Evaluating phonetic spellers for user-generated content in Brazilian Portuguese

Gustavo Augusto de Mendonça Almeida, Lucas Avanço, Magali Sanches Duran, Erick Fonseca, Maria das Graças Volpe Nunes, and Sandra Maria Aluísio

Interinstitutional Center for Computational Linguistics (NILC)
Institute of Mathematical and Computer Sciences
University of São Paulo
gustavoauma@gmail.com, avanco89@gmail.com,
magali.duran@uol.com.br, erickrfonseca@gmail.com,
gracan@icmc.usp.br, sandra@icmc.usp.br

Abstract. Recently, spell checking (or spelling correction systems) has regained attention due to the need of normalizing user-generated content (UGC) on the web. UGC presents new challenges to spellers, as its register is much more informal and contains much more variability than traditional spelling correction systems can handle. This paper proposes two new approaches to deal with spelling correction of UGC in Brazilian Portuguese (BP), both of which take into account phonetic errors. The first approach is based on three phonetic modules running in a pipeline. The second one is based on machine learning, with soft decision making, and considers context-sensitive misspellings. We compared our methods with others on a human annotated UGC corpus of reviews of products. The machine learning approach surpassed all other methods, with 78.0% correction rate, very low false positive (0.7%) and false negative rate (21.9%).

1 Introduction

Spell checking is a very well-known and studied task of natural language processing (NLP), being present in applications used by the general public, including word processors and search engines. Most of spell checking methods are based on large dictionaries to detect non-words, mainly related to typographic errors caused by key adjacency or fast key stroking. Currently, with the recent boom of mobile devices with small touchscreens and tiny keyboards, one can miss the keystrokes, and thus spell checking has regained attention [1].

Dictionary-based approaches can be ineffective when the task is to detect and correct spelling mistakes which coincidentally correspond to existing words (real-word errors). Different from non-word errors, real-word errors generate variant forms that are ambiguous with other words in the language and must be addressed considering context. For instance, in "eu vou comprar" (*I will buy*-INF), if the last character from "comprar" (*to buy*-INF) is deleted, it will produce "compra" (*buys*-PRES.3SG) which is also a valid verb form in BP. Therefore, disambiguation must be performed taking into account the surrounding tokens: it is rare to have an inflected verb form after the auxiliary "vou", much more likely is to observe an infinitive in such context to make the compound future tense "vou comprar" (will buy).

Several approaches have been proposed to deal with these errors: mixed trigram models [2], confusion sets [3], improvements on the trigram-based noisy-channel model [4] and [5], use of GoogleWeb 1T 3-gram data set and a normalized and modified version of the Longest Common Subsequence string matching algorithm [6], a graph-based method using contextual and PoS features and the double metaphone algorithm to represent phonetic similarity [7]. As an example, although MS Word (from 2007 version onwards) claims to include a contextual spelling checker, an independent evaluation of it found high precision but low recall in a sample of 1400 errors [8]. Hunspell [9], the open-source speller for LibreOffice, uses n-gram similarity, rules and dictionary based pronunciation data in order to provide suggestions for spelling errors.

Errors due to phonetic similarity also impose difficulties to spell checkers. They occur when a writer knows well the pronunciation of a word but not how to spell it. This kind of error requires new approaches to combine phonetic models and models for correcting typographic and/or real-word errors. In [10], for example, the authors use a linear combination of two measures – the Levenshtein distance between two strings and the Levenshtein distance between their Soundex [11] codes.

In the last decade, some researchers have revisited spell checking issues motivated by web applications, such as search query engines and sentiment analysis tools based on UGC, e.g. Twitter data or product reviews. Normalization of UGC has received great attention also because the performance of NLP tools (e.g. taggers, parsers and named entity recognizers) is greatly decreased when applied to it. Besides misspelled words, this kind of text presents a long list of problems, such as acronyms and proper names with inconsistent capitalization, abbreviations introduced by chat-speak style, slang terms mimicking the spoken language, loanwords from English as technical jargon, as well as problems related to ungrammatical language and lack of punctuation [12–15].

In [15] the authors propose a spell checker for Brazilian Portuguese (BP) to work on the top of Web text collectors, and tested their method on news portals and on informal texts collected from Twitter in BP. However, they do not inform the error correction rate of the system. Furthermore, while their focus is on the response time of the application, they do not address real-word errors.

