill and Moore (2000) begs the question what kind of errors it failed to correct. A discussion of these residual errors appeared in Toutanova and Moore (2002), where Toutanova argues the residual errors can mainly be seen as examples of phonetic errors and adds a separate error model for word pronunciations, thereby achieving even better performance. Both studies report on the recall of their noisy-channel based systems under ideal conditions, i.e. with all the correct forms for the words to be corrected present in the dictionary. This means that in the terminology we have used throughout, they report upper bound recall and upper bound best-first ranking recall. Though not explicitly stated, the reason why the systems were tested with a dictionary present and not without as we did in our first test, must have been to study the extent to which what we have come to call 'crowding' affects performance. The strategy employed by Brill and Moore (2000) is that they try to 'find the word $w \in$ of D (the dictionary) that is most likely to have been erroneously input as s'. Their error model is the noisy channel model and is based on the probability that when a user intends to type the string α, (s)he types the string β instead. This probability is derived from a training corpus of errors, in their case, 80% of a 10,000 word corpus of what they say are common English spelling errors, paired with their correct spelling. The approach they take is to have their system return an n-best list of

correction candidates according to the error model and then to rescore these candidates by taking into account the source probabilities.

In Subsection 4.2.2 we have evaluated TISC in the same way as was described in Brill and Moore (2000).

Likely due to space limitations, Brill and Moore (2000) provide very little information on their 10,000 typo/correction pair list. We wanted to be able to compare our approach to theirs. The list is considered proprietary by Microsoft. This precludes direct comparison by testing TISC on the same evaluation set. In order to nevertheless open up the possibility of comparing both approaches, the authors provided us with the next best thing: the list with the actual words masked in such a way that the typo pattern is apparent but the actual underlying words are not. This was done by replacing all matching characters by *M*, marking a deletion with *D*, an insertion by *I* and a substitution by *S*. Transpositions of two adjacent characters were marked with a single *T*. The pair *actress*/*acress is then represented by the string: MMDMMMM. Note that especially with pairs involving multiple edits, various readings may be possible. We do not know exactly which readings their transcription system was biased to. We transcribed our own RCV1 list in the way set out in Appendix 1. We do not think that our conclusions regarding the differences between the composition of the lists will be unduly biased by possible transcription differences.

The full Brill/Moore list contains 9,774 items of which there are 1,463 unique patterns. The 12,094 items RCV1 list contains 1,163 unique patterns. The larger diversity in the Brill/Moore list is no doubt due to the greater share of multiple errors in it. It is immediately obvious that substitution errors predominate.

Some patterns recur with great frequency. The three most common patterns in the Brill/Moore list as displayed in Table 5.1 would actually fit the *-ise(d)/ize(d)* variation. Remember that we did not mark allowable variation as being erroneous in the RCV1. However, as the examples in Table 5.2 show, the patterns may represent very diverse errors, not mere systematical variation. In the table we have marked recurrent erroneous substrings. It would be these that would be learned by the noisy channel approach taken by Brill and Moore (2000). These, however, would not present undue problems to TISC, without any training and thus without the need for training material. We actually think the major strength of the system proposed by Brill and Moore (2000) lies in the fact that it can handle multi-point multiple errors well. Its weaknesses may very well be comparable to the ones we have detailed for MPT and ASPELL, namely loss in precision. Then again, that might be mitigated by the more specific error-targetting allowed by modelling

Rank	Brill/Moore	frq.	RCV1	frq.
1	MMMMSMM	258	MMMDMMMM	126
2	MMMMSM	251	MMMMMDMMM	125
3	MMMMMSMM	214	MMMMDMMMM	118
4	MMDMMM	205	MMMMDMMM	117
5	MMMSM	201	MMMMMDMMMM	107
6	MMMMSMMMM	195	MMDMMMMMM	106
7	MMMSMMMM	181	MMMDMMMMM	105
8	MMMSMMM	178	MMDMMMMM	104
9	MMMSMM	166	MMDMMMM	104
10	MMMMSMMM	164	MMMMMDMM	102
11	MMMMMSMMM	151	MMMMIMMMM	98
12	MMMDMMM	148	MMMMIMMM	98
13	MMDMMMM	146	MMMMMMDMM	97
14	MMMMMSM	142	MMMMMIMMM	96
15	MMMMMMSMMM	132	MMMMDMMMMM	95
16	MMMMMMSMM	132	MMMMMMDMMM	92
17	MMMMMMSM	115	MMMDMMM	90
18	MMSMM	110	MMIMMMMMM	87
19	MMMMMSMMMM	109	MMMIMMMMM	85
20	MMIMMMM	108	MMDMMM	85
21	MMIMMM	104	MMMMDMM	84
22	MMMDMMMM	91	MMMDMMMMMM	82
23	MMSMMM	89	MMMMMIMMMM	81
24	MMMSMMMMM	85	MMMIMMMM	81
25	MMMMDMMM	81	MMMMMMMDMM	80

TABLE 5.1 Comparison of top 25 patterns and their frequencies in left: Brill and Moore (2000) and right: RCV1 list.

longer substrings and the positional information employed. But this we do not know: the authors did not in any way test this, by measuring only their system's recall.

In the Brill and Moore (2000) list we find the first deletion pattern on rank 4, whereas in the RCV1 list we observe nothing but deletions until rank 11, where we find an insertion pattern, after which we observe only a mix of deletion and insertion patterns in the top 25. In the RCV1 the first transposition pattern: MMMMTMM with frequency 78, was observed at rank 28. It is only at rank 91 in the RCV1 list that we find the first substitution pattern not involving capitalization: MMMMMSMMM with 38 occurrences. Note that this pattern corresponds to

Correct / Typo	Correct / Typo	Correct / Typo
abnormals / abnornals	cigarette / cigaratte	postponed / postpgned
abundance / abundence	communist / commumist	published / publiched
adherence / adherance	company's / compahy's	retreated / retreited
affiliate / affillate	jubilance / jubilence	scheduled / schedeled
algorithm / algorythm	necessary / necesaary	secretary / secrerary
aluminium / alumimium	obstacles / obstables	severance / severence
ambit*ious* / ambit*uous*	operating / operaring	something / sometning
anci*llary* / anci*liary*	perimeter / perimiter	stratagem / strategem
assid*uous* / assid*ious*	permanent / permament	sybaritic / sybaratic
automatic / automotic	permitted / permiited	utilities / utililies
auxi*liary* / auxi*llary*	pessimism / pessisism	vigilance / vigilence
awareness / awaremess	plausible / plausable	worldwide / worldvide
challenge / challange	plummeted / plummtted	

TABLE 5.2 38 correction/typo pairs from the RCV1 list fit the pattern MMM-MMSMMM. Underlined: recurrent *-ance/-ence* variation. Overbraced: recurrent *ious/uous* variation. Underbraced: recurrent *llary/liary* variation. Note the bidirectionality of all three recurrent types of variation.

the pattern on rank 3 in the Brill-list. Numbers two and one of their list follow at rank 113 with frequency 34 and rank 158 with frequency 25, respectively. So it is not that we have not observed the same patterns. It is that we have observed other patterns, involving other types of transformations, more frequently. As a matter of fact, the lists share 453 patterns or 20.85%.

What the table shows most clearly is that while both approaches have the stated aim of performing spelling correction, the actual definition of spelling correction handled differs widely.

Table 5.3 lists the error type statistics we obtained from this list. We see there are a few types of errors not present in the list: run-ons and splits, especially. We do not know whether capitalization errors are present, but given the first character statistics, we doubt it. In terms of LD-distribution we see a fairly normal slope, which is nevertheless less steep than what we observed in the RCV1. This means there are somewhat less LD 1 errors than what we observed, but more LD 3 and higher errors. The main difference lies in the share taken by substitution errors: these represent nearly half of all the errors. This is very different from what we observed, in the RCV1 substitution errors occur least often of the four basic categories. We also see that in the Brill/Moore list there are about 10% more multi-point multiple errors. In our list we have examples such as: *vehicle* *vechile, *restaurants*: *restaraunts,

Category	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	Total	%
deletion	1,818	20	1				1,839	18.815
insertion	1,250	5					1,255	12.840
substit.	4,167	448	24	1			4,640	47.473
transpos.		304					304	3.110
multiple		1,024	254	53	6	1	1,338	13.689
multisingle		305	84	8			398	4.072
1st ch. del.	(34)	(14)	(6)	(6)			(60)	(0.614)
1st ch. ins.	(3)	(1)	(1)				(5)	(0.051)
1st ch. sub.	(55)	(30)	(16)	(10)	(1)		(112)	(1.146)
Total	7,235	2,106	363	61	7	1	9,773	
%	74.023	21.547	3.714	0.624	0.072	0.010		100

TABLE 5.3 Statistics of the error categories in the Brill and Moore (2000) typo/correction list. Counts between brackets are subsumed by the parent category.

*restaruants. Note that the way we count these, non-adjacent transposition errors are counted as multi-point multiple errors.

The high number of substitution errors is explained by Dr. Moore as follows (personal communication):

> It appears, anecdotally, that the vast majority of errors in our data are cognitive errors. It seems to me that what type of errors predominate would depend on the source, and in the case of published text, the point in the production chain where the errors occur. Microsoft is interested mainly in errors that occur at the beginning of the chain, during composition, since this tends to be where Microsoft products (Word, PowerPoint) are used. I would expect that cognitive errors would predominate there. In the case of published news text, which I presume the Reuters corpus is, I believe the standard production chain is composition ⇒ copy editing ⇒ typesetting. In this process, most cognitive errors should be corrected by copy editing, and it may be that most of the errors in the final product are actually introduced in typesetting. It would not surprise me if deletions predominate here, since typesetters strive for speed, which would tend to lead to skipping characters or failing to hit the key for a character hard enough to register.

In Table 5.3 we present the breakdown of the nearly 2,000 item test set employed by Brill and Moore (2000). This was obtained from the full list by taking every fifth item. We see that this strategy produces a nearly identical error distribution as we saw in the full list, which allows us to conclude that the test set is a reliable subset of the full list, which should allow for a valid estimation of the system's recall.

It is only by virtue of this statistical breakdown in error types that we have been able to see that the problem addressed by Brill and Moore (2000) is very different in nature than the problem we have been addressing throughout this work. The information available in the paper

Category	LD 1	LD 2	LD 3	LD 4	LD 5	LD 6	Total	%
deletion	364	2					366	18.731
insertion	228	1					229	11.720
substit.	852	78	1				931	47.646
transpos.		62					62	3.173
multiple		216	61	11	1		289	14.790
multisingle		61	15		1		77	3.941
1st ch. del.	(8)	(2)	(1)				(11)	(0.563)
1st ch. sub.	(9)	(6)	(2)	(5)			(22)	(1.126)
Total	1,444	420	77	12	1		1,954	
%	73.900	21.494	3.941	0.614	0.051			100

TABLE 5.4 Statistics of the error categories in the Brill and Moore (2000) test set. Counts between brackets are subsumed by the parent category.

concerning the 10.000 typo/correct word list, which amounts to the statement: 'a list of 10,000 common English misspellings', did not allow us to deduce that their system is meant to be able to correct highly deformed word strings. We address primarily the naturally occurring mild cases. Both MPT and ASPELL cater for the orthographically challenged. We think that our evaluations have conclusively shown that this has a heavy cost as regards a system's precision. The authors have not measured the precision of their system, but we think the system by Brill and Moore (2000) with its very powerful correction mechanism is also in danger of allowing the scope of the search in terms of LDs covered to be too wide.

Catering for poor spellers further seems to inevitably entail interactive spelling help during composition. Interactive use entails that the writer views the context, which means the system can be limited to isolated-word correction. Hence we have seen no further research in context-sensitive non-word error correction and no regard has been given to context and its possible contribution to correct ranking. Finally, we now understand why no advance has been made towards fully automatic, non-humanly supervised spelling correction.

On the basis of TISC's performance in the evaluations, we should conclude that it probably cannot in its present implementation perform equally well as the Brill and Moore (2000) system as regards multi-point multiple errors. Its strength would lie in the unsupervised detection and correction of those typos which either escaped the copy-editing process or were introduced during type-setting. It could be put to good use just before going to press, as a quick final check on the copy. So TISC would fill an altogether other niche than either MPT or ASPELL.

Powers (1997) uses a copyleft book to test the system he proposes for the detection of confused words. Of a limited set of confusables, his system detected 6 instances in this highly-edited text. When reading

the book's copyrighted version by Penguin Books we had seen and noted one typo, on page nine: '[Graham] Bell himself, as an *elecution teacher and gifted public speaker, [...]'. A logical step was therefore to run the electronic version of the text through TISC and see whether our system could be fruitfully applied to a highly-edited text to weed out editorial oversights at little cost.

The copyleft book is Bruce Sterling's 'The Hacker Crackdown - Law and disorder on the electronic frontier', which is available on the web[1]. The book describes the history of the tele- and datacommunications industry and the attendant use and abuse that is and can be made of cyberspace, the 'parallel world' the industry has spawned. The book is not overly technical, but nevertheless and unavoidably contains a certain degree of domain-specific terminology. It actually contains a transcript of 6 pages of telecommunications administrative technospeak and directory listings of bulletin board systems, full of the kind of cryptic abbreviations one finds in directories. The book certainly presents a challenge to a spelling checking and correction system.

We ran TISC with settings which should give good recall. We used the NYT-BNC lexicon with cut-off frequency 5. We set the input text frequency threshold at 5. We used the CICCL-derived list for absolute correction, i.e. we still used our fully-unsupervised system, although we might have added or used our Reuters RCV1 list of manually acquired typos. We set the LD at 2 and percentiles at 80%.

