Spelling Correction for Search Engine Queries

Bruno Martins and Mário J. Silva

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
1749-016 Lisboa, Portugal

bmartins@xldb.di.fc.ul.pt,mjs@di.fc.ul.pt

Abstract Search engines have become the primary means of accessing information on the Web. However, recent studies show misspelled words are very common in queries to these systems. When users misspell query, the results are incorrect or provide inconclusive information. In this work, we discuss the integration of a spelling correction component into tumba!, our community Web search engine. We present an algorithm that attempts to select the best choice among all possible corrections for a misspelled term, and discuss its implementation based on a ternary search tree data structure.

1 Introduction

Millions of people use the Web to obtain needed information, with search engines currently answering tens of millions of queries every day. However, with the increasing popularity of these tools, spelling errors are also increasingly more frequent. Between 10 to 12 percent of all query terms entered into Web search engines are misspelled [10]. A large number of Web pages also contain misspelled words. Web search is thus a task of information retrieval in an environment of faulty texts and queries. Even with misspelled terms in the queries, search engines often retrieve several matching documents – those containing spelling errors themselves. However, the best and most "authoritative" pages are often missed, as they are likely to contain only the correctly spelled forms. An interactive spelling facility that informs users of possible misspells and presents appropriate corrections to their queries could bring improvements in terms of precision, recall, and user effort. Google was the first major search engine to offer this facility [8].

One of the key requirements imposed by the Web environment on a spelling checker is that it should be capable of selecting the best choice among all possible corrections for a misspelled word, instead of giving a list of choices as in word processor spelling checking tools. Users of Web search systems already give little attention to query formulation, and we feel that overloading them with an interactive correction mechanism would not be well accepted. It is therefore important to make the right choice among all possible corrections autonomously.

This work presents the development of a spelling correction component for tumba!, our community search engine for the Portuguese Web [29]. In tumba! we check the query for misspelled terms while results are being retrieved. If errors are detected, we

provide a suggestive link to a new "possibly correct" query, together with the search results for the original one.

The rest of this paper is organized as follows: the next section presents the terminology used throughout this work. Section 3 gives an overview on previous approaches to spelling correction. Section 4 presents the ternary search tree data structure, used in our system for storing the dictionary. Section 5 details our algorithm and the heuristics behind it. Section 6 describes the data sources used to build the dictionary. Section 7 describes experimental results. Finally, Section 8 points our conclusions and directions for future work.

2 Terminology

Information Retrieval (IR) concerns with the problem of providing relevant documents in response to a user's query [2]. The most commonly used IR tools are **Web search engines**, which have become a fact of life for most Internet users. Search engines use software robots to survey the Web, retrieving and indexing HTML documents. Queries are checked against the keyword indexes, and the best matches are returned.

Precision and **Recall** are the most popular metrics in evaluating IR systems. Precision is the percentage of retrieved documents that the searcher is actually interested on. Recall, on the other hand, is the percentage of relevant documents retrieved from the set of all documents, this way referring to how much information is retrieved by the search. The ultimate goal of an information retrieval system is to achieve recall with high precision.

Spelling has always been an issue in computer-based text tools. Two main problems can be identified in this context: **Error detection**, which is the process of finding misspelled words, and **Error correction**, which is the process of suggesting correct words to a misspelled one. Although other approaches exist, most spelling checking tools are based on a **dictionary** which contains a set of words which are considered to be correct. The problem of spelling correction can be defined abstractly as follows: Given an alphabet σ, a dictionary D consisting of strings in σ^* and a string s, where $s \notin D$ and $s \in \sigma^*$, find the word $w \in D$ that is most likely to have been erroneously input as s.

Spelling errors can be divided into two broad categories: **typographic errors**, which occur because the typist accidentally presses the wrong key, presses two keys, presses the keys in the wrong order, etc; and **phonetic errors**, where the misspelling is pronounced the same as the intended word but the spelling is wrong. Phonetic errors are harder to correct because they distort the word more than a single insertion, deletion or substitution. In this case, we want to be able to key in something that sounds like the misspelled word (a "phonetic code") and perform a "fuzzy" search for close matches. The search for candidate correct forms can be done at typographic level, and then refined using this method.