Spell checking is a well-developed area of research within NLP, which is now crossing the boundaries between science and engineering, by introducing new sources of knowledge and methods which were successfully applied to different scenarios. This paper presents two new spell checking methods for UGC in BP. The first one deals with phonetically motivated errors, a recurrent problem in UCG not addressed by traditional spell checkers, and the second one deals additionally with real-word errors. We present a comparison of these methods with a baseline system and JaSpell over a new and large benchmark corpus for this task. The corpus is also a contribution of our study[1], containing product reviews with 38,128 tokens and 4,083 annotated errors.

This paper is structured as follows. In Section 2 we describe our methods, the setup of the experiments and the corpus we compiled. In Section 3 we present the results.

[1] The small benchmark of 120 tokens used in [16] and [17] is not representative of our scenario.

In Section 4 we discuss related work on spelling correction of phonetic and real-word errors. To conclude, the final remarks are outlined in Section 5.

2 Experimental Settings and Methods

In this Section we present the four methods compared in our evaluation. Two of them are used by existing spellers; one is taken as baseline and the other is taken as benchmark. The remaining two are novel methods developed within the project reported herein. After describing in detail the novel methods, we present the corpus specifically developed to evaluate BP spellers, as well as the evaluation metrics.

2.1 Method I - Baseline

We use as a baseline the open source Java Spelling Checking Package, JaSpell[2]. JaSpell can be considered a strong baseline and is employed at the tumba! Portuguese Web search engine to support interactive spelling checking of user queries. It classifies the candidates for correcting a misspelled word according to their frequency in a large corpus together with other heuristics, such as keyboard proximity or phonetic keys, provided by the Double Metaphone algorithm [18] for English. At the time this speller was developed there was no version of these rules for the Portuguese language[3].

2.2 Method II - Benchmark

The method presented in [19] is taken as benchmark. It combines phonetic knowledge in the form of a set of rules and the algorithm Soundex, and was inspired by the analysis of errors of the same corpus of products' reviews [20] that inspired our proposals.

Furthermore, as this method aims to normalize web texts, it performs automatic spelling correction. To increase the accuracy of the first hit, it relies on some ranking heuristics, which consider the phonetic proximity between the wrong input word and the candidates to replace it. If the typed word does not belong to the lexicon, a set of candidates if generated with words from the lexicon having an edit distance to the original word of one or two.

Then, a set of phonetic rules for Brazilian Portuguese codifies letters and digraphs which have similar sounds in a specific code. If necessary, the next step performs the algorithm Soundex, slightly modified for BP. Finally, if none of these algorithms is able to suggest a correction, the candidate with the highest frequency in a reference corpus among the ones with the least edition-distance is suggested. The lexicon used is the Unitex-PB[4] and the frequency list was taken from Corpus Brasileiro[5].

[2] http://jaspell.sourceforge.net/

[3] Currently, a BP version of the phonetic rules can be found at http://sourceforge.net/projects/metaphoneptbr/

[4] http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/

[5] http://corpusbrasileiro.pucsp.br/cb/

2.3 Method III - Grapheme-to-Phoneme based Method (GPM)

By testing the benchmark method, we noticed that many of the wrong corrections were related to a gap between the application of phonetic rules and the Soundex module. The letter-to-sound rules were developed specially for the spelling correction, therefore, they are very accurate for the task but have a low recall, since many words do not possess the misspelling patterns which they try to model. In contrast, the transcriptions generated by the adapted Soundex algorithm are too broad and many phonetically different words are given the same code. For instance, the words "perto" (*near*) and "forte" (*strong*) are both transcribed with the Soundex code "1630", in spite of being very distinct phonetically: "perto" corresponds to ['pɛx.tʊ], and "forte" to ['fɔx.tʃɪ].

To fill this gap, we propose the use of a general-purpose grapheme-to-phoneme converter to be executed prior to the Soundex module. We employed Aeiouado's grapheme-to-phoneme converter [21], since it is the state of the art in grapheme-to-phoneme transcription for Brazilian Portuguese.

The usage of the grapheme-to-phoneme converter is a bit different from a simple pipeline. According to Toutanova [22], phonetic-based errors usually need larger edit distances to be detected. For instance, the word "durex" (*sellotape*) and one of its misspelled forms "duréquis" have an edit distance of 5 units, despite having very similar phonetic forms: [du'rɛks] $\sim$ [du'rɛkɪs].