The book consists of 120,717 word tokens, good for 13,855 word types. Of these, TISC returned a manageable list of 397 items. Most of these were false positives. This is probably unavoidable given text such as

> Then the two groups conflated into the Legion of Doom/Hackers, or LoD/H. When the original "hacker" wing, Messrs. "Compu-Phreak" and "Phucked Agent 04," found other matters to occupy their time, the extra "/H" slowly atrophied out of the name;

TISC managed to validate the first hacker name on the basis of input text evidence. All things considered, TISC's offering of CCs for the second name was not so far off the mark: shucked, chucked, ducked, Tucked, fucked. The ranking was off: the lexicon does not have the word, let alone the bigrams; it is a hapax in the text. A lot of the false positives, however, are compounds which through bad tokenization had lost their hyphen in those cases where the LPC was at the end of the original line and the RPC at the beginning of the next. This is something we need to look into in the future. An example of these

[1] http://www.dcs.gla.ac.uk/SF-Archives/Bruce.Sterling/The.Hacker.Crackdown/

is: *freespirit*, CCs: free-spirit, free spirit, free spirits, freer spirit, tree spirit. The space-dash-nothing variation rule did not fire here through the presence of the last three CCs. The fact that TISC draws attention to these cases is actually correct.

More importantly, the list proved to contain 22 (21 + *elecution) in-original-text true positives: typos which had escaped the author's and editor's attention. We list them in Table 5.5. While the correct type *harassed* does not appear in the text, we do find: *harassing* [1], *harassment* [4] and *harass* [1]. We think the status of *mimickry as a typo may be disputable, but it is not in the dictionaries we consulted. Further in the text we find *mimicking* [2] and *mimic* [1]. The type *nineteenth* appears only in compound form: *nineteenth-century* [7] and *unprecedentedly* only as the adjectival form *unprecedented* [8]. Cases marked by '-P' also survived the editing process prior to publishing by Penguin Books.

This result shows that TISC is capable of filling the niche we see for it as a system that allows for a quick final pre-publishing check. We believe this also shows that our system would be a valuable addition to any professional text writer's toolbox.

5.2 How to further boost TISC's performance?

5.2.1 Related research: useful corpus preprocessing techniques

We have seen that TISC's recall in real-world tests on English lags slightly behind that of ASPELL and MPT. We have shown by evaluating the correction mechanism on its own that it is capable of correcting virtually any type of error within the bounds of the types we have observed in the RCV1. We lose recall due to the noise in the lexicon which, it was found, it is not always possible to circumvent with the techniques we have applied. In what follows we will describe two techniques that by their application in the corpus preprocessing phase prior to deriving TISC lexicons and COOC-tables would most certainly help to reduce the level of noise incorporated and would thereby allow TISC to achieve better recall, if not even higher precision as well. We have in Chapter 2 described the minimal preprocessing we have applied prior to evaluation. We here describe proven techniques for largely eliminating capitalization and accent related noise. Both are in themselves regarded as simple problems. However, both techniques have been proven to aid in NLP tasks. Lita et al. (2003) convincingly argue for normalizing capitalization to be an essential part of preprocessing. The distribution of the Dutch TWC corpus contains a set of corpus normalization tools.

Correction	Original text typo in context
anomalous [2]-P	one of the many *anomolous outcomes of the Hacker Crackdown .
anomalous [2]-P	only one Barlow , and he was a fairly *anomolous individual .
around [59]-P	It will be an impossible millstone *aroung the neck of tomorrow's organizations .
becoming [6]	The puzzle *becaming much stranger some five minutes later .
besieged [0]	the tiny band of techno-rat brothers (rarely , sisters) are a *beseiged vanguard
beyond [10]	altered his consciousness *beyone the ability to
Cincinnati [1]	Racketeering Bureau conduct "Operation Sundevil" raids in *Cincinnatti, Detroit, Los Angeles
considered [39]	began by saying he *consided computer-intrusion to be morally wrong
entertainingly [1]	which always looked *entertainly wacky , but certainly harmless enough
graffiti-tagged [0]-P	Some passing stranger has *grafitti-tagged this door
graffito [0]-P	it bore a huge , badly-erased , spray-can *grafitto around its bottom
harassed [0]	and they will be *harrassed or arrested.
hierarchy [2]	the central switching stations , which are ranked in levels of *heirarchy , up to the long-distance
hierarchy [2]	a symbolic , deliberate slap in the face for the Apple corporate *heirarchy .
mimicry [0]	certainly not cruel *mimickry , one-upmanship and outrageous speculation
nineteenth [0]	" pioneers " of the *nineteeth and twenty-first centuries
top-to-bottom [0]	the Bell employee corps were nurtured *top-to-botton on a corporate ethos of public service .
non-hierarchical [1]	The Internet is decentralized , *non-heirarchical , almost anarchic .
unmistakable [2]	she tells him with *unmistakeable sixty-thousand-watt sincerity .
unmistakable [2]	gives off an *unmistakeable air of the bohemian literatus
unprecedentedly [0]	AT&T suspected there might be shakedown problems with the new and *unprecedently sophisticated System 7 network .

TABLE 5.5 Twenty-one extra typos present in the original text of the Hacker's Crackdown and detected by TISC. Text frequency of correct types between square brackets. Items marked by -P are also in the copyrighted Penguin paper edition.

How and why to go about text normalization, at least for Dutch, is in detail set out in Ordelman (2003). Our own system would undoubtedly benefit from applying these tools prior to deriving lexicons from the corpora.

Accent restoration Yarowsky (1994) categorizes the problem of accent restoration in the same class of errors as capitalization restoration. Both are part of the class of closely-related problems which further includes word-sense disambiguation, word choice selection in machine translation and homograph and homophone disambiguation. An overview of corpus-based approaches to accent restoration is provided in Yarowsky (1999). It was shown by Simard and Deslauriers (2001) that accent restoration can be performed with over 99% accuracy for French.

A language independent solution is offered by Mihalcea and Nastase (2002). Instead of relying on dictionaries and other resources such as Part-Of-Speech taggers, which may well not be available for languages for which very few such resources exist, the task is learned on the letter level by Machine Learning methods from a modest size corpus of raw text containing diacritics. The features required for the Machine Learner are directly derivable from the corpus in that they consist of nothing but a window, i.e. a context, of surrounding characters. They found a window of five characters preceding and 5 characters following the ambiguous character works best. For four languages for which few resources exist they collected corpora containing only between about 1.5 and 3 million words and reached over 98% accuracy.

Their method is based on the 'forgetting examples is harmful in language learning' observation due to Daelemans et al. (1999) and therefore employs Memory Based Learning as the chosen Machine Learner.

Mihalcea and Nastase (2002) also provide an overview of the diacritics used in 38 European languages. They conclude that only English has no accent restoration problem. Dutch, on the other hand, they list as having by far most diacritics of all: 21. All those listed involve combinations of vowels with diacritical marks. This points out to us that the alphabet we used in this study may not have been complete. The 12 diacritics we incorporated in it were the ones observed within the Dutch corpora we employed, i.e. the ILK and TWC corpora.

Capitalization errors Whether capitalization errors pose a problem very much depends on the application. In many applications case differences are obliterated by upper- or lowercasing everything. Badly capitalized words can be seen as a specific type of substitution error and are handled as such by the system we propose.

However, we are sure our system would benefit from applying **truecasing**, defined by Lita et al. (2003) as the process of restoring case information to raw text. Besides enhancing rEaDaBILiTY, truecasing improves the quality of case-carrying data, bringing into the picture new corpora originally considered too noisy for various NLP tasks. It can be used as a normalization tool across corpora in order to produce consistent, context-sensitive, case information. The statistical model Lita et al. (2003) propose captures local context through a word trigram language model. Making use of the collection of local contexts for a given ambiguous word, the case label is decided at the sentence level on the basis of a Hidden Markov Model from which the highest probability state sequence can be computed to yield the desired case information. The system is evaluated on three NLP tasks. First, as regards named entity recognition a 26% F-score improvement is obtained. Second, in automatic content extraction from text obtained by automatic speech recognition, which was therefore uncased prior to applying truecasing, mention detection is improved by a factor of eight. Third, legibility of machine translation output, which is also uncased, is improved by 80% as measured by the relevant metric.

Chelba and Acero (2004) report on how to adapt a Maximum Entropy Capitalizer by means of relatively small quantities of in-domain data. As is common practice in statistical approaches to NLP tasks, the data used to train the system are taken from a disjoint corpus or corpus than the data used to test. It is here shown how to use small quantities of still disjoint, but same domain data as the test data to further improve automatic capitalization.

5.3 Evaluation of the evaluations

What constitutes a reliable evaluation set? We have above shown that the test set used by Brill and Moore (2000) is representative of their full data set. Apart from the question whether a test set is representative of the data, one may wonder what size a test should be to be representative of the various types of errors one may find and thus to constitute a reliable test set.

The aim of the evaluations on 10 sets of 2000 randomly chosen typos was to answer that question. Often in the literature systems are compared and evaluated on far smaller test sets. By the 'order of magnitude' rule of thumb we proposed in Chapter 2, for a test set to be representative over the range of LDs 1, 2 and 3, the set ought to contain a ratio of one thousand typos with LD 1, one hundred LD 2 cases and ten LD 3 cases. The LD-distribution of our ten sets was shown in

Table 4.5. What we see there is that the randomly chosen sets pretty well conform to the rule-of-thumb. We also noted a very low standard deviation in the performance attained by TISC on those tests. We take the low standard deviation to indicate a high degree of homogeneity in their composition. We conclude from this that given a random selection of 2,000 cases from a sufficiently large sample, i.e. 10,000 items as in Brill and Moore (2000) or 12,094 RCV1 items, should give a reliable test set.

Circumstances may preclude collecting test sets of this size. We have above discussed the problems associated with collecting real-world non-word errors from published newpapers. A typical daily edition yields 2 non-word errors on average. It would then require three years' worth of proofreading the daily edition to collect a sample of sufficient size, by our own standards. A possible way out would be to collect a random sample of 2,000 typos from a corpus, as we did from the Reuters RCV1, with limited context. For a further subsample of, say, 200 randomly chosen typos, to collect and proofread the full articles. Then, to provide a description of the data along the lines we have done here. Finally, to evaluate as we have done on the lists for English, and on the texts for Dutch. This would provide the full picture, for a particular language, for a particular spelling detection and correction system. We now see that doing this would have made our own evaluations for both languages more complete.

What constitutes reliable evaluation data? Bigert (2005) proposes a system for the fully automatic evaluation of NLP systems. Chapter 10 in Bigert's recent PhD-dissertation is devoted entirely to the unsupervised evaluation of three Swedish spelling checking systems: Stava (Kann et al., 2001) and the Swedish versions of ISPELL and MPT. He advocates the use of artificially created typos and provides software, called MISSPLEL, to introduce these into clean text, as well as software for the automatic evaluation of NLP-systems' output, called AUTOEVAL (Bigert et al., 2003). The main motivation behind this is to reduce the expensive and time-consuming manual labour involved in evaluation.

Bigert (2005) (p. 88) writes:

> We have chosen to use [Damerau type errors] to keep the evaluation procedure language independent. In the light of the previous work [i.e. Agirre et al. (1998) and Paggio and Underwood (1995)], our contribution is a detailed and thorough investigation of Swedish spelling checkers as well as an open-source test bed for unsupervised evaluation of spell checkers, applicable to any language and text type.

In what follows we show that:

- introducing only Damerau-type errors does not allow for a complete measurement of spelling correction systems' strengths and weaknesses
- the evaluation procedure is not language independent
- the metrics borrowed from the previous work do not include measuring the systems' precision and therefore provide an incomplete assessment

These three criticisms translate into an incomplete evaluation of the Swedish spelling checkers and an open-source test bed that cannot provide what is claimed it provides.

Artificially created Damerau-type typos

We have three objections to the artificially created Damerau-type typos. First, Bigert (2005) introduces typos in what he assumes is clean text: 14,000 word tokens, about 1,000 sentences, from a Swedish corpus. The corpus was the Stockholm-Umeå corpus (further abbreviated as SUC). On page 89, however, he writes:

> The word lists for MISSPLEL were built from the SUC corpus while none of the spell checkers obtained dictionaries or other information from SUC. Thus, the SUC defined whether or not a word was misspelled.

We see a potential problem in this quote. On page 54, Bigert writes:

> The dictionary required to determine if a word is existing or not can be built unsupervised from large amounts of text.

He adds that this was exploited for the spelling checker evaluations he performed. Now we have seen in Chapter 2 in the present work that a word type list derived from a large corpus may contain more than 20% of typos. Included in this number are the most recurrent typos in the language. So, in introducing artificial errors and checking for the resulting string's existence in such a list, all recurrent errors which happen to be produced by the error generator, would be validated by this list and not be introduced. This would bias the system towards introducing the less recurrent types of error.