3 Related Work

Web information retrieval systems have been around for quite some time now, having become the primary means of accessing information the Web [1,8]. Early systems

engines did not check query spelling but since April 2001, several webwide search engines, including Excite and Google, provide dynamic spelling checking, while others such as Yahoo, simply tracked common misspellings of frequent queries, such as movie star names. Technical details for these systems are unavailable, but they seem to be based on spelling algorithms and statistical frequencies.

Algorithmic techniques for detecting and correcting spelling errors in text has also a long and robust history in computer science [20]. Previous studies have also addressed the use of spelling correctors in the context of user interfaces [13]. Spelling checkers (sometimes called "spell checkers" by people who need syntax checkers) are nowadays common tools for many languages, and many proposals can also be found on the literature. Proposed methods include edit distance [11,31,21], rule-based techniques [32], *n*-grams [25,33]probabilistic techniques [18], neural nets [28,6,17], similarity key techniques [34,23], or combinations [16,22]. All of these methods can be thought of as calculating a distance between the misspelled word and each word in the dictionary. The shorter the distance, the higher the dictionary word is ranked as a good correction.

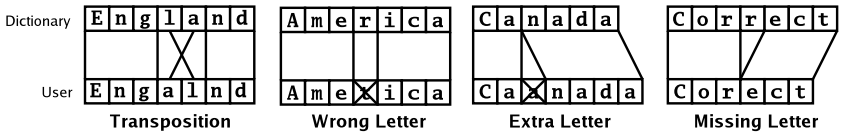

Figure 1. The four most common spelling errors.

Edit distance is a simple technique. The distance between two words is the number of editing operations required to transform one into another. Analysis of errors – mainly typing errors – in very large text files have found that the great majority of wrong spellings (80-95%) differ from the correct spellings in just one of the four ways described in Figure 1. The editing operations to consider should therefore correspond to these four errors, and candidate corrections include the words that differ from the original in a minimum number of editing operations [11]. Recent works are experimenting with modeling more powerful edit operations, allowing generic string-to-string edits [7]. Additional heuristics are also typically used to complement techniques based on edit distance. For instance, in the case of typographic errors, the keyboard layout is very important. It is much more usual to accidentally substitute a key by another if they are placed near each other on the keyboard.

Similarity key methods are based on transforming words into similarity keys that reflect their characteristics. The words in the dictionary and the words to test are both transformed into similarity keys. All words in the dictionary sharing the same key with a word being tested are candidates to return as corrections. An example of this method is the popular Soundex system. Soundex (the name stands for "Indexing on sound") was devised to help with the problem of phonetic errors [12,19]. It takes an English word and produces a four digit representation, in a rough-and-ready way designed to preserve the salient features of the phonetic pronunciation of the word.

The metaphone algorithm is also a system for transforming words into codes based on phonetic properties [23,24]. However, unlike Soundex, which operates on a letter-by-letter scheme, metaphone analyzes both single consonants and groups of letters called diphthongs, according to a set of rules for grouping consonants, and then mapping groups to metaphone codes. The disadvantage of this algorithm is that it is specific to the English language. A version of these rules for the Portuguese language has, to the best of our knowledge, not yet been proposed. Still, there has been recent research on machine learning methods for letter-to-phoneme conversion [15,30]. Application of these techniques to Portuguese should be straightforward, providing he have enough training data.

More recent studies on error correction propose the use of context, attempting to detect words which are misused but spelled correctly [5,14]. Spelling checkers based on isolated word methods would see the following sentence as correct: *a paragraph cud half mini flaws but wood bee past by the spill checker*. However, since in search engines users oddly type more than tree terms for a query, it would be a waste to make context dependent correction. Isolated word methods should prove sufficient for our task.

4 Ternary Search Trees

In this work, we use a ternary search tree (TST) data structure for storing the dictionary in memory. TSTs are a type of trie that is limited to three children per node [3,4]. Trie is the common definition for a tree storing strings, in which there is one node for every common prefix and the strings are stored in extra leaf nodes. TSTs have been successfully used for several years in searching dictionaries. Search times in this structure are $O(log(n)+k)$ in the worst case, where n is the number of strings in the tree and k is the length of the string being searched for. In a detailed analysis of various implementations of trie structures, the authors concluded that "*Ternary Search Tries are an effective data structure from the information theoretic point of view since a search costs typically about $log(n)$ comparisons on real life textual data. [...] This justifies using ternary search tries as a method of choice for managing textual data*" [9].