Therefore, instead of simply increasing the edit distance, which would imply in having a larger number of candidates to filter, we decided to do the reverse process. We transcribed the Unitex-PB dictionary and stored it into a database, with the transcriptions as keys. Thus, in order to obtain words which are phonetically, we transcribe the input and look it up in the database. Considering the "duréquis" example, we would first transcribe it as [du'rɛ.kɪs], and then check if there are any words in the database with that transcription. In this case, it would return "durex", the expected form.

The only difference of GPM in comparison with Method II lies in the G2P transcription match, which takes place prior to Soundex. In spite of being better than the baseline because they tackle phonetic-motivated errors, Method II and GPM have a limitation: they do not correct real word errors. The following method is intended to overcome this shortcoming by using context information.

2.4 Method IV – GPM in a Machine Learning framework (GPM-ML)

Method IV has the advantage of bringing together many approaches to spelling correction into a machine learning framework. Its architecture is described in Figure 1.

The method is based on three main steps: (i) candidate word generation, (ii) feature extraction and (iii) candidate selection. The first encompasses three modules which produce a large number of suggestions, considering orthographic, phonetic and diacritic similarities. For producing suggestions which are typographically similar, the Levenshtein distance is used: for each input word, we select all words in a dictionary which diverge by at most 2 units from the input. For instance, suppose the user intended to write "mesa" (*table*), but missed a keystroke and typed "meda" instead. The Levenshtein module would generate a number of suggestions including an edit distance of 1

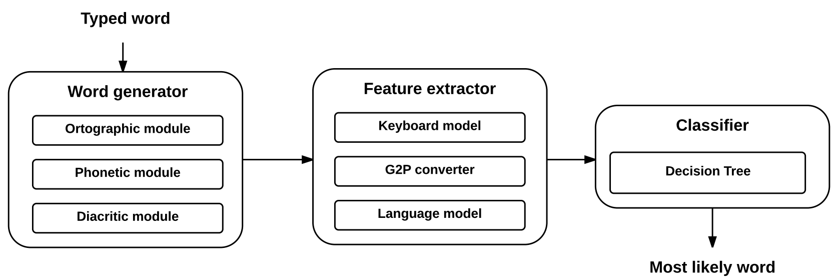

Fig. 1. *Architecture of the GPM-ML*

or 2, such as ”medo” (*fear*), ”meta” (*goal*), ”moda” (*fashion*), ”nada” (*nothing*), ”mexe” (*he/she moves*) etc.

For computational efficiency, we stored the dictionary in a trie structure. A revised version of the Unitex-PB was employed as our reference dictionary (*circa* 550,000 words)[6].

As for phonetic similarity, we used the same procedure as with GPM. Thus, in order to generate suggestions which are phonetically similar to the word typed by the user, we look up in the database.

The diacritic module generates words which differ from the input in terms of diacritic symbols. This module was proposed because we observed that most of the misspellings in the corpus were caused by a lack or misuse of diacritics. BP has five types of diacritics: accute (´), cedilla (ç), circumflex (ˆ), grave (`) and tilde (˜), and they often indicate different vowel quality, timbre or stress.

However, they are rarely used in UGC, and the reader uses the context to disambiguate the intended word. In order to deal with this problem, the diacritic model generates, given a input word, all possible word combinations of diacritics. Once more, the Unitex-PB is used as reference.

After candidate generation, the feature extraction phase takes place. The aim is to allow the classifier to compare these words with the one typed by the user, in such a way that the classifier can choose to keep the typed word or to replace it with one of generated suggestions.

As misspelling errors may be of different nature (such as typographical, phonological or related to diacritics), we try to select features that encompass all these phenomena. For each word suggestion produced in the word generation phase, we extract 14 features, as described in Table 1.

The probabilities come from a language model trained over a subset of the Corpus Brasileiro (*circa* 10 million tokens). Good-Turing smoothing is used to estimate the probability of unseen trigrams.

After feature extraction, the word selection phase comes into play. The classifier was implemented through scikit-learn [23] and comprises an optimized version of the

[6] The dictionary is available upon request.