Second, MISSPLEL, the program to introduce typos into text, can be configured to introduce 4 types of errors: Damerau-type errors, split compounds, competence related errors and syntactic errors. However, in his evaluations Bigert introduces only Damerau-type errors. MISSPLEL thus introduces only LD 1 errors: single character deletions, insertions or substitutions, or transpositions of two adjacent characters. Now, STAVA and ISPELL handle precisely, and only, the LD 1 types of errors MISSPLEL introduces. MPT, as we have demonstrated in Chapter 4, is fully capable of handling higher LDs. This strength of MPT is not

Results	STAVA	ISPELL	MPT
error coverage	92.2%	97.3%	95.5%
'precision'	97.2%	92.8%	89.4%

TABLE 5.6 Evaluation results of three Swedish spelling correction systems due to Bigert (2005). Error coverage and 'precision' due to Agirre et al. (1998). Error coverage is our overall true recall on detection. 'Precision' is our overall true recall on correction.

at all measured. In Table 5.6 we reproduce the evaluation results on the three Swedish systems tested by Bigert. ISPELL appears to perform best on recall. In light of its performance on English and Dutch as presented in Chapter 4 in the present study, we think this is an artefact of using artificially introduced typos of the Damerau-type. What this shows is that ISPELL's correction strategy is robust towards the artificially introduced errors. MPT most likely employs language-specific, probably noisy-channel based, correction strategies which would not be robust against artificially introduced typos. However, to prove this would require testing the systems on real-world typos. What gave ASPELL and MPT the edge in recall in our tests, was the fact that they can handle higher LD typos well. These were not introduced by MISSPLEL. The system proposed by Bigert also does not measure the attendant loss in precision given a system's great recall capabilities on higher LD typos, which we have demonstrated for both MPT and ASPELL. That, however, is due to the metrics he uses, which go back to prior work by Agirre et al. (1998) and Paggio and Underwood (1995). We further discuss the metrics in more detail.

Third: by default, MISSPLEL assigns equiprobability to the four Damerau types. In Chapter 2 we have demonstrated that, first, the natural LD-distribution of typos follows a Zipfian distribution and is in no way restricted to LD 1. Second, that the types of errors are not evenly distributed: that for typos we find about three times as many deletions and two and a half times as many insertions than both transpositions and substitutions. Bigert concedes in his concluding remarks that it is difficult to determine which types of errors to introduce and to what amounts, which leads to a rather strange circularity in his thinking in that he states this could be learned from error annotated corpora: it was the unavailability of error annotated corpora which prompted the introduction of artificial errors in artificial quantities in the first place.

Language dependency of typographical errors The assumption that performance errors, i.e. typographical rather than cognitive errors, are language independent is not valid. As discussed in Chapter

2, Pollock and Zamora (1983) have shown that typos are character frequency related more than anything else. Character frequencies differ per language, Dutch e.g. uses the k far more often than English. This fact is, for instance, being exploited to determine what character encoding a web browser should display a specific web page in given the language is not specifically identified in the page's code (Li and Momoi, 2001). The LD-distribution of typos is probably language-independent. The actual transformations occurring will not be. In order to accurately model these, an in-depth study of the types of error actually occurring in a particular language along the lines of Pollock and Zamora (1983) will be required.

Confused evaluation metrics

As we discussed and showed graphically at the beginning of Chapter 4, the aim of a spelling detection and correction system is to maximize the overlap between the items retrieved and the set of target items, the typos within the text. Full overlap would give 100% recall and 100% precision in the sense of the terms as they were were defined by van Rijsbergen (1975) and used throughout this work. In what follows the terms will recur in reference to subsequent work in which they were used with very divergent definitions, giving rise to possible cause for confusion. In our evaluations we have tried to capture a system's overall performance, i.e. the performance on both the detection and the correction of typos, in one single measure: the F-score. Besides that, we have then reported the best-first ranking score, again expressed in a single F-score. Besides being practical, we believe we have shown that the metrics of precision and recall have the ability to shed valuable light on detection and correction strategies used in the implementations of which little or nothing is publicly known otherwise.

The spelling error detection and correction evaluation metrics in Bigert (2005) are in fact a mixture of metrics defined by Agirre et al. (1998) and Paggio and Underwood (1995). The metrics **Error Coverage** and **'precision'** are due to Agirre et al. (1998). Error Coverage is defined as 'the amount of errors [the system] detects of all errors in the misspelled text'. **'Precision'** as due to Agirre et al. (1998) is then 'the number of detected errors where the original word is among the suggestions (including errors with no suggestions)'. In fact, Agirre's 'error coverage' is Van Rijsbergen's recall applied to the detection phase. Agirre's 'precision' is overall recall in the correction phase. Agirre et al. (1998) worked with error lists and therefore did not measure precision. The metric **lexical coverage** is due to Paggio and Underwood (1995) and is also referred to as 'recall': 'the degree to which the checker accepts all the valid words of a language (does not produce 'false flag-

gings')'.

It is unfortunate and confusing that the two key evaluation terms, precision and recall, as defined by van Rijsbergen (1975) have thus been introduced into the field of evaluating spelling error detection and correction with quite different meanings.

As is explained in TEMAA (1996) measuring lexical coverage involves creating 'base' lists for the particular language, on the basis of corpora. A base list would contain the top frequency word types of the language. The list is then used to gauge the system's lexical coverage: it is run through the correction system and measured how many of the words are not accepted. We think that given base lists derived from different corpora the actual lexical coverage results obtained are bound to diverge to a great extent and will therefore not result in consistent measurements.

The TEMAA (1996) work complements ISO 9126 (Software Product Evaluation - Quality Characteristics and Guidelines for their Use) and has thereby become an international standard to be followed. This is perhaps why Bigert adopts lexical coverage and error coverage as evaluation metrics. Lexical coverage is measured on the clean text. Error coverage is then measured on the text with typos introduced. There is no relationship between the two measurements: they cannot be combined into one single score as e.g. Van Rijsbergen's recall and precision can be in the F-score. This means that the scores obtained by lexical and error coverage are less informative: we have seen that higher recall affects precision, remember the rather extreme case of ASPELL. By using the measures lexical coverage and error coverage, this connection does not show up. By employing the metrics recall and precision and by employing typos within their natural context instead of using separately and rather arbitrarily compiled lists of typos and valid words, we believe the task of evaluation is beter served by being accomplished in one go.

Bigert (2005) (pag. 63) investigating the impact fully automatic correction might have on a parser's performance writes:

> Using the first suggestion from STAVA we would correctly change about 85% of the misspelled words into the correct word. However, the remaining 15% of the misspelled words would be changed into another, unrelated word.

He next presents results obtained by one of the parsers he evaluates after he effected fully automatic correction and notices a consistent drop in performance, greater the more errors the evaluation file contained, rather than an improvement. This leads him to conclude:

> Evidently the 15% words that are changed into an unrelated word make the processing difficult since the tagger's inherent robustness to misspelled words cannot be used.

We offer another conclusion, which Bigert could not reach since his evaluation does not produce the required statistics. Bigert does not measure precision as defined by van Rijsbergen (1975):

$$Precision = \boldsymbol{P} = \frac{TP}{TP + FP}$$

Bigert does not measure the number of false positives. He therefore fails to see to what extent correct words in the text are erroneously replaced by the automatic spelling correction procedure. We have shown the low precision obtained by isolated-word correction systems on Dutch full text. There is no reason to believe STAVA for Swedish would do any better than ISPELL, ASPELL, or MPT in that respect. Swedish, like Dutch, is a compounding language. Precision for Dutch, we have seen, was well below 10%. Bigert concluded that the ratio of false positives to false negatives lay at the root of the reduced performance of the parser after letting the system correct the text automatically. While there is that, the amount of false positives will be much higher than the amount of false negatives. Their combined effects are what causes the parser's weaker performance.

We fail to see why Bigert did not continue in line with the metrics he used in the evaluation of the algorithm he proposes for detecting real-word errors. He there, as in the parser evaluation, uses van Rijsbergen precision and recall, as we have done. The TEMAA (1996) recommendations were also not adopted by Starlander and Popescu-Belis (2002), who nevertheless refer to the work and state explicitly to want to follow the ISO standards. Their notions of precision and recall are precisely ours, fit for deriving the F-score. Starlander and Popescu-Belis (2002), however, diverge from us in that they do not measure precision at the correction level:

> The capacity of the program to suggest the right correction for the detected mistake must finally be measured. It seems fair to take here into account only the suggestions concerning mistakes that were correctly detected. Otherwise, for wrongly hypothesized mistakes, the suggestion would be necessarily wrong (because the text is correct) and this would penalize the checker twice (once for the previous score [on detection] and once here).

It will readily be seen that we did penalize the system for 'wrongly hypothesized mistakes': to us they are precision errors and we have shown that these constitute an important part of a system's performance. In

that we have not measured the performance on detection alone, we have not unfairly penalized the systems twice. We have shown that it is because the error model employed by the system is too powerful that a lot of these false detections arise. It is therefore futile to want to measure a system's detection capabilities or lexical coverage not taking into account the correction mechanism. In fact we think that in these approaches to evaluation the stress is laid too strongly on the detection side of things. This may have historical reasons, indeed what we in Chapter 1 have called first generation systems such as SPELL performed only detection. So the way of looking at the performance of second generation systems was still strongly influenced by what first generation systems were capable of. TISC is a context-sensitive third generation system. We have indeed compared it to second generation isolated-word systems. By doing this on third generation evaluation terms, by acknowledging the typos' context is important and helps to define the system's level of performance, we have been able to highlight weaknesses in the second generation systems that before were probably not deemed relevant or were taken to be unavoidable. We stand firmly by our approach to evaluation.

5.4 What level of performance is necessary for fully automatic spelling error correction?

In order for a system to be useful for fully automatic use, the system would naturally have to be able to detect and correct more of the possibly very few typos present than it would erroneously replace any of the far more prevalent correct words in the text, as well as the typos it replaces by the wrong correct word. At what level of performance would fully automatic spelling error detection and correction be achieved? We have so far presented evaluation results in terms of recall and precision as combined in the F-score. At first sight, neither of these measures tells us about the possible usefulness of a system for fully automatic, humanly unsupervised spelling error detection and correction. Recall does not measure a system's performance in spelling error detection. A good F-score, we have seen, can be obtained by having a great recall, to the detriment of precision, or by focusing on great precision, to the detriment of recall. Both can naturally also be in balance. But what should one focus on, if one wishes to achieve a level of performance that would be acceptable for fully-automatic use? The F-score alone simply does not tell us that. Yet the information collected to obtain the F-score does tell us more. It is worth pondering what precision really says about a system's performance. If precision is 0.5, half of the

items returned actually correct a typo. This is irrespective of how many of the total amount of typos present are actually corrected. The other half of the items returned replace correct words by other correct words and the resulting text cannot actually be said to have been improved. If precision is higher than 0.5, the system is fit for automatic correction purposes: more typos will have been removed than correct words erroneously replaced. If lower, the system erroneously replaces more correct words than it corrects true errors: the text in actual fact deteriorates.

Seeing the levels of precision obtained by ISPELL or MPT, neither are fit for automatic correction[2]. Neither is TISC fit when it is focused on obtaining the best possible recall. It is fittest when on the other hand it is focused on obtaining the best possible precision. If we look at the precision that goes with the best F-score we obtained for TISC, we see that for Dutch when the errors are effectively presented within the full articles they appeared in, we actually only reach break-even point. If we were to let the system work automatically, it would replace one correct word by another for every typo it would correct. While spelling checking a single full story is a realistic task, viewed from the point of detecting errors within the full text of the whole daily edition of a newspaper or a corpus of a newspaper's output over a couple of years, performance would undoubtedly drop well below this break-even point. So, on that level, we have not achieved automatic correction.

5.5 Evaluating another metric: F-score versus AUC

In the above we have seen that precision is strongly determinative of a system's fitness for automatic correction. We have known at least since Pollock and Zamora (1984) (p. 104) that 'Automatic correction requires a much more precise detection phase than manual correction and, surprisingly, it seems easier to achieve high accuracy in correction than in detection.' Throughout the development of TISC we have used the F-score to gauge progress. The F-score has guided us in deciding on the value of the various components and of changes to the implementation. We here explore whether had we instead used another metric, the **area under the** ROC **curve** or AUC, we would likely have taken different decisions. Bradley (1997) was the first to advocate the AUC. He shows that the AUC is a more sensitive measure than overall accuracy. Ling et al. (2003) formally prove that the AUC is a statistically consistent and more discriminating measure than accuracy. Flach

[2]In Reynaert (2004) we proposed the ratio of errors corrected versus correct words erroneously replaced to be a measure for a system's fitness for automatic correction or FAC. The information given by the metric is actually given in normalized format by precision.

(2003) provides a derivation of ROC space from first principles through three-dimensional ROC space and the skew ratio, i.e. the ratio of negative examples over positive examples. The skew ratio is used to obtain a two-dimensional ROC space from the three-dimensional ROC space. In two-dimensional space different isometric plots are derived depending on the skew. This allows for analysing the metrics accuracy, precision and F-score (and others, irrelevant to our purposes here) in terms of their sensitivity to skew.

The AUC is a single scalar value between 0.0 and 1.0 representing a system's performance. The AUC is a reduction of a Receiver Operating Characteristic or ROC curve depicting performance. A ROC curve is obtained by plotting the systems' False Positive Rates on the X-axis and the True Positive Rates on the Y-axis (Fawcett, 2003). ROC curves are insensitive to changes in class distribution. In that the AUC is derived from these, it too should be insensitive. To calculate the AUC we need to know the True and False Positive Rates of a classifier.

- True Positive Rate = TPR = $\frac{TP}{P}$
- False Positive Rate = FPR = $\frac{FP}{N}$

Remember we have shown in Chapter 4 that TPR = recall $\boldsymbol{R}$.