Figure 2 illustrates a TST. The structure stores key-value pairs, where keys are the words and values are integers corresponding to the word frequency. As we can see, each node of the tree stores one letter and has three children. A search compares the current character in the search string with the character at the node. If the search character comes lexically first, the search goes to the left child; if the search character comes after, the search goes to the right child. When the search character is equal, the search goes to the middle child, and proceeds to the next character in the search string.

TSTs combine the time efficiency of tries with the space efficiency of binary search trees. They are faster than hashing for many typical search problems, and support a broad range of useful operations, like finding all keys having a given prefix, suffix, or infix, or finding those keys that closely match a given pattern.

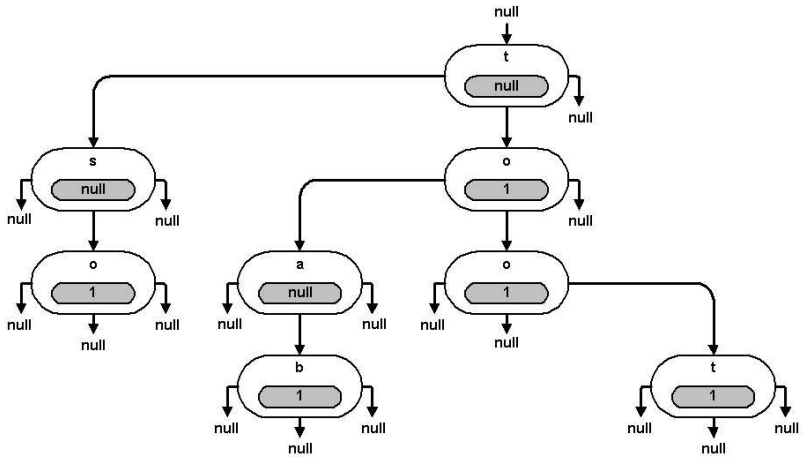

Figure 2. A ternary search tree storing the words “to”, “too”, “toot”, “tab” and “so”, all with an associated frequency of 1.

5 Spelling Correction Algorithm

A TST data structure stores the dictionary. For each stored word, we also keep a frequency count, originally obtained from the analysis of a large corpora. To choose among possible corrections for a misspelled word, we use these word frequency counts as a popularity ranking, together with other information such as metaphone keys. Although we do not have a specific text-to-phoneme algorithm for the Portuguese language, using the standard metaphone algorithm yields in practice good results.

Queries entered in the search engine are parsed and the individual terms are extracted, with non word tokens ignored. Each word is then converted to lower case, and checked to see if it is correctly spelled. Correctly spelled words found in user queries are updated in the dictionary, by incrementing their frequency count. This way, we use the information in the queries as feedback to the system, and the spelling checker can adapt to the patterns in user’s searches by adjusting its behavior. For the misspelled words, a correctly spelled form is generated. Finally, a new query is presented to the user as a suggestion, together with the results page for the original query. By clicking on the suggestion, the user can reformulate the query.

Our system integrates a wide range of heuristics and the algorithm used for making the suggestions for each misspelled word is divided in two phases. In the first, we generate a set of candidate suggestions. In the second, we select the best.

The first phase of the algorithm can be further decomposed into 9 steps. In each step, we look up the dictionary for words that relate to the original misspelling, under specific conditions:

1. Differ in one character from the original word.
2. Differ in two characters from the original word.

3. Differ in one letter removed or added.
4. Differ in one letter removed or added, plus one letter different.
5. Differ in repeated characters removed.
6. Correspond to 2 concatenated words (space between words eliminated).
7. Differ in having two consecutive letters exchanged and one character different.
8. Have the original word as a prefix.
9. Differ in repeated characters removed and 1 character different

In each step, we also move on directly to the second phase of the algorithm if one or more matching words are found (i.e., if there are candidate correct forms that only differ in one character from the original misspelled word, a correct form that differs in more characters and is therefore more complex will never be chosen).

In the second phase, we start with a list of possible corrections. We then try to select the best one, following these heuristics:

1. If there is one solution that differs only in accented characters, we automatically return it. Typing words without correct accents is a very common mistake in the Portuguese language (20% according to Medeiros [22]).
2. If there is one solution that differs only in one character, with the error corresponding to an adjacent letter in the same row of the keyboard (the QWERTY layout is assumed), we automatically return it.
3. If there are solutions that have the same metaphone key as the original string, we return the smallest one, that is, the one with less characters.
4. If there is one solution that differs only in one character, with the error corresponding to an adjacent letter in an adjacent row of the keyboard, we automatically return it.
5. In the last case, we return the smallest word.

