# I saw TREE trees in the park: How to correct real-word spelling mistakes

**Davide Fossati, Barbara Di Eugenio**

University of Illinois
Chicago, IL, USA
dfossa1@uic.edu, bdieugen@cs.uic.edu

**Abstract**

This paper presents a context sensitive spell checking system that uses mixed trigram models, and introduces a new empirically grounded method for building confusion sets. The proposed method has been implemented, tested, and evaluated in terms of coverage, precision, and recall. The results show that the method is effective.

## 1. Introduction

The detection and correction of spelling mistakes that result in real words of the target language, also known as *real-word spell checking*, is the most challenging task for a spell checking system. These errors, such as in "I come **form* Chicago" ("form" was typed when "from" was intended) or "I saw **tree* trees in the park" ("tree" instead of "three"), can only be discovered taking context into account.

One might think correcting spelling mistakes is a solved problem, since every word processor now includes a spell checker. However, the majority of those systems are not able to catch the kind of errors described above.

One might wonder if these errors occur often in practice. Indeed, empirical studies have estimated that errors resulting in valid words account from 25% to more than 50% of the errors, depending on the application (Mitton, 1987; Kukich, 1992). Thus, it appears that this challenging problem has not received the attention it deserves.

Recently, extensive work has been done on the characterization of spelling mistakes in web documents (Ringlstetter et al., 2006). This work shows that the amount of spelling mistakes in the web is impressive, providing further motivation for new research on spell checkers.

Different approaches to tackle the issue of real-word spell checking have been tried in the literature. Symbolic approaches (Heidorn et al., 1986) try to detect errors by parsing each sentence and checking for grammatical anomalies. More recently, some statistical methods have been tried, including the usage of word n-gram models (Mays et al., 1991; Berlinsky-Schine, 2004; Wilcox-O'Hearn et al., 2008), POS tagging (Marshall, 1983; Garside et al., 1987; Golding and Schabes, 1996), Bayesian classifiers (Gale et al., 1993), decision lists (Yarowsky, 1994), Bayesian hybrid methods (Golding, 1995), a combination of POS and Bayesian methods (Golding and Schabes, 1996), and Latent Semantic Analysis (Jones and Martin, 1997). Methods that exploit semantic similarity between words have also been proposed (Hirst and Budanitsky, 2005; Budanitsky and Hirst, 2006).

The spell checker of the recently released Microsoft Word 2007 is able to detect and correct some real-word mistakes (Microsoft Corporation, 2006). The proprietary, closed-source nature of this software makes it difficult to assess the technology and methods used in it. An initial, independent evaluation of that system can be found in (Hirst, 2008).

The main problem with word n-grams is data sparseness, even with a fairly large amount of training data. In fact, a study (Berlinsky-Schine, 2004) reported better performance using word bigrams rather than word trigrams, most likely because of the data sparseness problem. POS based methods suffer much less from the sparse data problem, but such approaches are unable to detect misspelled words resulting in words with the same part of speech. Bayesian methods, on the other hand, are better able to detect these cases, but have worse general performance. These last two methods give better results when combined together (Golding and Schabes, 1996).

A significant issue with many proposed systems is coverage. For example, the sophisticated machine learning method proposed in (Golding and Roth, 1996; Golding and Roth, 1999; Carlson et al., 2001) has been "scaled up" to cover approximately 500 words in the latter work. Although this is a significant improvement over the 20-40 words covered by previous research, the method is still far from covering the majority of the English language.

A slightly different application area in which statistical contextual spell checking has been studied is Optical Character Recognition (OCR). For this application, Markov Models approaches based on letter n-grams have been shown to be quite successful (Tong and Evans, 1996).

To tackle the problem of data sparseness of word trigrams models, and overcome the lack of information provided by POS trigrams models, a mixed trigram model has been proposed (Fossati and Di Eugenio, 2007). That study shows how words and parts of speech can be combined together in the computation of trigrams, and how the mixed trigrams can be used to detect, and possibly correct, dictionary words that are likely to be typos in the given context. An important limitation of the method proposed by (Fossati and Di Eugenio, 2007) is an oversimplified determination of confusion sets.

This paper presents a new system that uses a mixed trigram model, and introduces a new empirically grounded method for constructing the confusion sets of words. The resulting model is relatively compact, yet suitable for building a high-coverage real-word spell checking system.