The formula for the AUC of a single discrete classifier is (as derived from Fawcett (2003)[3]):

- Area under the curve = AUC = $((0.5 * (tpr * fpr)) + (tpr * (1.0 - fpr)) + (0.5 * ((1.0 - tpr) * (1.0 - fpr))))$

How this works is illustrated on the basis of Figure 5.1. The single point P denotes the intersection of a discrete classifier's performance in terms of True Positive Rate and False Positive Rate. From there the line is drawn to the point (0,0) and to the point (1,1). The AUC is the area under these lines. This is measured by the formula as the sum of the areas of triangle A (the first term of the formula), rectangle B (the second term), and triangle C (the third term).

In Subsection 4.3 we have shown that the performance of MPT as expressed by the F-score degrades significantly given the typos are presented with more and more context. More and more context offers more and more scope for a system to report False Negatives. In Table 5.7 we report the AUC for the experiments performed in Subsection 4.3. We see that the AUC remains virtually constant, while the F-score decreases by

[3]We are indebted to Dr. Antal van den Bosch for this derivation

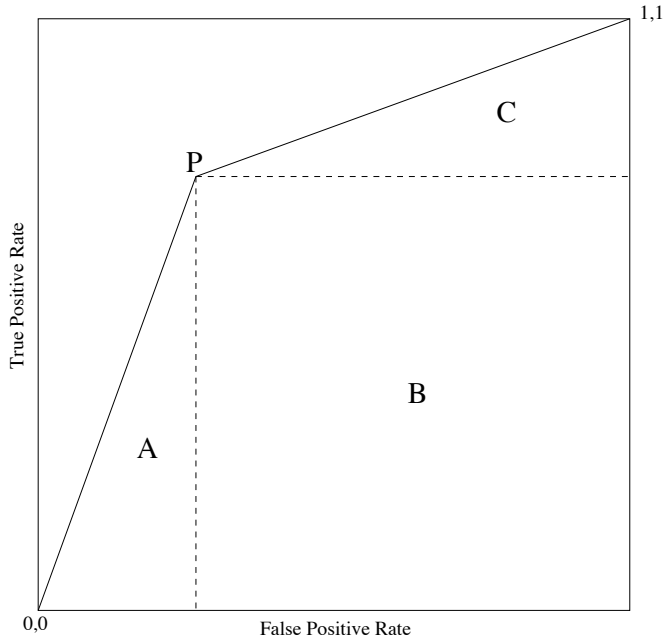

FIGURE 5.1 ROC curve for a single discrete classifier.

System	Context	Tokens	Rec.	Prec.	F-score	AUC
true scores: overall						
MPT	1 sentence	2,840	0.659	0.444	0.530	0.811
MPT	3 sentences	6,843	0.659	0.249	0.362	0.811
MPT	full article	25,749	0.659	0.081	0.143	0.812

TABLE 5.7 MPT: effect of context size on overall recall, precision, F-score and AUC.

almost 40% going from typos presented within a one sentence context to the full article they appeared in. Fawcett (2003) (p. 8) writes: 'class prevalence may change drastically without altering the fundamental characteristics of the class, i.e. the target concept'. This he illustrates with plots showing essentially the same difference between AUC and precision-recall curves under varying class skew as we demonstrate in Table 5.7.

Table 5.8 now allows us to again compare the systems we tested in Chapter 4, though this time in terms of the AUC. We now see that the systems' ranking is different: the AUC defines the TISC version optimized for recall as the best corrector, then comes MPT and only then the TISC version which had the highest F-SCORE. While we see only a 3% difference in terms of the F-score between the TISC version optimized in F-score terms and the version with the highest precision, we observe a 10% drop in terms of the AUC between the AUC-preferred system and the highest precision TISC. This reranking of systems was also noted by Fawcett (2003) (p. 9): 'In some cases the decision of which classifier has superior performance can change with a shifted distribution'. The difference in system ranking between F-score and AUC is due to the fact that the AUC is insensitive to class skew changes and the F-score is sensitive to class skew changes. To again quote Fawcett (2003) (p. 7–8):

> Consider the confusion matrix [...]. Note that the class distribution (the proportion of positive to negative instances) is the relationship of the left (+) column to the right (-) column. Any performance metric that uses values from both columns will be inherently sensitive to class skews. Metrics such as accuracy, precision [...] and F scores use values from both columns of the confusion matrix. As a class distribution changes these measures will change as well, even if the fundamental classifier performance does not. ROC graphs are based upon TP rate and FP rate, in which each dimension is a strict columnar ratio, so do not depend on class distributions.

Consider the following: a human proofreader is given three texts to

proofread: the single sentence Metro evaluation set, the three sentences set and the full article evaluation set. All three contain the same set of 126 typos. Yet the error to token ratio of the first is 4,33%, that of the second 1,80%, while it is 0,47% for the latter. The proofreader has to work his way through 2,840 word tokens, 6,843 word tokens and 25,750 word tokens respectively. The task clearly differs. Likewise for the MPT: its AUC score may very well be the same on all three texts, but the amount of work it presents to the user differs widely: in the first instance, it calls attention to only 187 items, to 333 in the second and 1031 in the third case. The proportion of true to false alarms differs considerably. An evaluation metric should, we think, report the fact. The situation concerning the AUC is quite simply this: with ever increasing total number of tokens, given the definition of FPR above, the FPR reduces to virtually zero and the TPR or recall becomes ever more predominant. Hence AUC's preference for correction systems that do well on recall. A spelling detection and correction system is sort of hybrid. Fundamental correction performance may not vary given more context (though in TISC's case we have shown that recall may improve given more context). Fundamental detection performance certainly changes. If one is only interested in correction performance one can rest content with measuring recall as we have done in Evaluation 1. In that case, one focuses on nothing but typos, makes sure no confusion in performance arises through the effect of the lexicon and decides which is the best corrector. However, as we have shown extensively, correction is only one part of the task. Detection is another and more context effectively changes the classifier's performance. In that this needs measuring, the AUC is not the best metric to decide on which corrector performs best.

Of course, this conclusion has heavy consequences for the work that is required to properly evaluate a spelling detection and correction system. If we had a metric that could tell us which corrector works best regardless of the error to token ratio, we would be in the more comfortable position of being able to use, say, typos within their single sentence context. While still presenting a sizeable amount of work to gather 2,000 typos within a single sentence context, we are here forced to conclude that this does not suffice. What seems to be required is typos in their natural distribution. This is an unfortunate situation, but we see no way around it.

Consider these two hypothetical correction systems: for a 10,000 token text containing 1% of typos, i.e. 100 typos, system A returns the full 10,000 item list with 100 best-first ranked corrections. System B returns only 100 items with 50 best-first ranked CCs. The scores are listed in Table 5.9. It can be seen that in terms of the AUC both systems

System	Dictionary	Recall	Precision	F-score	AUC
true scores: rank 1					
ISPELL (-c)	dutch	0.548	0.047	0.087	0.748
ISPELL (+c)	dutch	0.460	0.065	0.114	0.714
MPT	dutch (NL)	0.611	0.075	0.133	0.788
TISC-R	ILK-TWC/2	**0.675**	0.139	0.231	**0.827**
TISC-P	ILK-TWC/2	0.460	**0.563**	0.507	0.729
TISC-F	ILK-TWC/2	0.571	0.511	**0.539**	0.784

TABLE 5.8 Best-first ranking results for Dutch Metro2: system evaluated, dictionary used (for TISC: /2 indicates lexicon cut-off at frequency 2), recall, precision, F-score and AUC. In bold: highest score per metric.

System	Returned	Corrected	Rec.	Prec.	F-score	AUC
true scores: rank 1						
A	10,000	100	1	0.010	0.020	0.750
B	100	50	0.50	0.50	0.50	0.747

TABLE 5.9 Results for two hypothetical correctors on the basis of a fictitious 10,000 word token text containing 100 typos. Shown are items returned, items corrected, recall, precision, F-score and AUC.

are near equivalent, with a score which is halfway between random and perfect behaviour. The F-score for system B tells us that the job was half done. For system A the F-score clearly indicates that this is not so, with precision stating the obvious fact that the full list returned is a hundred times longer than the one returned by system B.

System B thus requires only one hundreth times the work to get half the job done right than system A requires to get the full job done. In the absence of fully automatic systems with great precision as well as great recall, we think system B, requiring us to examine a 100 item list to reduce error with 50% is the better option than system A requiring us to examine the full list to get the job done to perfection. Humans being error-prone, the latter is unlikely to come to pass, anyway. The former, TISC currently achieves. This is an order of magnitude better than what is achieved by the isolated-error correction systems, which in themselves reduce the total amount of work by an order of magnitude. We believe this information is better captured by the combination of recall and precision scores than by the AUC.

5.6 Evaluation versus real life

There is quite a large element of artificiality in evaluating a spelling error detection and correction system. We have undertaken the task of identifying thousands of typos in a list, thereby obtaining in effect two lists: one of these contains our positive class, i.e. the object of our studies: typos and the other our negative class: validated words. Yet, all we used both lists for was evaluation purposes: to show that the system we built performs adequately and demonstrably better than its predecessors.

In real life, we would use these lists to enhance our corpus-derived system: we would give TISC the list of typos to perform absolute correction where possible. We would also give TISC the list of validated words, to perform absolute detection of valid words and avoid perhaps the majority of wrong decisions it is bound to make due to data-sparseness. If we were to attempt fully automatic spelling detection and correction in real life, we would simply marshall all the resources at our disposal. We would use absolute correction, we would certainly employ CICCL, we would normalize the corpora used more thoroughly during preprocessing. We would not only use a disjoint out-of-domain corpus but at least incorporate what 'came before': we would most certainly train our system on the specific newspaper's output till now, to correct tomorrow's edition.

5.7 Summary

In this chapter we have first discussed the state-of-the-art as proposed in the literature and on the basis of our evaluation of the data used in those studies, have had to conclude that the approach by Brill and Moore (2000) differs from ours to such an extent that comparison is futile. While their system is geared to correcting the output of possibly very poor spellers, ours aims at reducing the residual noise after text has been edited and published. We have explored this niche for TISC by running a copyleft book, representing highly-edited text, through it. We found the system in actual fact identified 22 typos which had escaped the attention of author and editors, qualifying it for its role as a quick residual noise eliminator. We have then discussed what would constitute a representative and reliable test set for evaluating spelling error detection and correction systems. We have criticized a recent proposal for fully automatic evaluation of spelling checkers both on the basis of its underlying assumptions, of the kinds and distribution of the errors automatically introduced as well as the metrics adopted for evaluation. We have shown that confusion was created in the field of evaluation

by unfortunate use of terminology. We also examined at what level of performance a spelling error detection and correction system would be usable for fully automatic use. We have explained that van Rijsbergen precision shows exactly when a system is fit for fully automatic correction, i.e. when precision is higher than 0.5 more errors are corrected than correct words erroneously replaced. We have had to conclude that for Dutch, the work presented here presently only reaches break-even point.

6

Multilingual TISC

In this chapter we wish to explore whether TISC can be used unmodified to adequately perform spelling error detection and correction on mixed language text. We equip the system both with a bilingual, English-Dutch, and a trilingual, English-Dutch-French, lexicon. We then present it with a mixed language English-Dutch evaluation file and gauge its performance, all else being equal to the monolingual evaluations.

This work was inspired by the EAGLES-I (1996) final report on 'Evaluation of Natural Language Processing Systems', which lists as a 'dream tool'[1]:

> A multilingual spelling checker which automatically recognizes what language is being dealt with and switches to the appropriate spelling checker for that language.

We here explore the possibility of *not* performing explicit language detection. This was prompted by the observation that language detection in an isolated-word system may easily get confused. Take the Dutch newspaper Metro headline 'Crime passionel in Gronings zwembad' [Crime of passion in Groningen swimming-pool (21-10-2003)], which is a typical example of mixed language text, containing the typo *passionel, which in French should be spelled *passionnel*. The Microsoft Proofing Tools (MPT), for instance, can be set to automatically detect the language. Given the journalist is Dutch, the MPT would typically have Dutch as its default language and so does not switch languages given the headline's first word *crime* is a loan-word and present in the Dutch dictionary. It then encounters **passionel* and proposes the correct, Dutch, forms: *passionele* and its lemma *passioneel*. Whereupon the journalist, not being too versant in French, is likely to let his original pseudo-French **passionel* stand. The Dutch part of the web provides many more instances of this same error, as does the English,

[1] http://www.issco.unige.ch/projects/ewg96/node259.html

Anagram key	anagrams
75123219269	gerti, giert, griet, regit, riget, tiger, tigre
95176774701	ce tigre
95236791144	gerits, gierst, gister, greist, griest, griste, tigers, tigres
95666874202	de griet, de tigre, dreig te, giert de, tigre de
103824131401	agresti, gaiters, sigaret, staiger, tigresa, tirages
107081254058	dreigt u, du tigre, it urged, u dertig, u dreigt, urged it
115780446077	de gierst, de tigres, gerst die, get rides, griste de, its greed, tigres de
126783862653	gister te, greet its, it greets, tigres et, tire gets
127194825933	de rustig, drug ties, it surged, rustig de, surgit de, tigres du, urged its
129962785833	a stringed, and tigers, art design, dangers it, de ratings, drang iets, gradins et, grand site, granted is, gratin des, is granted, its danger, its garden, rating des, ratings de, red giants, sign trade, tigers and, tigre dans

TABLE 6.1 Extract from a trilingual (English, Dutch, French) TISC lexicon with the anagram keys and associated, chained anagrams.

for that matter. Our system being context-sensitive, we therefore explore whether its word bigrams alone aid the detection and correction of this kind of error, even when no further explicit language detection is done and no switching to another language dictionary occurs, its dictionary containing a mix of its various languages.