We follow the list of heuristics sequentially, and only move to the next if no matching words are found. If there is more than one word satisfying the conditions for each heuristic, we first try to return the one where the first character is equal to the correctly spelled word. If there is still more than one word, we return the one that has the highest frequency count.

6 Data Sources and Associated Problems

The dictionary for the spelling checking system is a normal text file, where each line contains a term and its associated frequency. The sources of Portuguese words and word frequencies for the dictionary were the texts from the Natura-Publico and the Natura-Minho corpora [27,26]. The first one is made of the two first paragraphs of news articles from *Publico* in the years of 1991, 1992, 1993 and 1994. The second, corresponds to the full articles in 86 days of editions of the newspaper *Diario do Minho*, spread across the years of 1998 and 1999.

The dictionary strongly affects the quality of a spelling checking system. If it is too small, not only will the candidate list for misspellings be severely limited, but the user will also be frustrated by too many false rejections of words that are correct. On the

other hand, a lexicon that is too large may not detect misspellings when they occur, due to the dense "word space".

News articles capture the majority of the words commonly used, as well as technical terms, proper names, common foreign words, or references to entities. However, such large corpora often contain many spelling errors [26]. We use word frequencies to choose among possible corrections, which to some extent should deal with this problem. As misspelled terms are, in principle, less frequent over the corpus than their corresponding correct form, only on rare occasions should the spelling checker provide an erroneous suggestion.

The Web environment introduces difficulties. It is general in subject, as opposed to domain specific, and multilingualism issues are also common. While spelling checkers in text editors use standard and personal dictionaries, search engine spelling checkers should be more closely tied to the content they index, providing suggestions based on the content of the corpus. This would avoid the dead-end effect of suggesting a word that is correctly spelled but not included in any words on the site, and add access to names and codes which will not be in any dictionary. However, using a search engine's inverted index as the basis of the spelling dictionary only works well when the content has been copy edited, or when an editor is available to check the word list and reject misspellings.

7 Evaluation Experiments

Some experiments were performed in order to quantitatively evaluate our spelling correction mechanism.

We were first interested in evaluating the quality of the proposed suggestions. To achieve this, we compared the suggestions produced by our spelling checker against Aspell – see the project homepage at `http://aspell.sourceforge.net/`. Aspell is a popular interactive spelling checking program for Unix environments. Its strength comes from merging the metaphone algorithm with a near miss strategy, this way correcting phonetic errors and making better suggestions for seriously misspelled words. The algorithm behind Aspell is therefore quite similar to the one used in our work, and the quality of the results in both systems should be similar.

We used a hand-compiled list of 120 common misspellings, obtained from *CiberDúvidas da Lingua Portuguesa* (`http://ciberduvidas.sapo.pt/\-php/\-glossario.php`) and by inspecting the query logs for the search engine. The table below shows the list of misspelled terms used, the correctly spelled word, and the suggestions produced. In the table, a "*" means that the algorithm did not detect the misspelling and a "-" means the algorithm failed in returning a suggestion.

Correct Form	Spelling Error	Our Algorithm	ASpell
ameixial	ameixeal	ameixial	ameixial
artífice	artífece	artífice	artífice
camoniano	camoneano	camoniano	camoniano
definido	defenido	definido	defendo
lampião	lampeão	lampião	lampião
oficina	ofecina	oficina	oficina