Table 1. *List of features*

Feature	Description
1. TYPEDORGEN	whether the word was typed by the user or was produced in the word generation phase;
2. ISTYPO	1 if the word was generated by the typographical module; 0 otherwise;
3. ISPHONE	1 if the word was generated by the phonetic module; 0 otherwise;
4. ISDIAC	1 if the word was generated by the diacritic module; 0 otherwise;
5. TYPEDPROB	the unigram probability of the word typed;
6. GENUNIPROB	the unigram probability of the word suggestion;
7. TYPEDTRIPROB	the trigram probability of the word typed;
8. GENTRIPROB	the trigram probability of the word suggestion;
9. TYPOLEVDIST	the levenshtein distance between the typed word and the suggestion;
10. INSKEYDIST	the sum of the key insertion distances;
11. DELKEYDIST	the sum of the key deletion distances;
12. REPLKEYDIST	the sum of the key replacement distances;
13. KEYDISTS	the sum of all previous three types of key distances;
14. PHONELEVDIST	the levenshtein distance between of the phonetic transcription of the typed word and of the suggestion.

CART algorithm, trained over the dataset presented in Section 2.5, with the features we discussed, and evaluated through 5-fold cross-validation. Several other classification algorithms were tested, but since our features contain both nominal and numerical data, and some of them are dependent, the decision tree classifier achieved the best performance.

2.5 Dataset

The evaluation corpus was compiled specially for this research and is composed of a set of annotated product reviews, written by users on Buscapé[7], a Brazilian price comparison search engine. All misspelled words were marked, the correct expected form was suggested and the misspelling category was indicated.

Considering that the amount of errors is not very large, we used a non-probabilistic sampling technique similar to snowball sampling [24], in order to obtain a reasonable amount of data with incorrect orthography. A list of orthographic errors with frequency greater than 3 in the the corpus of product reviews compiled by [20] was used to preselect from the same corpus sentences with at least one incorrect word. Among those, 1,699 sentences were randomly selected to compose the corpus (38,128 tokens). All these sentences were annotated by two linguists with prior experience in corpus annotation. The inter-rater agreement for the error detection task is described in Table 2.

Table 2. *Inter-rater agreement for the error detection task*

		Annot. B		
		Correct	**Wrong**	**Total**
Annot. A	**Correct**	33,988	512	34,500
	Wrong	76	3,559	3,635
	Total	34,064	4,071	38,135

The agreement was evaluated by means of the kappa test [25]. The κ value for the error detection task was 0.915, which stands for good reliability or almost perfect agree-

[7] http://www.buscape.com.br/

ment [26]. The final version of the corpus used to evaluate all methods was achieved by submitting both annotations to an adjudication phase, in which all discrepancies were resolved. We noticed that most annotation problems consisted of whether or not to correct abbreviations, loanwords, proper nouns, internet slang, and technical jargon. In order to enrich the annotation and the evaluation procedure, we classified the misspellings into five categories:

1. TYPO: misspellings which encompass a typographical problem (character insertion, deletion, replacement or transposition), usually related to key adjacency or fast typing; e.g. ”obrsevei” instead of ”observei” (*I noticed*) and real-word errors such as ”clama” (*cries out*-PRES3.SG) instead of ”calma” (*calm*).
2. PHONO: cognitive misspellings produced by lack of understanding of letter-to-sound correspondences, e.g. ”esselente” for ”excelente” (*excelent*), since both ”ss” and ”xc”, in this context, sound like [s]. Real-word errors also occur, such as ”compra” (*buys*-PRES3.SG) instead of ”comprar” (*to buy*).
3. DIAC: this class identifies misspellings which are related to the inserting, deleting or replacing diacritics in a given word, e.g. ”organizacao” instead of ”organização” (*organization*) and real-word errors, such as ”sabia” (*knew*-IMPF.3SG) instead of ”sábia” (*wise*-FEM.SG).
4. INT_SLANG: use of internet slang or emoticons, such as ”vc” instead of ”você” (*you*), ”kkkkkk” (to indicate laughter) or ”:-)”.
5. OTHER: other types of errors that do not belong to any of the above classes, such as abbreviations, loanwords, proper nouns, technical jargon; e.g. ”aprox” for ”aproximadamente” (*approximately*).

The distribution of each of these categories of errors can be found in Table 3. Interestingly, Internet slang, which is usually assumed to be a core problem in UGC, was not a frequent error pattern, accounting for only 4.9% of the occurrences.

The difference between the total number of counts in Table 2 and 3 is caused by spurious forms which were reconsidered or removed in the adjudication phase. In addition to the five categories previously listed, we also classified the misspellings into either contextual or non-contextual, i.e. if the misspelled word corresponds to another existing word in the dictionary, it is considered a contextual error (or real-word error). For instance, if the intended word was “está” (*he/she/it is*), but the user typed “esta”, without the acute accent, it is classified as a contextual error, since “esta” is also a word in BP which means *this* FEM.