6.1 A multilingual lexicon makes a multilingual TISC

To make a multilingual version we concatenate the different languages' bigram lists. All other preprocessing is done as described for the monolingual version in Chapter 3, Subsection 3.3.1. We present an extract of the trilingual lexicon in Table 6.1.

6.2 Evaluation

6.2.1 Evaluation method rationale

To put TISC's performance on this task into perspective, we offset its results against the scores obtained by ISPELL when it is equipped with the same corpus-derived multilingual dictionaries. For both systems we used the lexicon obtained with bigram list cut-off at frequency 8.

6.2.2 TISC test settings

For testing the bi- and trilingual TISC versions, we limit ourselves to tests with the bigram list cut-off set at frequency 8. The aim of the test is primarily to show that bi- and trilingual spelling error detection and correction using the approach taken here is feasible, not to search for the best possible parameter settings or lexicon cut-off.

All test were run with the LD-limit at edit distance 3. We ran the tests with the following percentiles and mean settings: percentiles: 25, 50, 75; mean: mean value times: 0.25, 0.50, 1, 2, 3, 5.

6.2.3 Composition of the evaluation files

Dutch: For Dutch we used the Metro2 benchmark set as described in Subsection 4.3.1.

English: For English we used the Reuters RCV1 benchmark set as described in Subsection 4.2.1.

Dutch-English: For the bilingual tests, we concatenated the Metro2 and Reuters RCV1 benchmark sets and sorted the lines alphabetically, thereby obtaining a mixed language file.

6.2.4 Scoring and evaluation results

We measure performance on the full task of detecting and correcting errors in context for monolingual test sets, given both a bi- and trilingual lexicon, for both English and Dutch. The same is repeated for mixed language text, where the Metro Dutch benchmark set and the Reuters English benchmark set have been mixed. The results presented in Table 6.2 were obtained on the word tokens, for all systems. We now report not only scores in term of recall and precision, as combined in the F-score, but also the area-under-the-curve (AUC).

As can be seen from the table the AUC consistently favours the highest recall. For only one of the subtests do we find an AUC which is higher than the AUC associated to the highest recall obtained. The highest precision on the Dutch monolingual test remains below 0.5: remember we have not searched for the settings resulting in the best performance. Compare rather to the results obtained with the same settings on the fully monolingual task. While we may see a significant drop in F-score due to loss in precision, the AUC may nevertheless remain relatively unaffected.

It can also be seen that adding French to an already bilingual lexicon does not drastically change the percentile/mean values obtained. For both the monolingual and bilingual tests the almost consistently same mean/percentile settings produce the best results we have seen here in terms of recall, precision or F-score.

Settings	R	P	F	AUC
Dutch: Metro benchmark set				
ISPELL: BI	0.538	0.189	0.279	0.502
TISC-AUC: BI/Ax2	0.675	0.339	0.451	0.834
TISC-R: BI/Ax0.5	0.675	0.165	0.265	0.829
TISC-P: BI/25%	0.524	0.485	0.504	0.761
TISC-F: BI/50%	0.579	0.480	0.525	0.788
ISPELL: TRI	0.532	0.189	0.279	0.502
TISC-AUC: TRI/Ax2	0.659	0.309	0.420	0.826
TISC-R: TRI/Ax0.5	0.667	0.158	0.256	0.825
TISC-P: TRI/50%	0.579	0.406	0.477	0.788
TISC-F: TRI/50%	0.579	0.406	0.477	0.788
English: Reuters benchmark set				
ISPELL: BI	0.597	0.530	0.562	0.439
TISC-R: BI/Ax0.5	0.853	0.649	0.737	0.919
TISC-P: BI/25%	0.769	0.875	0.818	0.883
TISC-F: BI/50%	0.789	0.865	0.825	0.893
ISPELL: TRI	0.586	0.522	0.552	0.426
TISC-R: TRI/Ax0.5	0.852	0.654	0.740	0.919
TISC-P: TRI/25%	0.763	0.875	0.815	0.880
TISC-F: TRI/50%	0.784	0.868	0.824	0.890
Dutch-English: Mixed Metro-Reuters benchmarks				
ISPELL: BI	0.595	0.497	0.541	0.457
TISC-R: BI/Ax0.5	0.843	0.583	0.689	0.914
TISC-P: BI/25%	0.756	0.845	0.798	0.876
TISC-F: BI/50%	0.778	0.835	0.805	0.887
ISPELL: TRI	0.583	0.489	0.532	0.446
TISC-R: TRI/Ax0.5	0.842	0.589	0.693	0.914
TISC-P: TRI/25%	0.752	0.846	0.796	0.874
TISC-F: TRI/50%	0.775	0.836	0.804	0.885
D-E Upper bounds: averaged monolingual results				
ISPELL	0.795	0.029	0.056	0.696
TISC-R	0.848	0.600	0.703	0.917
TISC-P	0.765	0.856	0.808	0.881
TISC-F	0.790	0.842	0.815	0.893

TABLE 6.2 Statistics of best test scores on best-first ranking. Shown are the test settings in terms of bilingual or trilingual lexicon used, and the percentile or mean value employed. Given are the scores on recall R, precision P, F-score F and area-under-the-curve AUC.

6.2.5 Discussion

Bilingual task: Table 6.2 presents the best results on the bilingual English-Dutch correction task obtained by TISC and ISPELL with dictionaries based on the same bilingual (D-E) (BI) and trilingual (D-E-F) (TRI) bigram lists. These results are contrasted to upper bounds represented by the average of the monolingual results on the two evaluation sets obtained by ISPELL. The result shown is the average of ISPELL's results for Dutch obtained without using the -C switch and the English result obtained with the largest native ISPELL dictionary.

A rather striking result is that ISPELL's performance is drastically improved by providing it with a much larger dictionary. The presence of names alone in the dictionary provided by us must account for the better part of the gain in precision. What we see is an order of magnitude improvement in F-score, even though we lost 20% recall due to the noise in the lexicon and ISPELL's lack of features to handle such noise. TISC's in-built features for handling such noise, i.e. reliance on the Zipf filter thresholds instead of simple dictionary verification, allows it to largely maintain the levels of performance reached on the monolingual tasks as represented by the upper bounds, even with a bi- or trilingual lexicon.

What we primarily learn from this exercise is that TISC is robust in light of mixed languages. We have shown that mixing two or three languages hardly affects correction performance. This is a valuable finding.

Simply mixing three languages seems to have no adverse effect on TISC's capability of performing correction to these levels of performance. Nevertheless, the fact remains that this strategy entails that one particular type of errors will go undetected, namely those errors in a specific language that result in a valid word in one of the other languages in this type of multilingual system. These would have to be called *bilingual or translingual confusables*. Our evaluation files happen to contain a few of them, e.g. *polite* which should have read *politie* [police] in the Dutch evaluation set. The fact that these are a lot rarer than errors which do not form a valid word in any of the languages, obscures their effect. Note that these would throw a non-context-aware system which does attempt to do language detection off balance. We think context-awareness here too should help remedy this shortcoming of our non-language-detecting approach. Provided the error detection module is made to take into account the word bigram information in much the same way as the error correction module currently does, it should also be possible to detect these anomalies. This may be a nice pointer to the way we should direct our future work, in that this at least

hints at ways the harder task of detecting and remedying monolingual confusables may be tackled.

Another corollary of these results settles a question which has engaged researchers, off and on, since Peterson (1986). The discussion is about the size of the dictionary and its effects on the performance of spelling checkers. We have here mixed three languages, all three of which have huge numbers of cognate words, through historical circumstances. This offers even more opportunity than in the monolingual situation for a misspelling to result into an existing word, be it a word from another language. We hardly observe an effect on the performance of TISC. We conclude that given our approach the issue is settled in favour of Damerau and Mays (1989), who challenged Peterson's conclusions that spelling correction dictionaries should be kept small. Given the improvement in precision we have shown is possible by having a larger dictionary, even for ISPELL, we think dictionaries should be made far more comprehensive. Context-sensitivity takes care of the rest.

6.3 Summary

In this chapter we have established that multilingual spelling correction without prior language detection is feasible. We have shown that all it takes for TISC to be turned into a well performing multilingual spelling error detection and correction system, is to equip it with a multilingual lexicon. Performance levels remain stable, whether the lexicon is bi- or trilingual, showing that context-sensitivity alone is sufficient to keep the languages separate. We have also established that an isolated-word corrector such as ISPELL can benefit enormously by being equipped with a much larger dictionary than it is normally equipped with. The gain is in precision, which we have shown throughout Chapters 4 to 6 to be a highly important aspect of any spelling error detection and correction system's performance.

7

Conclusion

To conclude this dissertation we give an overview of what each chapter has allowed us to summarize.

7.1 Summation

In Chapter 1 we have provided an introduction into the field of spelling error detection and correction. We have provided the necessary terminology, after which we have discussed what distinguishes an erroneous word variant from a conventionally accepted form. We have shown that the work by George Kingsley Zipf provides insight into what constitutes a language's vocabulary and into the distribution of words within the vocabulary. We have from these insights derived implications for the design of a spelling error detection and correction system. Next we have given an overview of approximate string matching techniques and how they relate to spelling error detection and correction. We have briefly positioned our own core correction mechanism based on a non-phonetic similarity key within this field. We have then surveyed some of the history of spelling error detection and correction research and concluded that whereas research into real-word error detection on the basis of context has drawn a lot of attention, research into non-word spelling correction has largely stuck on the isolated-word level. We have then discussed the noisy channel modelling approach to spelling error correction and pointed out that this is necessarily a language-specific technique. We have discussed the relationship between ranking the correction candidates and taking account of the typo's context. We have then introduced the name for the system we propose: Text-Induced Spelling Correction. We have finally outlined the main contributions of the present work and provided an overview of its chapters.

In Chapter 2 we first introduced the corpora we used throughout this work. We then described how we collected and studied a large sample

of typos in English which should be representative of what one may encounter in a contemporary corpus. We have concluded that our findings do not bear out Kukich's qualification that given the availability of spelling checkers fewer non-word errors occur today than there used to before the advent of spelling checkers. Put quite simply: the non-word error problem has not gone away and does not even show signs of being on the retreat. The extent to which a contemporary corpus has been shown to contain non-word errors is to all intents and purposes the same as reported by Kukich (1992). Our findings are also in line with Pollock and Zamora (1983) (p. 53), who report an overall incidence of 0.20% of misspellings in the databases they studied and state that this is 'probably what one should expect in raw keyboarding by experienced operators'. We found an incidence of 1 in 400 tokens in the Reuters RCV1 corpus, or 0.25%. We have take this figure to constitute the natural distribution of typos in keyboarded text throughout the rest of this work.

We have not tried to quantify the size of the real-word problem: real words cannot be detected by studying a frequency list. Neither have we tried to quantify the proportion of cognitive errors versus typographical errors. It is simply too hard to see what caused the error from the output. What we did quantify is that only very few typos have a Levenshtein distance or LD larger than three. We observed only one single case where the LD was 5 in 12,072 typos. This was *seeked for *sought*, which is a grammatical error, though nevertheless a non-word.

We showed that discarding the hapax legomena and dis legomena from a corpus only allows for removing about 66% of the variation, at a cost of losing about 33% of the real-word types. We detailed what the impact of this is on the frequency mass measured for one particular word, *government*. Arguably, for this particular word which most likely has a very even distribution over the corpus, the loss in frequency mass due to variations is negligible and will not unduly affect probability estimates. We counter-argued that the situation is likely to be more dramatic for words more bursty in nature. Given an automatic spelling correction system that achieves not only good recall, i.e. is capable of correcting the typos it finds, but also high precision in doing so, i.e. does not report real words to be non-words and replaces them by other real-words, up to 77,6% of the variants present in a corpus might be removed by correcting only those typos that are within LD 1. By correcting only the typos that are within LD 1 + 2, which given the computational resources available to date should be well within reach, up to 98.7% of the variation within a corpus might be removed. We think pursuing this goal has far better chances of alleviating the data

sparseness problem and of improving statistical language models than the common practice of hapaxing has.

In Chapter 3, we outlined our spelling detection and correction mechanisms. We introduced the notion of anagram-key based hashing, which allows for quick retrieval of correction candidates within the limits specified by both the alphabet and the LD limit imposed. We demonstrated how having word unigrams as well as word bigrams in the lexicon allows for the correction process to be applied not only to isolated words but also to words within their context. We showed how this facilitates best-first ranking. We illustrated how we handle compounds and when spelling correction on the tier of the compounding parts is effected. The resources used by TISC are derived from corpora in a completely unsupervised manner. This necessitates more informed typo detection strategies than simple dictionary look-up as performed by most spelling error detection systems. So we introduced the notion of Zipf Filters, thresholds set manually or derived from the corpus-derived lexicon and we described how these help us to distinguish between correct and erroneous word forms. We extensively used examples to illustrate how the various modules act and interact. We finally showed how the core correction mechanism can be applied to the TISC lexicons themselves to filter highly recurrent typos from them. The errors acquired in this way have further formed the basis for context-sensitive absolute spelling correction.