Continued on next page

Table 1 – continued from previous page

Correct Form	Spelling Error	Our Algorithm	ASpell
acerca	àcerca	acerca	acerca
açoriano	açoreano	açoriano	coreano
alcoolémia	alcoolemia	*	*
antepor	antepôr	*	antepor
árctico	artico	artigo	aórtico
antárctico	antártico	catártico	antárctico
bainha	baínha	bainha	bainha
bebé	bébé	bebé	bebe
bege	beje	*	beije
bênção	benção	*	*
beneficência	beneficiência	beneficência	beneficência
biopsia	biópsia	*	*
burburinho	borborinho	burburinho	burburinho
caiem	caem	*	*
calvície	calvíce	calvície	calvície
campeão	campião	campeão	campeão
comboio	combóio	comboio	comboio
compor	compôr	*	compor
comummente	comumente	comovente	comummente
constituia	constituía	*	*
constituiu	constituíu	constituiu	constituiu
cor	côr	*	*
crânio	crâneo	crânio	cárneo
despretensioso	despretencioso	despretensioso	despretensioso
pretensioso	pretencioso	pretensioso	pretensioso
definição	defenição	definição	definição
definir	defenir	definir	definir
desiquilíbrio	desequilibrio	desequilíbrio	desequilíbrio
dispender	despender	*	*
dignatários	dignitários	dignatários	digitarias
dispêndio	dispendio	*	dispêndio
ecrã	ecran	*	écran
emirados	emiratos	estratos	méritos
esquisito	esquesito	esquisito	esquisito
estratego	estratega	*	*
feminino	femenino	feminino	feminino
feminismo	femininismo	-	feminismo
fôr	for	*	*
gineceu	geneceu	gineceu	gineceu
gorjeta	gorgeta	gorjeta	gorjeta
granjear	grangear	granjear	granjear
guisar	guizar	guisar	gizar
hectare	hectar	*	*
halariedade	hilaridade	*	*
hiroshima	hiroxima	aproxima	próxima
ilacção	elação	ilação	ilação
indispensável	indespensável	indispensável	indispensável
inflacção	inflação	*	*
interveio	interviu	intervir	inter viu
intervindo	intervido	*	*
invocar	evocar	*	*
ípsilon	ipslon	ípsilon	ípsilon
irisar	irizar	irisar	razar
irupção	irrupção	*	*
esotérico	isotérico	*	*
jeropiga	geropiga	-	geórgia
juiz	juíz	*	juiz
lêem	lêm	lês	lema
linguista	linguísta	*	linguista
lisonjear	lisongear	lisonjear	lisonjear
logótipo	logotipo	logo tipo	logo tipo
saem	saiem	saiam	saem
saloiice	saloice	baloice	saloiice
sarjeta	sargeta	sarjeta	sarjeta
semear	semiar	semear	semear

Continued on next page

Table 1 – continued from previous page

Correct Form	Spelling Error	Our Algorithm	ASpell
suíça	suiça	suíça	suíça
supor	supôr	*	supor
rainha	raínha	rainha	rainha
raiz	raíz	*	raiz
raul	raúl	raul	raul
rédea	rédia	rédea	radia
regurgitar	regurjitar	regurgitar	regurgitar
rejeitar	regeitar	rejeitar	regatar
requeiro	requero	requere	requeiro
réstia	réstea	réstia	resta
rectaguarda	retaguarda	*	*
rubrica	rúbrica	*	*
quadricromia	quadricomia	-	quadriculai
quadruplicado	quadriplicado	quadruplicado	quadruplicado
quasímodo	quasimodo	-	quisido
quilo	kilo	*	Nilo
quilograma	kilograma	holograma	holograma
quilómetro	kilómetro	milímetro	milímetro
quis	quiz	quis	qui
paralisar	paralizar	paralisar	paralisar
perserverança	preseverança	perseverança	perseverança
persuasão	persuação	persuasão	persuasão
persuasão	presuasão	persuasão	persuasão
pirinéus	pirenéus	*	*
privilégio	previlégio	privilégio	privilegio
oceânia	oceania	*	*
opróbrio	opróbio	aeróbio	proíbo
organograma	organigrama	*	*
nonagésimo	nonagessimo	nonagésimo	nonagésimo
maciço	massiço	mássico	mássico
majestade	magestade	majestade	majestade
manjerico	mangerico	manjerico	manjerico
manjerona	mangerona	tangerina	tangerina
meteorologia	metereologia	meteorologia	meteorologia
miscigenação	miscegenação	miscigenação	miscigenação
trânsfuga	transfuga	transfira	transfira
transpôr	transpor	*	*
urano	úrano	*	*
ventoinha	ventoínha	ventoinha	ventoinha
verosímil	verosímel	*	*
vigilante	vegilante	vigilante	vigilante
vôo	voo	*	*
vultuoso	vultoso	*	*
xadrez	xadrês	xadrez	ladres
xamã	chamã	chama	chama
xelindró	xilindró	cilindro	cilindro
chiita	xiita	*	xiitas
zângão	zangão	*	*
zepelin	zeppelin	-	zeplim
zoo	zoô	zoo	coo

48.33% of the correct forms were correctly guessed and our algorithm outperformed Aspell by a slight margin of 1.66%. On the 120 misspellings, our algorithm failed in detecting a spelling error 38 times, and it failed on providing a suggestion only 5 times. Note that the data source used to build the dictionary has itself spelling errors. A careful process of reviewing the dictionary could improve results in the future.