The corpus has been made publicly available[8] and intends to be a benchmark for future research in spelling correction for user generated content in BP.

2.6 Evaluation Metrics

Four performance measures are used to evaluate the spellers: *Detection rate* is the ratio between the number of detected errors and the total number of existing errors. The *Correction rate* stands for the ratio between the number of corrected errors and the total

[8] `https://github.com/gustavoauma/propor_2016_speller`

Table 3. *Error distribution in corpus by category*

Misspelling type		Counts	% Total
TYPO	-	1,027	25.2
PHONO	Contextual	49	1.2
	Non-contextual	683	16.7
DIAC	Contextual	411	10.1
	Non-contextual	1,626	39.8
INT_SLANG	-	201	4.9
OTHER	-	86	2.1
Total/Avg		4,083	100.0

number of errors. *False positive rate* is the ratio between the number of false positives (correct words that are wrongly treated as errors) and the total number of correct words. The *False negative rate* consists of the ratio between the number of false negatives (wrong words that are considered correct) and the total number of errors.

In addition, the correction hit rates are evaluated by misspelling categories. In the analysis, we do not take into account the ”int_slang” and ”other” categories, since both show a very irregular behavior and constitute specific types of spelling correction.

3 Discussion

In Table 4, we summarize all methods' results. As one can observe, the GPM-ML achieved the best overall performance, with the best results in at least three rates: detection, correction and false positive rate. Both methods we proposed in this paper, GPM and GPM-ML, performed better than the baseline in all metrics; however, GPM did not show any improvement in comparison with the benchmark.

In fact, the addition of the grapheme-to-phoneme converter decreased the correction rate. By analyzing the output of GPM, we noticed that there seems to be some overlapping information between the phonetic rules and the grapheme-to-phoneme module. Apparently, the phonetic rules were able to cover all cases which could be solved by adding the grapheme-to-phoneme converter, and therefore our hypothesis was not supported.

Table 4. *Comparison of the Methods*

Method	Rate			
	Detection	Correction	FP	FN
Baseline JaSpell	74.0%	44.7%	5.9%	26.0%
Benchmark Rules&Soundex	83.4%	68.6%	1.7%	**16.6%**
GPM	83.4%	68.2%	1.7%	**16.6%**
GPM-ML	**84.9%**	**78.1%**	**0.7%**	21.9%

All methods showed a low rate of false positives, and the best value was found in GPM-ML (0.7%). The false positive rate is very important for spelling correction purposes and is related to the reliability of the speller. In the following we discuss the correction hit rates by misspelling categories, presented in Table 5.

The baseline JaSpell (Method I) presented an average correction rate of 44.7%. Its best results comprise non-contextual diacritic misspellings with a rate of 64.0%;

Table 5. *Comparison of Correction Rates*

Misspelling type	Errors	Correction rate by method I	II	III	IV
Typo	1,027	28.3%	**56.3%**	53.0%	55.4%
Phono Contextual	49	0.0%	0.0%	0.0%	**8.1%**
Phono Non-contextual	683	48.2%	85.1%	**87.1%**	81.1%
Diac Contextual	411	9.2%	26.5%	26.5%	**64.5%**
Diac Non-contextual	1626	64.0%	82.2%	82.4%	**96.6%**
Total/Weighted Avg	3,796	44.7%	68.6%	68.1%	**78.0%**

worst results are found in contextual phonological errors, in which not a single case was corrected by the speller. Typographical misspellings were also very troublesome, with a correction hit rate of 28.3%. These results indicate that this method is not suitable for real world applications which deal with user generated content (it is important to notice that the JaSpell was not developed for this text domain).

The benchmark Rules&Soundex (Method II) achieved a correction rate of 68.6%, a relative gain of 53.4% in comparison to the baseline. The best results are, once more, related to the non-contextual diacritic misspellings (82.2%), the major error class. The best improvements compared to the baseline appear in phonological errors influenced by context (85.1%), with a relative increase of 76.6%. These results are coherent with the results reported by [19], since they claim that the method focuses on phonetically motivated misspellings.

As already mentioned, GPM (Method III) did not show any gain in comparison with the benchmark. As can be noticed, it had a small positive impact in what regards to the phonological errors, raising the correction rate of non-contextual phonological misspellings from 85.1% to 87.1% (2.3% gain).