On the basis of extensive evaluations in Chapter 4 we have established that:

- the correction mechanism we propose can resolve virtually any type of error encountered in a real-world corpus. The few types of errors it cannot resolve typically involve higher LDs and multiple errors which are very rare in keyboard-input text.
- using a smaller alphabet, i.e. an alphabet not including character trigram values, results only in a minimal loss of performance.
- performing correction not only on the isolated word tier, but also looking at the immediate context has only a slight effect on overall recall but improves the best-first ranking of the CCs.
- the corpus derived lexicons can to a certain extent be cleaned in an unsupervised way.
- for the full task of detecting and correcting typos within running text we have seen that precision keeps rising the more information is provided to the system, i.e. the lower the frequency cut-offs used.
- with manually set Zipf Filter thresholds this results in loss of recall through crowding by incorrect variants included in the lexicon. This

calls for more rigorous corpus normalisation during pre-processing and for applying absolute correction.

- using corpus-derived thresholds we can manipulate the levels of precision and recall and can force the system to focus on the one. This is to the detriment of the other, but a good balance can be achieved.
- given a realistic amount of context, i.e. typos within their full newspaper article, we reach a level where for every error removed, only one correct word would be replaced, if the system were run in a fully unsupervised, automatic fashion. This we have shown to be the case for Dutch, which we have also shown to pose greater challenges to spelling error detection and correction systems than English.
- for both languages English and Dutch, we have shown that TISC outperforms the state-of-the-art systems available today. For English we have shown that this is the case for ASPELL, ISPELL and MPT. For Dutch: ISPELL and MPT.

On the basis of these findings we conclude that Text-Induced Spelling Correction is a viable alternative to the approaches to spelling error detection and correction as embedded in the state-of-the-art systems available today.

In Chapter 5 we first discussed the state-of-the-art as proposed in the literature and on the basis of our evaluation of the data used in those studies, have had to conclude that the approach by Brill and Moore (2000) differs from ours to such an extent that comparison is futile. While their system is geared to correcting the output of possibly very poor spellers, ours aims at reducing the residual noise after text has been edited and published. We then discussed what would constitute a representative and reliable test set for evaluating spelling error detection and correction systems. We have criticized a recent proposal for fully automatic evaluation of spelling checkers both on the basis of its underlying assumptions, of the kinds and distribution of the errors automatically introduced, as well as the metrics adopted for evaluation. We have shown that confusion was created in the field of evaluation by unfortunate use of terminology. Next we examined what the F-score can tell us about whether a spelling error detection and correction system would be usable for fully automatic use. We have shown that van Rijsbergen precision shows exactly when a system is fit for fully automatic correction, i.e. when precision is higher than 0.5 more errors are corrected than correct words erroneously replaced. We have had to conclude that for Dutch, the work presented here presently only reaches break-even point. We have explained that were we to really attempt fully automatic spelling detection and correction, we would marshall

all the resources at our disposal. We would use absolute correction, we would most certainly employ Corpus-Induced Corpus Clean-up or CICCL, we would normalize the corpora used more thoroughly during preprocessing. We would not only use a disjoint out-of-domain corpus but at least incorporate what 'came before': we would most certainly train our system on the specific newspaper's output till now, to correct tomorrow's edition.

We have pointed out that there is quite a large element of artificiality in evaluating a spelling error detection and correction system. We have identified thousands of typos in a list, thereby obtaining in effect two lists: one of which contains typos and the other validated words. Yet, all we used both lists for was evaluation purposes: to show that the system we built performs adequately and demonstrably better than its predecessors. In real life, what we would use these lists for would in fact be to enhance our corpus-derived system: we would give TISC the list of typos to perform absolute correction where possible. We would also give TISC the list of validated words, to perform absolute detection of valid words and avoid perhaps the majority of wrong decisions it makes due to data-sparseness.

What we also set out to do was build a system that would enable one to take a text that manually requires perhaps days to proofread, run it through and in the time of a short break returns a manageably short list containing on the one hand words for which it did not have sufficient information about to validate and on the other hand words forms that truly do not belong in a properly spelled text: typos. This we actually did for a copyleft English book, which represents highly edited text. Of the 400 words returned by TISC, 22 proved to be actual typos which had been missed in editing steps. Compared to what was available before, this represents a major step forward.

In Chapter 6 we established that multilingual spelling correction without prior language detection is feasible. We have shown that all it takes for TISC to be turned into a well-performing multilingual spelling error detection and correction system, is to equip it with a multilingual lexicon. Performance levels remain stable, whether the lexicon is bi- or trilingual, showing that context-sensitivity alone is sufficient to keep the languages separate. We also established that an isolated-word corrector such as ISPELL can benefit enormously from being equipped with a much larger dictionary than it is normally equipped with. The gain is in precision, which we have shown throughout Chapters 4 to 6 to be a highly important aspect of any spelling error detection and correction system's performance.

7.2 Final note: roll your own TISC!

As a final note, we want to draw due attention to the fact that we have developed a competitive spelling checking and correction system using nothing besides our novel algorithm and electronically available collections of text. For languages such as Dutch and English, of course, a great deal of natural language processing resources are available. We have deliberately ignored these as we wanted to demonstrate the feasibility of achieving what we have achieved without having recourse to these resources. There are a great many languages in this world for which little or no such resources have as yet been developed. The inexpensive and language-independent approach outlined here, we hope, may help to remedy the lack of spelling aids for underresourced languages. It will be clear from the above that in order to develop a competitive spelling error detection and correction system for another language than the ones we have been working with, no higher mathematics is required. A corpus, a working knowledge of the language and some programming skills are all that is required. We have made it appear that the corpus needs to be enormous. That is not the case. In actual fact, the smaller the corpus, the more feasible it is to apply Corpus-Induced Corpus Clean-up or CICCL, which is after all not much more than the core correction mechanism, and to verify its output manually. While we have worked without a trusted dictionary, this is no prerequisite, either. Given a trusted dictionary TISC may just as well be made into a normative spelling error detection and correction system. We have tried to do as much as possible without language-specific rules. That, too, is not at all a strict requirement. One may very well chose to model the specifics of one's language. The results will probably be only the better for it. All we have shown, is that this is not strictly necessary. And modelling specifics largely precludes multilingual spelling correction. We still find all this amazing. And great fun, too ... Enjoy!

A

Obtaining statistics from error lists

A.1 Obtaining the error type statistics

In Chapter 2 we presented statistics on the correction/typos pairs in the list culled from the RCV1. We here explicitly detail how the counts were obtained.

Given a correct word/typo pair we describe from the point of view of the correct word what changes occurred within the correct word to produce the typo.

Not to get into too fine detail about the actual identity of the characters involved, we limit this to specifying that characters either match, were deleted, inserted, substituted, transposed or combinations thereof. This is done by replacing all matching characters by *M*, marking a deletion with *S*, an insertion by *I* and a substitution by *S*. Transpositions of two adjacent characters were marked with a single *T*. The pair *actress*/*acress is then represented by the string: MMDMMMM.

The GNU open scource provides a Perl package called Text::Brew which is an implementation of what is there called the Brew edit distance [1]. As this package is available to all[2], we use it as the basis for adding a wrapper program to it, which effects some desirable rewritings and collects the actual error type statistics.

Necessary output rewriting Output of the module when given the correct word/typo pair *actress*/*acress (naturally without the asterisk) is: 'The Brew distance for (actress,acress) is 1, obtained with the edits: INITIAL, MATCH, MATCH, DELETION, MATCH, MATCH, MATCH, MATCH'. The distance one is in effect the Levenshtein Distance (LD).

This output string we capture and reduce to a more basic format:

[1] This is defined in http://ling.ohio-state.edu/∼cbrew/795M/string-distance.html

[2] http://search.cpan.org/∼kcivey/Text-Brew-0.02/lib/Text/Brew.pm

actress#acress#1#MMDMMMM. This means we discard the text and the INITIAL string, abbreviate the descriptions and only retain the essentials separated by a unique marker, here '#'. Now the package only returns MATCH, DELETION, INSERTION or SUBSTITUTION. Transpositions it is unaware of, as are many LD-implementations. However, we can easily capture transpositions: given the input strings and employing our anagram hashing algorithm, it is straightforward to find out which pairs only involve character transpositions. We calculate the anagram values for both strings and subtract the one from the other. Given the result is zero, we know the only transformation between both strings are displaced characters. For a transposition of adjacent characters, the rewritten string will contain two adjacent *S* symbols. This we now rewrite as a single *T*. We do not reduce the editing cost of 2 to 1, but this might be done, if all editing operations are to be assigned the same cost. Transposition of non-adjacent characters we leave as is. These will then be interpreted as multi-point multiple errors.

Initial character transformations are also differently parsed by the module than might be expected. These too we post-edit into what we think is a more intuitively correct interpretation. Given the pair *expected*/*pected the rewritten pattern reads SDDMMMMM, with an edit cost of 3. So the transformation is interpreted as a single substition, followed by two deletions. Actually, all that happened was that the first two characters were deleted. So we rewrite the initial *SDD* as *DD* and change the editing cost to 2. Single first character deletions are likewise rewritten from initial *SD* to *D* and the cost lowered to 1. It is likewise for first character insertions: initial *SII* to *II* and cost from 3 to 2, initial *SI* to *I* and cost from 2 to 1. First character substitutions are interpreted correctly and require no rewriting. First character transpositions were dealt with by the prior *SS* to *T* rewriting.

Collecting the statistics We can now start collecting the statistics. This we want to do per LD involved. We obtained the distance from the Brew edit distance module and tally for each category per LD how many cases we observed by iterating over the list of rewritten patterns. We start off with first character substitutions, which are recognizable as they have one or more initial *S*, only followed by one or more *Ms*. If we want to distinguish between first character substitutions that do or do not involve capitalization, we check whether for those cases where a first character substitution was observed, the completely lowercased input strings give an exact match or not. If they do, all that was substituted was a lower cased character for its uppercase version and we tally this category. Else we tally the category first character substitutions not

involving capitalization.

We next strip off initial and final *Ms*, reducing e.g. MMMIMMDMMMM to IMMD and MMMMMDDDMMMMM to DDD. If the resulting pattern still has matching *M(s)*, we tally the count for multi-point multiple typos, without further analysis of the actual multiple edits. Those that no longer have any matching *M(s)*, are all the single-point errors, whether or not they are multiple. We will distinguish between sequences of *Ds* only, *Is* only, *Ss* only, *Ts* only or combinations of two or more of these single error types. For the single error types, we again, per LD, tally the counts. The combinations are tallied together, per LD, as single-point multiple errors.

All that is further required is the necessary accounting and tabulating of percentages over the rows and the columns, making sure not to double count the first character statistics.

Summary

Text-Induced Spelling Correction

The main contribution of this dissertation is a novel approximate string matching algorithm for indexed text search. The algorithm is based on a hashing function which uniquely identifies strings composed of the same subsets of characters, i.e. anagrams, by means of a numeric value. The numeric value allows for searching for character strings differing from a particular string by a predefined number of characters. As such it forms an ideal basis for a novel spelling error detection and correction algorithm, which we call Text-Induced Spelling Correction. Our system uses nothing but lexical and word cooccurrence information derived from a corpus, a very large collection of texts in a particular language, to perform context-sensitive spelling error correction of non-words. Non-words are word strings produced unintentionally by a typist that deviate from a convention about how words are to be spelled in order to be considered real-words within the language.

The spelling error detection and correction mechanism we propose uses not only isolated word information, but also context information. It performs context-sensitive error correction by deriving useful knowledge from the text to be spelling checked. In Dutch, for instance, productive compounding precludes that a spelling checker's dictionary can ever be complete. However, often a wrongly spelled compound is either present in its correct form within the text, or its component parts are present within the text. By means of this knowledge our system is capable of correcting typos for which it does not have the correct word in its dictionary. Apart from this, some typos are ambiguous in that they may resolve into two or more different words. We investigate in depth the relationship between a typo and its context and propose a new algorithm for ranking correction candidates that specifically makes use of the typo's context to propose first *this* candidate rather than *that*

candidate. In the laboratory sentence: 'Her vehement *onjections to these painful *onjections were based on solid medical evidence, as well as a hearty dislike of needles.' the first occurrence of the typo should be resolved into *objections* the second into *injections*. Our system uses a context window and the corpus-derived word bigrams in its dictionary to perform that resolution.

Overview of the dissertation

In Chapter 1, we state our goals, briefly survey the history of spelling error detection and correction work and the state-of-the-art. We position our own approach in light of these and describe our contribution: **We show that it is possible to use nothing besides a large corpus of raw text in a particular language and nothing but unsupervised techniques based on the similarity key based correction algorithm we propose to derive a competive context-sensitive non-word spelling error detection and correction system for that particular language from the corpus.**

In Chapter 2, we discuss the corpora we used for this study and perform an in-depth analysis of the Reuters RCV1, a large corpus of contemporary English news stories. We found that over 20% of the non-capitalized word types in the corpus are erroneous word forms. This means that 1 in every 400 word tokens in the text is a non-word. On the basis of this we conclude that the non-word error problem has not yet been solved.

In Chapter 3 we provide a full description of the spelling detection and correction system we developed and which we have called 'Text-Induced Spelling Correction' or TISC.

In Chapter 4 we conduct in-depth evaluations of TISC for both English and Dutch and present the results on a variety of tasks. We compare the results with those obtained by three state-of-the-art systems available today. This allows us to conclude that TISC is as capable of correcting typos as the three other systems, but far more precise in doing so: it raises only one-tenth the false alarms these systems do.