## 2. Conceptual framework

A mixed trigram model can be represented by a Hidden Markov Model (HMM), where each state of the HMM is labeled either with a pair of POS tags or with a pair made

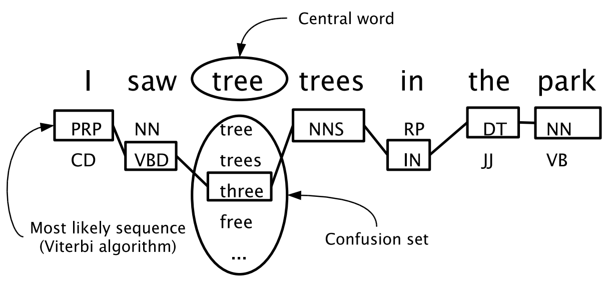

Figure 1: Example of detection process

of a POS tag and a valid dictionary word. The symbols emitted by the HMM's states are the words observed in the input sentences. The valid words that label some of the states represent words in the *confusion set* of the observed words. Given a sentence and a word to be checked (*central word*), the entire sentence is matched against the HMM. The POS-labeled states generate the words in the central word's context, whereas the states labeled with the confusion set generate the central word itself. The Viterbi algorithm is used to efficiently compute the most likely state sequence that matches the observed sentence. If the central word differs from the label of the state that emitted it, then a misspelling has likely occurred. In this case, the correct word is represented by the HMM's state, and the observed word is a misspelling of that word. Figure 1 shows a graphical representation of the detection process.

### 2.1. Central word probability estimation

The following formula represents the most likely sequence of labels associated to an input sentence, according to the model. $E$ is the sequence of labels (either POS tags or words), $e_i$ is the label associated to the word at position $i$ in the input sentence, and $w_i$ is the word at position $i$ in the sentence.

$$E = \text{argmax}_E \prod_{i=1}^{n} P(w_i|e_i)P(e_i|e_{i-1}e_{i-2})$$

If $i = k$ is the index of the central word, then the term $e_i = e_k$ is a word belonging to the confusion set of the observed word $w_i = w_k$, and $P(w_i|e_i) = P(w_k|e_k)$ represents the probability that a word $e_k$ gets transformed into $w_k$ because of a spelling mistake. The issue is how to estimate this probability. In previous work, we tried a function of edit distance between words, but that function had no empirical validation. Here, we want to estimate the probability $P(w_k|e_k)$ on a more empirically grounded basis. Unfortunately, lexical resources that can be used for such estimation are scarce. One of these resources is the Birkbeck spelling error corpus (Mitton, 1986), a collection of corpora of English spelling mistakes. Unfortunately, it is not possible to use this corpus to make direct estimations of misspelling frequencies, because frequency information is available only for a small subset of the corpus. For the same reason, the corpus cannot be used to make accurate inferences about the context in which spelling mistakes occur. However, this corpus provides a reasonably large set of word/misspelled word pairs. Being a collection, this corpus is highly heterogeneous in format and content. After preprocessing and cleaning, 31699 pairs have been extracted. Even though most of the mistakes represented in these pairs are non-words, and the main focus of this work is on real-word spelling mistakes, the collected pairs represent a useful source of information. For example, our database contains the pairs (*there*, **their*) and (*absence*, **absense*). The first pair represents a real-word mistake, whereas the second one is a "regular" non-word mistake. However, both cases show the same psycholinguistic source of error, i.e., homophony. So, we want to exploit this collection to try to make our system learn how a misspelling "looks like."
More formally, let us introduce a boolean variable $M$, where $M$ = true if a misspelling has occurred, and $M$ = false otherwise. To simplify the notation, we say that $P(M) := P(M = \text{true})$. The conditional probability $P(w_k|e_k)$ can be rewritten as follows:

$$P(w_k|e_k) = P(M|w_k, e_k)$$

This term still represents the probability of confounding a word $e_k$ with a different word $w_k$. The main idea is to approximate the joint probability distribution of $(w_k, e_k)$ with the distribution of a set of $n$ features, function of the words $w_k$ and $e_k$.

$$\begin{aligned} P(M|w_k, e_k) &\approx \\ &\approx P(M|F_1(w_k, e_k), \ldots, F_n(w_k, e_k)) = \\ &= \frac{P(F_1(w_k, e_k), \ldots, F_n(w_k, e_k)|M)P(M)}{P(F_1(w_k, e_k), \ldots, F_n(w_k, e_k))} \end{aligned}$$

The probability distribution $P(F_1(w_k, e_k), \ldots, F_n(w_k, e_k)|M)$ can be estimated from a corpus of word/misspelling pairs, like the one we derived from the Birkbeck collection. The distribution $P(F_1(w_k, e_k), \ldots, F_n(w_k, e_k))$ can be estimated in a similar way using the union of the previous pair set with the cross product of the vocabulary.
The set of features used in this work can be easily calculated from the words. They include the following functions:

$$\begin{aligned} F_1 &= \text{length}(e_k) \\ F_2 &= \text{length}(w_k) \\ F_3 &= \text{edit\_distance}(e_k, w_k) \\ F_4 &= \text{phonetic\_distance}(e_k, w_k) \end{aligned}$$