GPM-ML (Method IV) achieved the best performance among all methods in what regards correction hit rate (78.0%). A very high correction rate was achieved in non-contextual diacritic errors (96.6%) and non-contextual phonological errors (81.1%). The trigram Language Model proved to be effective for capturing some contextual misspellings, as can be seen by the contextual diacritic correction rate (64.5%). However, the method was not able to properly infer contextual phonological misspellings (8.1%). We hypothesize this result might be caused by the small number of contextual phonological errors in the training corpus (there were only 49 cases of contextual phonological misspellings). No significant improvement was found with respect to typographical errors (55.4%) in comparison to the other methods.

4 Related Work

The first approaches to spelling correction date back to Damerau [27] and address the problem by analyzing the edit distances. He proposes a speller based on a reference dictionary and on an algorithm to check for out-of-vocabulary (OOV) words. The method assumes that words not found in the dictionary have at most one error (a threshold established to avoid high computational cost), which was caused by a letter insertion, deletion, substitution or transposition.

An improved error model for spelling correction, which works for letter sequences of lengths up to 5 and is also able to deal with phonetic errors was proposed by [28]. It embeds a noisy channel model for spell checking based on string to string edits. This model depends on the probabilistic modeling of sub-string transformations.

As texts present several kinds of misspellings, no single method will cover all of them, and therefore it is natural to combine methods which supplement each other. This approach was pursued by [22], who included pronunciation information to the model of typographical error correction. [22] and also [29] took pronunciation into account by using grapheme-to-phoneme converters. The latter proposed the use of triphone analysis to combine phonemic transcription with trigram analysis, since they performed better than either grapheme-to-phoneme conversion or trigram analysis alone.

Our GPM method also combines models to correct typographical errors by using information on edition distance, pronunciation, grapheme-to-phoneme conversion and the output of the Soundex method. In this two-layer method, these modules are put in sequence, as we take advantage of the high precision of the phonetic rules before trying the converter; typographical errors are corrected in the last pass of the process.

5 Final Remarks

We compared four spelling correction methods for UGC in BP, two of which consisting of novel approaches proposed in this paper. The Method III (GPM) consisted of an upscale version of the benchmark method, containing an additional module with a grapheme-to-phoneme converter. This converter was intended to provide the speller with transcriptions that were not so fine-grained or specific as those generated by the phonetic rules and also not so coarse-grained as those created by Soundex; however, it didn't work as well as expected.

The Machine Learning version of GPM, the GPM-ML, instead, presented a good overall performance, as it is the only one that addresses the problem of real word errors. It surpasses all other methods in most situations, reaching a correction rate of 78.0%, with very low false positive (0.7%) and false negative (21.9%), thus establishing the new state of the art in spelling correction for UGC in BP. As for future work, we intend to improve GPM-ML by expanding the training database, testing other language models as well as new phone conventions. In addition, we plan to more thoroughly evaluate it on different testing corpora. We also envisage, in due course, the development of an Internet slang module.

Acknowledgments

Part of the results presented in this paper were obtained through research activity in the project titled "Semantic Processing of Brazilian Portuguese Texts", sponsored by *Samsung Eletrônica da Amazônia Ltda.* under the terms of Brazilian federal law number 8.248/91.