In Chapter 5 we evaluate the evaluations and conduct a survey of the various metrics one may use for evaluating a spelling detection and/or correction system. We discuss why the F-measure is better suited for evaluating spelling error detection and correction systems than the Area-Under-the-Curve of AUC because it measures more adequately the effect of the ratio of non-words versus real-words on the precision of the system. We compare with state-of-the-art-systems proposed in the literature and show how the approaches differ. While other systems

typically try to cater for the orthographically challenged, i.e. very bad spellers, our system is aimed at removing residual errors overlooked in the editing process of longer texts. We illustrate this by presenting nearly 20 typos detected by TISC in an online available published book of about 120,000 word tokens. We propose a system for the description of evaluation sets which should allow for better comparison of systems when the evaluation data are not publicly available.

Chapter 6 is devoted to multilingual spelling error detection and correction and the issues involved: we examine whether explicit language detection is a necessary prerequisite. We present results obtained on the spelling correction of English-Dutch mixed language text using a trilingual, English-Dutch-French, dictionary and show that this can adequately be done without explicit prior language detection.

In Chapter 7 we present our conclusions and discuss future work.

Samenvatting

TISC: Tekstgeïnduceerde Spellingscorrectie

De belangrijkste bijdrage van dit proefschrift is een nieuw algoritme voor benaderende stringmatching voor geïndexeerd zoeken in tekst. Bij exacte stringmatching wordt gezocht naar karakterreeksen die precies overeenstemmen met een bepaalde string. Bij benaderende stringmatching naar karakterreeksen die goed gelijken op, maar in bepaalde opzichten verschillen van, een bepaalde string. Het algoritme is gebaseerd op een hashing functie die het mogelijk maakt door middel van een numerieke waarde die strings te identificeren die bestaan uit precies dezelfde set karakters, met andere woorden: anagrammen. De numerieke waarde is een soort index die toegekend wordt aan strings bestaande uit dezelfde set karakters, ongeacht de volgorde van de karakters in de string. Deze index maakt het verder mogelijk die strings op te zoeken die met een vooraf vastgelegd aantal karakters verschillen van een gegeven string. Bijgevolg vormt deze hashing functie een ideale basis voor een nieuw spellingsfoutdetectie en -correctie algoritme, dat we Text-Induced Spelling Correction of Tekstgeïnduceerde Spellingscorrectie noemen. Het systeem gebruikt niets anders dan lexicale informatie en frequentie-informatie over het samen voorkomen van woorden in een groot corpus, een grote verzameling van teksten in elektronische vorm in een bepaalde taal, om contextgevoelige spellingcorrectie van niet-woorden te verrichten. Niet-woorden zijn woordstrings die onopzettelijk geproduceerd worden door een typist en die afwijken van de norm die bepaalt hoe woorden gespeld moeten worden om als aanvaardbare woorden binnen die taal beschouwd te kunnen worden.

Het door ons voorgestelde spellingsfoutdetectie en -correctiesysteem gebruikt niet enkel informatie over de losse woorden, maar ook contextinformatie. Het verricht contextgevoelige correctie door bruikbare kennis af te leiden uit de tekst waarvan de spelling gecontroleerd moet

worden. In het Nederlands, bijvoorbeeld, voorkomt de productiviteit van het samenstellen en aan elkaar schrijven van samengestelde woorden dat het woordenboek waarover een spellingscorrector beschikt ooit volledig kan zijn. Daar staat tegenover dat een verkeerd gespelde samenstelling vaak ofwel ook in zijn correcte vorm aanwezig is in de tekst ofwel dat de samenstellende delen los voorkomen in de aanvaarde spellingsvariant. Door gebruik te maken van deze informatie is ons systeem in staat fouten te corrigeren in woorden waarvoor de correcte vorm niet in het woordenboek voorkomt. Daarnaast is het zo dat bepaalde incorrecte woordvormen ambigu zijn in de zin dat ze gemakkelijk kunnen omgevormd worden tot meerdere aanvaarde woorden. We onderzoeken grondig de relatie tussen een incorrecte woordvorm en zijn context en stellen een nieuw algoritme voor voor het rangschikken van de correctiecandidaten. Dit gebruikt de context van de spelfout, met name de woorden die het fout gespelde woord omringen, om eerder *deze* candidaat dan *die* candidaat te suggereren. In de gefingeerde zin: 'Haar *onjectie tegen die pijnlijke *onjectie steunt op grondige medische redenen, maar ook op een gezonde afkeer van naalden' moet het eerste voorkomen van de spelfout verbeterd worden in *objectie*, de tweede in *injectie*. Ons systeem gebruikt een contextvenster en de uit het corpus geëxtraheerde woordbigrammen, die in het woordenboek opgenomen zijn, om dat op te lossen.

Overzicht van dit proefschrift

In Hoofdstuk 1 beschrijven we onze doelstellingen, geven we een kort overzicht van het voorafgaande onderzoek op het gebied van spellingsfoutdetectie- en correctie en stellen we vast wat de stand van zaken heden ten dage is. We bepalen hoe ons systeem zich hiertegenover geplaatst ziet en omschrijven onze bijdrage: **We tonen aan dat het mogelijk is een competitief context-gevoelig niet-woord spellingsfoutdetectie en -correctiesysteem voor een bepaalde taal af te leiden uit niets anders dan een grote verzameling niet-geannoteerde, ruwe tekst in die bepaalde taal door gebruik te maken van niet-gesuperviseerde technieken die gebaseerd zijn op het door ons voorgestelde en op een gelijkheidssleutel gebaseerde correctie-algoritme.**

In Hoofdstuk 2 bespreken we de corpora die we in deze studie gebruiken en voeren we een grondige analyse uit van het Reuters RCV1 corpus, een hedendaagse, grote verzameling van in het Engels geschreven nieuwsberichten. We vonden dat meer dan 20% van de woordtypes die niet begonnen met een hoofdletter, in feite fout gespelde

woorden zijn. Dit betekent dat per vierhonderd woorden in de lopende tekst er 1 niet-woord voorkomt. Op basis hiervan concluderen we dat het niet-woord probleem nog niet opgelost is.

In Hoofdstuk 3 beschrijven we het door ons ontworpen spelfoutdetectie en -correctiesysteem dat de Engelse naam 'TextInduced Spelling Correction' of kort: TISC meekreeg.

In Hoofdstuk 4 evalueren we TISC grondig voor het Engels en het Nederlands en presenteren we de resultaten die het systeem behaalt op een reeks verschillende taken. We vergelijken de behaalde resultaten met de resultaten behaald door drie state-of-the-art systemen die momenteel beschikbaar zijn. Dit laat ons toe te besluiten dat TISC op zijn minst zo goed in staat is als de andere systemen om spelfouten te corrigeren, maar daarbij veel preciezer te werk gaat: bij TISC komt 'vals alarm' tien maal minder voor dan bij de andere systemen.

In Hoofdstuk 5 evalueren we de evaluaties. We overschouwen de verschillende metrieken die gebruikt kunnen worden om spelfoutdetectie en -correctiesystemen te evalueren. We bespreken waarom we de F-maat metriek hiertoe beter geschikt vinden dan de Area-Under-the-Curve (het oppervlak onder de curve) of AUC-metriek. De F-maat meet beter het effect van de verhouding niet-woorden tot aanvaarde woorden op de precisie van een systeem. We vergelijken onze aanpak met state-of-the-art systemen die in de literatuur voorgesteld zijn en tonen op welke wijze de aanpak verschilt. Andere systemen richten zich specifiek op de problemen van orthografisch geprovoceerde mensen, met name heel slechte spellers. Ons systeem is er eerder op gericht de enkele aan het editoriaal proces van langere teksten ontsnapte spelfouten aan het licht te brengen en te corrigeren. Dit illustreren we door te tonen dat TISC er in slaagt een twintigtal spelfouten te detecteren in een op het internet beschikbaar gepubliceerd Engelstalig boek van ongeveer 120.000 woorden. We stellen ook een systeem voor om evaluatiesets zo te beschrijven dat een betere vergelijking tussen systemen mogelijk wordt in het geval de evaluatiesets zelf niet publiek beschikbaar zijn.

In Hoofdstuk 6 behandelen we multilinguale spellingsfoutdetectie en -correctie en wat daarbij komt kijken: we bekijken of expliciete voorafgaande taaldetectie daarbij een noodzakelijke vereiste is. We presenteren onder meer de resultaten die TISC behaalt bij de spellingcorrectie van een gemengde Engels-Nederlandse tekst met een drietalig, Engels-Nederlands-Frans, lexicon. We tonen aan dat deze taak adequaat vervuld kan worden zonder expliciete voorafgaande taaldetectie.

In Hoofdstuk 7 besluiten we met de conclusies die bereikt werden in dit werk en bespreken we het werk dat ons wacht in de toekomst.

List of abbreviations

10x2K	Ten sets of 2,000 typos randomly chosen, with overlap, from **12K**.
12K	The list of 12,038 typos randomly chosen from the 33,488 typos culled from RCV1 and provided manually with the correction dictated by the corpus context.
AAV	Anagram key Values in the Alphabet
AUC	Area-Under-the-Curve metric
AV	Anagram key Value
BI	Bigram
BNC	British National Corpus
CC	Correction Candidate
CICCL	Corpus-Induced Corpus Clean-Up
COOC	Cooccurrence Count
F	F-score
FN	False Negatives
FPR	False Positive Rate
FP	False Positives
FR	Frequency of Retrieval (of a particular CC, by TISC.)
FR1	F-score on best-first ranking (rank 1)
FRQ	Frequency
IAV	Input word Anagram Value
IDF	Inverse Document Frequency
ILK	Induction of Linguistic Knowledge (Research Group at Tilburg University)
LD	Levenshtein Distance
LNRE	Large Numbers of Rare Events
LPB	Left Part of a Bigram

LPC	Left Part of a Compound
MPT	Microsoft Proofing Tools
NYT	The New York Times
OCR	Optical Character Reading
OR	Overall Recall
POS	Part-Of-Speech
P	Precision
PR1	Precision on best-first ranking (rank 1)
R	Recall
RCV1	Reuters Corpus Volume I
ROC	Receiver Operating Characteristic curve
ROUL	Roularta Magazines Corpus
RPB	Right Part of a Bigram
RPC	Right Part of a Compound
RR1	Recall on best-first ranking (measuring recall only for rank 1)
RR5	Recall on five-first ranking (measuring recall on ranks 1 to 5)
SUC	Stockholm-Umeå corpus
TAV	input type or Typo derived Anagram key Values
TISC	Text-Induced Spelling Correction
TISC-AUC	TISC version obtaining the highest AUC-score
TISC-F	TISC version obtaining the highest F-score
TISC-P	TISC optimized on precision
TISC-R	TISC optimized on recall
TN	True Negatives
TP	True Positives
TPR	True Positive Rate
TR	True Recall
TRI	Trilingual
TWC	Twente Corpus
TWC2	Year 2002 Upgrade to the Twente Corpus
UBR	Upper Bound Recall
UK	United Kingdom
UR	Upgraded Ranking
URF	Upgraded Retrieval Frequencies
URL	Universal Resource Locator
US	United States
WWW	World-Wide Web
XML	Extensible Markup Language

References

Agirre, Eneko, Koldo Gojenola, Kepa Sarasola, and Atro Voutilainen. 1998. Towards a single proposal in spelling correction. In *Proceedings of the 17th international conference on Computational linguistics*, pages 22–28. Association for Computational Linguistics.

Baayen, R. Harald. 1996. The effects of lexical specialization on the growth curve of the vocabulary. *Computational Linguistics* 22(4):455–480.

Baayen, R. Harald. 2001. *Word Frequency Distributions*. Dordrecht: Kluwer Academic Publishers. Text, Speech and Language Technology, vol. 18, series editors: Nancy Ide and Jean Véronis.

Baxter, R., P. Christen, and T. Churches. 2003. A comparison of fast blocking methods for record linkage. In *Proceedings of ACM SIGKDD'03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 25–27.

Bellman, Richard. 1957. *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bigert, Johnny. 2005. *Automatic and unsupervised methods in natural language processing*. Ph.D. thesis, KTH Numerical Analysis and Computer Science, Stockholm, Sweden. Trita-NA, ISSN 0348-2952.

Bigert, Johnny, L. Ericson, and A. Solis. 2003. Missplel and AutoEval: Two generic tools for automatic evaluation. In *Proceedings of the Nordic Conference in Computational Linguistics 2003*. Reykjavik, Iceland.

Bradley, Andrew P. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition* 30(7):1145–1159.

Brill, Eric and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the ACL*, pages 286–293.

Carlson, Andrew J., Jeffrey Rosen, and Dan Roth. 2001. Scaling up context-sensitive text correction. In *IAAI*, pages 45–50. American Association for Artificial Intelligence.

Chelba, Ciprian and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In D. Lin and D. Wu, eds., *Proceedings of EMNLP 2004*, pages 285–292. Barcelona, Spain: Association for Computational Linguistics.

Clarkson, P.R. and R. Rosenfeld. 1997. Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings ESCA Eurospeech 1997*.

Comeau, Donald C. and W. John Wilbur. 2004. Non-word identification or spell checking without a dictionary. *Journal of the American Society for Information Science and Technology* 55(2):169–177.

Crystal, David. 1985. *A dictionary of Linguistics and Phonetics*. Oxford: Basil Blackwell.

Cucerzan, Silviu and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In D. Lin and D. Wu, eds., *Proceedings of EMNLP 2004*, pages 293–300. Barcelona, Spain: Association for Computational Linguistics.

Curran, James R. and Miles Osborne. 2002. A very very large corpus doesn't always yield reliable estimates. In *Proceedings of CoNLL-2002*, pages 126–131. Taipei, Taiwan.

Daelemans, Walter. 1987. *Studies in Language Technology. An Object-Oriented computer model of Morphological aspects of Dutch*. Ph.D. thesis, Katholieke Universiteit Leuven, Departement Linguïstiek, Louvain, Belgium.