A common step usually taken to deal with joint probability distributions is to assume independence of the features and approximate the joint probability of the feature set with the product of the probabilities of the individual features. In this case, however, this looks like a dangerous step to take, since the features are intuitively somewhat correlated. To verify this, a correlational analysis was done on part of the data set, that indeed showed a relatively high degree of correlation. For example, on the spelling mistake corpus, the correlation coefficient between the length of the correct word and the edit distance between the correct and misspelled word is $0.29$ (the correlation coefficient of two random variables can range between $-1$ and $1$, where $0$

means that the two variables are uncorrelated, 1 means that they are completely directly correlated, and $-1$ means they are totally inversely correlated). Thus, it is better to evaluate the joint probability distribution directly with Maximum Likelihood Estimation. This is feasible, because the chosen features have a relatively small range of values.
The *phonetic distance* between two words is calculated by first converting each word into some kind of phonetic representation, then computing the edit distance between the two phonetic representations. There are several algorithms that can convert a string of characters into an approximate phonetic representation. Three of them are *soundex* (AA.VV., 2007b), *refined soundex* (AA.VV., 2007a), and *double metaphone* (Phillips, 2000). Double metaphone looks like the most interesting one for two reasons: first, it is the most recent one, and it has been designed to overcome some of the limitations of the previous ones. Second, the correlational analysis showed an interesting phenomenon: even though the correlation coefficient between edit distance and double metaphone is somewhat high (0.32) when calculated on the cross product of the dictionary words, it is surprisingly low (0.01) when calculated on the misspelling corpus. This fact suggests that the double metaphone distance might be a good feature to use together with the edit distance for this task.

### 2.2. Confusion sets

The method proposed above for the estimation of the lexical likelihood $P(w_k|e_k)$ can also be used to determine the *confusion set* of a word. In previous work, we constructed the confusion set of a word by taking all the words in the dictionary with edit distance less or equal than 2 to the original word. A big problem with this strategy is that the confusion sets of short words (words with 1 or 2 characters) were unmanageably large, so in practice short words were just skipped. The new feature-based scoring mechanism allows the construction of reasonably sized confusion sets for short words too. The procedure works as follows. For each word in the dictionary, all the other words are scored using the proposed features. If the confusion likelihood of the pair of words exceeds a threshold, the second word is inserted into the confusion set of the first one. The threshold is determined empirically. Notice that, according to the chosen features, confusion sets are not necessarily symmetrical, i.e., if a word $w_1$ is in the confusion set of $w_2$, the word $w_2$ is not necessarily in the confusion set of $w_1$.

### 2.3. False positive reduction

A crucial problem of many real-word spell checkers is the very high false positive rate. To reduce the false positive rate, the total probability values associated to the sequences discovered by the Viterbi algorithm can be used to build confidence values for the spell checker. The system will mark a possible typo only if the confidence value associated to it is high enough.

$$\text{confidence} = \log(P(S_C)) - \log(P(S_O))$$

In the above formula, $P(S_C)$ is the overall probability of the sentence where the central word has been replaced by the most likely correction, and $P(S_O)$ is the probability of the sentence where the central word has not been replaced. The confidence threshold value is determined empirically.

## 3. Corpora

**Training data.** This work relies on two different types of training data: a collection of misspelled pairs, already described in section 2.1, and a corpus of correct sentences annotated for parts of speech. In our previous work, we used the WSJ Penn Treebank (Marcus et al., 1993) to train the mixed trigram model. Experiments with three different sizes of the training data showed that the accuracy of the model's predictions did not increase significantly using more training data, but the overall coverage of the model was larger. Thus, in addition to the WSJ Penn Treebank, the current version of the system uses the Stanford POS tagger (Toutanova et al., 2003) to automatically tag plain text and use it as additional training data. The idea is that, under the assumption that the accuracy of the system does not decrease too much because of the mistakes of the tagger, this could be an effective way to increase the coverage of the spell checker even more. The following additional texts have been collected and used as additional training corpora: *The New American Bible* (8.5 Mb) (AA.VV., 2008b), *The Project Gutenberg Encyclopedia* (section 1, 8.1 Mb), and four novels by Mark Twain (2.1 Mb total) (AA.VV., 2008a).

**Test data.** Unfortunately, there is a serious lack of data available to the research community suitable for testing real-word spell checkers. Many researchers adopted the strategy of creating a test set artificially, by randomly replacing words in otherwise correct sentences. We used the same approach here. More details on our test set are presented in section 5.