References

1. Duan, H., Hsu, B.P.: Online Spelling Correction for Query Completion. In: Proceedings of the 20th International Conference on World Wide Web. WWW '11, NY, USA, ACM (2011) 117–126
2. Fossati, D., Di Eugenio, B.: A Mixed Trigrams Approach for Context Sensitive Spell Checking. In Gelbukh, A., ed.: Computational Linguistics and Intelligent Text Processing. Volume 4394 of Lecture Notes in Computer Science., Springer (2007) 623–633
3. Fossati, D., Di Eugenio, B.: I saw TREE trees in the park: How to Correct Real-Word Spelling Mistakes. In: Proceedings of the Sixth International Conference on Language Resources and Evaluation LREC 2008
4. Mays, E., Damerau, F.J., Mercer, R.L.: Context based spelling correction. Information Processing & Management **27**(5) (1991) 517–522
5. Wilcox-O'Hearn, A., Hirst, G., Budanitsky, A.: Real-word Spelling Correction with Trigrams: A Reconsideration of the Mays, Damerau, and Mercer Model. In: Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing. CICLing'08 (2008) 605–616
6. Islam, A., Inkpen, D.: Real-word spelling correction using Google web 1tn-gram data set. In: In ACM International Conference on Information and Knowledge Management CIKM 2009. (2009) 1689–1692
7. Sonmez, C., Ozgur, A.: A Graph-based Approach for Contextual Text Normalization. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing EMNLP 2014. (2014) 313 – 324
8. Hirst, G.: An evaluation of the contextual spelling checker of Microsoft Office Word 2007 (2008)
9. Németh, L.: Hunspell. Dostupno na: http://hunspell. sourceforge. net/[01.10. 2013] (2010)
10. Zampieri, M., Amorim, R.: Between Sound and Spelling: Combining Phonetics and Clustering Algorithms to Improve Target Word Recovery. In: Proceedings of the 9th International Conference on Natural Language Processing PolTAL 2014. (2014) 438–449
11. Rusell, R.C.: US Patent 1261167 issued 1918-04-02. (1918)
12. Duran, M., Avanço, L., Aluísio, S., Pardo, T., Nunes, M.G.V.: Some Issues on the Normalization of a Corpus of Products Reviews in Portuguese. In: Proceedings of the 9th Web as Corpus Workshop WaC-9, Gothenburg, Sweden (April 2014) 22–28
13. De Clercq, O., Schulz, S., Desmet, B., Lefever, E., Hoste, V.: Normalization of Dutch User-Generated Content. In: Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013. (2013) 179–188
14. Han, B., Cook, P., Baldwin, T.: Lexical Normalization for Social Media Text. ACM Trans. Intelligent System Technology **4**(1) (February 2013) 5:1–5:27
15. Andrade, G., Teixeira, F., Xavier, C., Oliveira, R., Rocha, L., Evsukoff, A.: HASCH: High Performance Automatic Spell Checker for Portuguese Texts from the Web. Proceedings of the International Conference on Computational Science **9**(0) (2012) 403 – 411
16. Martins, B., Silva, M.J.: Spelling Correction for Search Engine Queries. In: Proceedings of the 4th International Conference EsTAL 2004 – España for Natural Language Processing. Volume 3230 of Lecture Notes in Computer Science. (2004) 372–383
17. Ahmed, F., Luca, E.W.D., Nürnberger, A.: Revised N-Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness. Polibits (12 2009) 39–48
18. Philips, L.: The double metaphone search algorithm. C/C++ Users Journal **18**(6) (2000)
19. Avanço, L., Duran, M., Nunes, M.G.V.: Towards a Phonetic Brazilian Portuguese Spell Checker. In: Proceedings of ToRPorEsp Workshop PROPOR 2014, São Carlos, Brazil (2014) 24–31

20. Hartmann, N., Avanço, L., Balage, P., Duran, M., Nunes, M.G.V., Pardo, T., Aluísio, S.: A Large Corpus of Product Reviews in Portuguese: Tackling Out-Of-Vocabulary Words. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation LREC'14. (2014) 3866–3871
21. Mendonça, G., Aluísio, S.: Using a hybrid approach to build a pronunciation dictionary for Brazilian Portuguese. In: Proceedings of the 15th Annual Conference of the International Speech Communication Association INTERSPEECH 2014, Singapore (2014)
22. Toutanova, K., Moore, R.C.: Pronunciation Modeling for Improved Spelling Correction. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. ACL '02 (2002) 144–151
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830
24. Browne, K.: Snowball sampling: using social networks to research non-heterosexual women. International Journal of Social Research Methodology **8**(1) (2005) 47–60
25. Carletta, J.: Assessing Agreement on Classification Tasks: The Kappa Statistic. Computational Linguistics **22**(2) (June 1996) 249–254
26. Landis, J.R., Koch, G.G.: The Measurement of Observer Agreement for Categorical Data. Biometrics **33**(1) (1977) pp. 159–174
27. Damerau, F.J.: A Technique for Computer Detection and Correction of Spelling Errors. Communications of ACM **7**(3) (mar 1964) 171–176
28. Brill, E., Moore, R.C.: An Improved Error Model for Noisy Channel Spelling Correction. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. ACL '00 (2000) 286–293
29. van Berkel, B., Smedt, K.D.: Triphone Analysis: A Combined Method for the Correction of Orthographical and Typographical Errors. In: Proceedings of the Second Conference on Applied Natural Language Processing, Austin, Texas, USA (February 1988) 77–83