Kukich points out that most researchers report accuracy levels above 90% when the first three candidates are considered instead of the first guess [20]. Guessing the one right suggestion to present to the user is much harder than simply identifying misspelled words and present a list of possible corrections.

In the second experiment, we took some measures from the integration of our spelling checker with a search engine for the Portuguese Web. We tried to see if, by using the spelling correction component, there were improvements in terms of precision and recall in our system. Using a hand compiled list of misspelled queries, we measured the number of retrieved documents in the original query, and the number of retrieved documents in the transformed query. We also had an human evaluator accessing the quality of the first ten results returned by the search engine, that is, measuring how many documents in the first ten results were relevant to the query.

Misspelled Query	# Relevant Results	Correct Query	# Relevant Results
camoneano	5	camoniano	10
açoreano	10	açoriano	10
calvíce	3	calvície	10
campião	9	campeão	10
femenino	9	feminino	10
guizar	6	guisar	10
raínha	10	rainha	10
regurjitar	0	regurgitar	10
magestade	9	majestade	10
mangerico	9	manjerico	10
metereologia	10	meteorologia	10
vegilante	0	vigilante	10
xadrês	9	xadrez	10
zôo	0	zoo	10

Table 2. Results from the Integration of the Spelling Checker With Tumba!

Results confirm our initial hypothesis that integrating spelling correction in Web search tools can bring substantial improvements. Although many pages were returned in response to misspelled queries (and in some cases all pages were indeed relevant), the results for the correctly spelled queries were always of better quality and more relevant.

8 Conclusions and Future Work

This paper presented the integration of a spelling correction component into tumba!, a Portuguese community Web search engine. The key challenge in this work was determining how to pick the most appropriate spelling correction for a mistyped query from a number of possible candidates.

The spelling checker uses a ternary search tree data structure for storing the dictionary. As source data, we used a large textual corpus of from two popular Portuguese newspapers. The evaluation showed that our system gives results of acceptable quality, and that integrating spelling correction in Web search tools can be beneficial. However, the validation work could be improved with more test data to support our claims.

An important area for future work concerns phonetic error correction. We would like to experiment with machine learning text-to-phoneme techniques that could adapt to the Portuguese language, instead of using the standard metaphone algorithm [15,30]. We also find that queries in our search engine often contain company names, acronyms, foreign words and names, etc. Having a dictionary that can account for all these cases

is very hard, and large dictionaries may result in inability to detect misspellings due to the dense "word space". However, keeping two separate dictionaries, one in the TST used for correction and another in an hash-table used only for checking valid words, could yield interesting results. Studying ways of using the corpus of Web pages and the logs from our system, as the basis for the spelling checker, is also a strong objective for future work. Since our system imports dictionaries in the form of ASCII word lists, we do however have an infrastructure that facilitates lexicon management.

9 Acknowledgments

Special thanks to our colleagues and fellow members of the tumba! development, and to the various members of Linguateca, for their valuable insights and suggestions. This research was partially supported by the FCCN - Fundação para a Computação Científica Nacional, FCT - Fundação para a Ciência e Tecnologia, and FFCUL - Fundação da Faculdade de Ciências da Universidade de Lisboa, under grants POSI/SRI/47071/2002- (project GREASE) and SFRH/BD/10757/2002 (FCT scholarship).