Daelemans, Walter, Antal van den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning* 34:11–41.

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

Damerau, Fred J. and Eric Mays. 1989. An examination of undetected typing errors. *Information Processing and Management* 25(6):659–664.

EAGLES-I. 1996. Final Report. In *Evaluation of Natural Language Processing Systems*, vol. EAGLES DOCUMENT EAG-EWG-PR.2.

Evert, Stefan. 2004. A simple LNRE model for random character sequences. In *Proceedings of the 7èmes Journées Internationales d'Analyse Statistique des Données Textuelles*, pages 411–422. Louvain-la-Neuve, Belgium.

Fawcett, Tom. 2003. ROC graphs: Notes and practical considerations for data mining researchers. Tech. rep., HP Laboratories, Palo Alto, USA.

Ferrer i Cancho, Ramon. 2005a. Decoding least effort and scaling in signal frequency distributions. *Physica A: Statistical Mechanics and its Applications* 345(1-2):275–284.

Ferrer i Cancho, Ramon. 2005b. The variation of Zipf's law in human language. *European Physical Journal B* 44:249–257.

Ferrer i Cancho, Ramon, Oliver Riordan, and Bela Bollobas. 2005. The consequences of Zipf's law for syntax and symbolic reference. *Proceedings of The Royal Society of London. Series B, Biological Sciences* .

Ferrer i Cancho, Ramon and Ricard V. Solé. 2001. Two regimes in the frequency of words and the origins of complex lexicons: Zipf's law revisited. *Journal of Quantitative Linguistics* 8(3):165–173.

Ferrer i Cancho, Ramon and Ricard V. Solé. 2002. Zipf's law and random texts. *Advances in Complex Systems* 5(1):1–6.

Ferrer i Cancho, Ramon and Ricard V. Solé. 2003. Least effort and the origins of scaling in human language. *PNAS* 100:788–791.

Flach, Peter A. 2003. The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In T. Fawcett and N. Mishra, eds., *ICML*, pages 194–201. AAAI Press. ISBN 1-57735-189-4.

Gadd, T.N. 1990. Phonix: The algorithm. *Program: Automated library and information systems* 24(4):363–366.

Gates, Arthur Irving. 1937. *Spelling difficulties in 3867 words*. New York: Teachers College, Columbia University.

Golding, Andrew R. 1995. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 39–53. ACL, Boston, MA.

Golding, A. R. and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning* 34(1-3):107–130. Special Issue on Machine Learning and Natural Language.

Golding, Andrew R. and Yves Schabes. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *ACL-96*, pages 71–78. ACL, Santa Cruz, CA.

Graff, David. 2003. The New York Times Newswire Service. *English Gigaword LDC-2003T05* .

Grannis, S.J., J.M. Overhage, and C. McDonald. 2004. Real world performance of approximate string comparators for use in patient matching. In M. F. et al., ed., *Medinfo 2004*, pages 43–47. Amsterdam: IOS Press.

Grudin, Jonathan T. 1983. Error patterns in novice and skilled transcription typing. In W. E. Cooper, ed., *Cognitive Aspects of Skilled Typewriting*, pages 121–139. New York: Springer-Verlag.

Ha, Le Quan, E.I. Sicilia-Garcia, Ji Ming, and F.J. Smith. 2003. Extension of Zipf's law to word and character n-grams for English and Chinese. *Computational Linguistics and Chinese Language Processing* 8(1):77–102.

Huang, Jin Hu and David Powers. 2001. Large scale experiments on correction of confused words. In *ACSC '01: Proceedings of the 24th Australasian conference on Computer science*, pages 77–82. IEEE Computer Society. ISBN 0-7695-0963-0.

Ingels, Peter. 1997. *A Robust Text Processing Technique Applied to Lexical Error Recovery*. Master's thesis, Linkoping University, Sweden.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR. ISBN 0130950696.

Kann, Vigo, R. Domeij, J. Hollman, and M. Tillenius. 2001. Implementation aspects and applications of a spelling correction algorithm. In L. Uhlirova, G. Wimmer, G. Altmann, and R. Koehler, eds., *Text as a Linguistic Paradigm: Levels, Constituents, Constructs, volume 60 of Quantitative Linguistics*, pages 108–123. Trier, Germany: Wissenschaftlicher Verlag Trier.

Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *COLING-90*, vol. II, pages 205–211. Helsinki.

Khmaladze, E.V. 1987. The statistical analysis of large number of rare events. Tech. Rep. MS-R8804, Department of Mathematical Statistics, CWI, Amsterdam: Center for Mathematics and Computer Science.

Khmelev, D.V. and W.J. Teahan. 2003. A repetition based measure for verification of text collections and for text categorization. In *Proceedings of the twenty-sixth annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 03)*, pages 104–110.

Kim, W.G. and W. John Wilbur. 2001. Corpus-based statistical screening for content-bearing terms. *Journal of the American Society for Information Science* 52(3):247–259.

Kleinberg, Jon. 2002. Bursty and hierarchical structure in streams. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 91–101. New York, NY, USA: ACM Press. ISBN 1-58113-567-X.

Knuth, Donald E. 1981. *Sorting and Searching*, vol. 2 of *The Art of Computer Programming*, section 6.4, pages 513–558. Reading, Massachusetts: Addison-Wesley, 2nd edn.

Kornai, András. 2002. How many words are there? *Glottometrics* 4:61–86.

Kukich, Karen. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys* 24(4):377–439.

Leech, Geoffrey. 1992. 100 Million words of English: the British National Corpus. *Language Research* 28(1):1–13.

Levenshtein, V.I. 1965. Binary codes capable of correcting deletions, insertions, and reversals. In *Cybernetics and Control Theory*, vol. 10(8), pages 707–710. Original in: Doklady Nauk SSSR 163(4): 845–848 (1965).

Lewis, D., Y. Yang, T.G. Rose, and F. Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research* 5:361–397.

Li, Shanjian and Katsuhiko Momoi. 2001. A composite approach to language/encoding detection. In *Proceedings Nineteenth Unicode Conference*. San Jose, California: Netscape Communications Corp.

Ling, Charles X., Jin Huang, and Harry Zhang. 2003. AUC: a statistically consistent and more discriminating measure than accuracy. In *IJCAI*, pages 519–526.

Lita, Lucian Vlad, Abraham Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEcasIng. In *ACL-03*, pages 152–159.

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

Mays, Eric, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management* 27(5):517–522.

McIlroy, M.D. 1982. Development of a spelling list. *IEEE Trans. on Communications* 30:91–99.

Mihalcea, Rada F. and Vivi A. Nastase. 2002. Letter level learning for language independent diacritics restoration. In *Proceedings of CoNLL-2002*, pages 105–111. Taipei, Taiwan.

Morris, R. and L.L. Cherry. 1975. Computer detection of typographical errors. *IEEE Trans. on Professional Communication* PC-18(1):54–56.

Navarro, Gonzalo. 2001. A guided tour to approximate string matching. *ACM Computing Surveys* 33(1):31–88.

Odell, Margaret K. and Robert C. Russell. 1918/1922. U.S. Patents 1261167 (1918), 1435663 (1922). Cited in Knuth (1973).

Oflazer, Kemal. 1993. Two-level description of Turkish morphology. In *Proceedings, Sixth Conference of the European Chapter of the ACL.*

Ordelman, Roeland. 2003. *Dutch speech recognition in Multimedia Information Retrieval*. Ph.D. thesis, University of Twente, Enschede, The Netherlands. CTIT Ph.D.-series No. 03-56.

Paggio, P. and N.L. Underwood. 1995. Validating the TEMAA LE evaluation methodology: a case study on Danish spelling checkers. *Natural Language Engineering* 1(1):1–18.

Peterson, James L. 1986. A note on undetected typing errors. *Communications of the ACM* 29(7):633–637.

Philips, Lawrence. 1990. Hanging on the Metaphone. *Computer Language Magazine* 7(12).

Philips, Lawrence. 2000. The Double Metaphone search algorithm. *C/C++ Users Journal.* June 2000.

Pollock, J.J. and A. Zamora. 1983. Collection and characterization of spelling errors in scientific and scholarly text. *Journal of the American Society for Information Science* 34(1):51–58.

Pollock, Joseph J. and Antonio Zamora. 1984. Automatic spelling correction in scientific and scholarly text. *Commun. ACM* 27(4):358–368.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program* 14(3):130–127.

Powers, David M.W. 1997. Learning and application of differential grammars. In *CoNLL97. Proceedings of the Meeting of the ACL Special Interest Group in Natural Language Learning*, pages 88–96. Madrid: Association for Computational Linguistics.

Powers, David M.W. 1998. Applications and explanations of Zipf's law. In *Proceedings of the Joint International Conference on New Methods in Language Processing and Computational Natural Language Learning (NeMLaP3/CoNLL98)*, pages 151–160. Sydney: Somerset NL: ACL/SIGNLL/Flinders Univ.

Provost, Foster J., Tom Fawcett, and Ron Kohavi. 1998. The case against accuracy estimation for comparing induction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Reynaert, Martin. 2004. Text Induced Spelling Correction. In *Proceedings COLING 2004, Geneva*.

Rose, T., M. Stevenson, and M. Whitehead. 2002. The Reuters Corpus Volume 1—from yesterday's news to tomorrow's language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31. Las Palmas de Gran Canaria: IEEE Computer Society Press.

Sigurd, Bengt, Mats Eeg-Olofsson, and Joost van de Weijer. 2004. Word length, sentence length and frequency - Zipf revisited. *Studia Linguistica* 58(1):37–52.

Simard, Michel and Alexandre Deslauriers. 2001. Real-time automatic insertion of accents in French text. *Natural Language Engineering* 7(2):149–165.

Solé, Ricard V. Forthcoming. Scaling law in language evolution. In C. Cioffi-Revilla, ed., *Scaling laws in the social sciences*. Cambridge University Press.

Starlander, Marianne and Andrei Popescu-Belis. 2002. Corpus-based evaluation of a French spelling and grammar checker. In *LREC 2002 : Third International Conference on language resources and evaluation*, vol. 1, pages 268–274. Las Palmas de Gran Canaria, Spain: Paris : ELRA, European Language Resources.

Taghva, Kazem, Thomas Nartker, and Julie Borsack. 2004. Information access in the presence of OCR errors. In *HDP '04: Proceedings of the 1st ACM workshop on Hardcopy document processing*, pages 1–8. ACM Press. ISBN 1-58113-976-4.

Taghva, Kazem and Eric Stofsky. 2001. OCRSpell: an interactive spelling correction system for OCR errors in text. *International Journal on Document Analysis and Recognition* 3(3):125–137.

temaa. 1996. temaa. *A Testbed study of Evaluation Methodologies: Authoring Aids - Final Report - LRE-62-070*. Center for Sprogteknologi, Copenhagen, Denmark.

Tillenius, Mikael. 1996. *Efficient generation and ranking of spelling error corrections*. Master's thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden. NADA report TRITA-NA-E9621.

Tomita, M. 1986. *Efficient Parsing for Natural Language*. Dordrecht - Boston - Lancaster: Kluwer Academic Publishers.

Toutanova, Kristina and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 144–151.

Ukkonen, Esko. 1985. Algorithms for approximate string matching. *Information Control* 64(1-3):100–118.

Ukkonen, Esko. 1992. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science* 92(1):191–211.

van Berkel, Brigitte and Koenraad de Smedt. 1988. Triphone analysis: a combined method for the correction of orthographical and typographical errors. In *Proceedings of the second conference on Applied natural language processing*, pages 77–83. Morristown, NJ, USA: Association for Computational Linguistics.

van den Heuvel, Theo. 2003. De spelling onder controle? - waarom de spellingcorrector niet altijd doet wat we willen. *Onze Taal* 72(9):236–238.

van Rijsbergen, C. J. 1975. *Information Retrieval*. London: Butterworths.

Vosse, Theo. 1992. Detecting and correcting morpho-syntactic errors in real texts. In *Proceedings of the third conference on Applied natural language processing*, pages 111–118. Morristown, NJ, USA: Association for Computational Linguistics.

Vosse, Theo. 1994. *The Word Connection. Grammar-based spelling error correction in Dutch*. Ph.D. thesis, Rijksuniversiteit Leiden, Leiden, The Netherlands.

Wagner, Robert A. and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery* 21:168–173.

Webster's Third New International Dictionary. 1981. *Webster's Third New International Dictionary of the English Language Unabridged*. Springfield, MA, USA: Merriam-Webster. ISBN 0-87779-201-1, 0-87779-206-2.

Woordenlijst Nederlandse Taal, I. 1995. *Samengesteld door het Instituut voor Nederlandse Lexicografie in opdracht van de Nederlandse Taalunie*. Den Haag: SDU Uitgevers.

Yarowsky, David. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *ACL-94*, pages 88–95. ACL, Las Cruces, NM.

Yarowsky, David. 1999. Corpus-based techniques for restoring accents in Spanish and French text. In S. Armstrong and K. W. C. et al., eds., *Natural Language Processing Using Very Large Corpora*, pages 99–120. Dordrecht - The Netherlands: Kluwer Academics Publisher.

Zipf, George Kingsley. 1935. *The psycho-biology of language: an introduction to dynamic philology*. Cambridge, MA: The M.I.T. Press, 2nd edn.

Zobel, Justin and Philip Dart. 1995. Finding approximate matches in large lexicons. *Software – Practice and Experience* 25(3):331–345.