References

1. A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the web. *ACM Transactions on Internet Technology*, 1(1):2–43, 2001.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.
3. J. Bentley and R. Sedgewick. Fast algorithms for sorting and searching strings. In *Proceedings of SODA-97, the 8th ACM-SIAM Symposium on Discrete Algorithms*, 1997.
4. J. Bentley and R. Sedgewick. Ternary search trees. *Dr. Dobb's Journal*, 23(4):20–25, April 1998.
5. J. Bigert. Probabilistic detection of context-sensitive spelling errors. In *Proceedings of LREC-2004, the 4th International Conference on Language Resources and Evaluation*, 2004.
6. A. G. Bonfante. Uso de redes neurais para correção gramatical do português: Um estudo de caso. Master's thesis, Instituto de Ciências Matemáticas e da Computação da Universidade de São Paulo, São Carlos, São Paulo, Brazil, 1997. Dissertação de Mestrado.
7. E. Brill and R. C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of ACL-2000, the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, 2000.
8. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
9. J. Clément, P. Flajolet, and B. Vallée. The analysis of hybrid trie structures. In *Proceedings of DA-98, the 9th annual ACM-SIAM symposium on discrete algorithms*, pages 531–539. Society for Industrial and Applied Mathematics, 1998.
10. H. Dalianis. Evaluating a spelling support in a search engine. In *Proceedings of NLDB-2002, the 7th International Workshop on the Applications of Natural Language to Information Systems*, June 2002.
11. F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.
12. L. Davidson. Retrieval of mis-spelled names in an airline passenger record system. *Communications of the ACM*, 5(3):169–171, March 1962.

13. I. Durham, D. A. Lamb, and J. B. Saxe. Spelling correction in user interfaces. *Communications of the ACM*, 26(10):764–773, October 1983.
14. M. A. Elmi and M. Evens. Spelling correction using context. In C. Boitet and P. Whitelock, editors, *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 360–364, San Francisco, California, 1998. Morgan Kaufmann Publishers.
15. W. M. Fisher. A statistical text-to-phone function using n-grams and rules. In *Proceedings of ICASSP-99, the 1999 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 649–652, March 1999.
16. V. J. Hodge and J. Austin. An evaluation of phonetic spell checkers. Technical Report YCS 338, Department of Computer Science of the University of York, 2001.
17. V. J. Hodge and J. Austin. A novel binary spell checker. In *Proceedings of ICANN-01, the 11th International Conference on Articial Neural Networks*, August 2001.
18. R. L. Kashyap and J. Oommen. Spelling correction using probabilistic methods. *Pattern Recognition Letters*, 1985.
19. D. E. Knuth. *The Art of Computer Programming*, volume 3 / Sorting and Searching. Addison-Wesley Publishing Company, Reading, Massachusetts, 2nd edition, 1982.
20. K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–440, 1992.
21. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.
22. J. C. D. Medeiros. Processamento morfológico e correcção ortográfica do português. Master's thesis, Instituto Superior Técnico, 1995.
23. L. Philips. Hanging on the metaphone. *Computer Language*, 7(12):39–43, 1990.
24. L. Philips. The double-metaphone search algorithm. *C/C++ User's Journal*, 18(6), June 2000.
25. E. M. Riseman and A. R. Hanson. A contextual postprocessing system for error correction using binary n-grams. *IEEE Transactions on Computer Systems*, C-23(5):480–493, May 1974.
26. D. Santos and P. Rocha. Evaluating cetempúblico, a free resource for portuguese. In *Proceedings of ACL-2001, the 39th Annual Meeting of the Association for Computational Linguistics*, pages 442–449, July 2001.
27. D. Santos and L. Sarmento. O projecto AC/DC: acesso a corpora / disponibilização de corpora. In A. Mendes and T. Freitas, editors, *Actas do XVIII Encontro da Associação Portuguesa de Linguística*, pages 705–717, October 2002.
28. T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.
29. M. J. Silva. The case for a portuguese Web search engine. DI/FCUL TR 03–03, Department of Informatics, University of Lisbon, March 2003.
30. K. Toutanova and R. C. Moore. Pronunciation modeling for improved spelling correction, July 2002.
31. R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Communications of the ACM*, 1(21):168–173, 1974.
32. E. J. Yannakoudakis. Expert spelling error analysis and correction. In K. P. Jones, editor, *Proceedings of a Conference held by the Aslib Informatics Group and the Information Retrieval Group of the British Computer Society*, pages 39–52, March 1983.
33. E. M. Zamora, J. J. Pollock, and A. Zamora. The use of trigram analysis for spelling error detection. *Information Processing and Management*, 6(17):305–316, 1981.
34. J. Zobel and P. Dart. Phonetic string matching: Lessons from information retrieval. In *Proceedings of SIGIR-96, the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 166–172, 1996.