## 4. Notes on implementation

**Programming language.** The system has been entirely implemented using Java, version 5.0.

**Smoothing.** The system performs discounting and smoothing of the least frequent trigrams and bigrams using the Good-Turing method. Lexical likelihoods (i.e., $P(w_i|e_i)$) are not smoothed. The least frequent unigrams (dictionary entries) are pruned. This leads to three additional parameters, again to be determined empirically.

**Number recognition.** Many tokens (about 3% of the total tokens) in the training data represent numbers. Trying to spell-check them is not a good idea, and in fact many mistakes made by the statistical spell checker involve this kind of tokens. Thus, the system symbolically identifies numbers using a regular expression, and treats them differently than regular words. During training, numbers are replaced with a special token. During the run-time spell checking, central words identified as numbers are skipped, and the other numbers in the context of regular words are replaced with the appropriate special token.

**Unknown words.** Statistics about unknown words are collected by taking advantage of the least frequent words pruned from the training corpus. Those words are mapped to a special token during training. Whenever an unknown word is found at spell checking runtime, it is also mapped to this special token if it appears in the context of the central word; if the central word is unknown, it is skipped.

## 5. Evaluation

The system has been empirically evaluated. In the results reported in Table 2, the main reference for comparison is the system presented in (Fossati and Di Eugenio, 2007), where the numbers have been made directly comparable.

### 5.1. Parameters

There are seven relevant parameters to be tuned in the system. Five of them have to be be fixed when training the model, whereas two can be varied at spell checking runtime. Table 1 shows the parameters and their meaning.
The values of these parameters have been chosen by experimentation over a validation set of 150 sentences, of the same type (but different instances) of the final test set. To determine sensible values for the *confidence* parameter, experiments have been run on the validation sentences, setting confidence to zero (maximum sensitivity). Then, average and standard deviation of the confidence associated to true positives (real errors detected by the system) and false positives (correct words erroneously detected as wrong by the system) have been calculated. The average confidence of true positives is about 3.1, whereas the average confidence of false positives is about 1.9. Unfortunately, statistically these values are not sufficiently different. In fact, the standard deviations are, respectively, about 2.2 and 1.6. This indicates that it is not possible to accurately separate true positives and false positives using the confidence in the way it is currently defined. So, it is not realistic to expect improvements in performance by varying this parameter.

### 5.2. Test set

Our test set, the same used for the evaluation of the previous version of our system, is a collection of 500 sentences in section 20 of the WSJ Penn Treebank, where one word (at least 3 character long) in each sentence has been randomly replaced with another word in the dictionary having edit distance less or equal than 2 to the original one. The choice of this test set allows for a direct comparison between the two systems. Also, this artificial creation of the test set is common in the literature, because of lack of available data. Notice though that artificial mistakes do not necessarily reflect the most frequent mistakes made by people.

### 5.3. Performance measures

Although the spell checking community has frequently used *hit rate* and *false positive rate*, these are not necessarily the best measures one might use. In fact, the usage of false positive rates (defined as the ratio of false positives over the total number of words checked) tends to overestimate the performance of a system. The reason is that this rate includes a large number of true negatives, which are usually not as interesting as are true positives, false positives, and false negatives. In this respect, the traditional IR measures of *precision*, *recall*, and *f-score* seem more informative of the real performance of the system. Notice that the definition of hit rate corresponds to that of recall. Another important measure is *coverage*, defined as the ratio between the words actually checked and the words skipped by the system.

### 5.4. Results

Table 2 reports the result of selected runs of the new system with different configuration of the parameters. The first row shows the best run of our previous system, where precision and coverage have been recalculated.
The results show an important increase in the coverage of the spell checker, mainly due to the fact that the system's limitation of skipping the words with less than three character has been removed, thanks to the new method for building the confusion sets. As expected, the automatically tagged additional training data had some positive effect on coverage as well, at the expense of some loss in precision and recall.
The precision of the system has noticeably increased, due to all the precautions taken to reduce the false positives. However, the recall has drastically decreased. This is partly due to the already mentioned bias in the test set. Overall, the f-score of the new system has increased. The runtime parameters have trading-off effects on precision and recall, but they don't seem to have big effects on the f-score.
Additional empirical experimentation on more realistic data, including tests of statistical significance, is left to future work.
It is difficult to directly compare the various results published in the literature, because of substantial differences in the data sets and evaluation metrics adopted by different researchers. For example, the experiments reported in (Berlinsky-Schine, 2004) scored a maximum of 92% detection hit rate and 30% false positive rate with a word bigrams model; 55% detection hit rate and 18% false positive rate with a word trigrams model. The results published in (Budanitsky and Hirst, 2006) report a maximum f-score of 0.14 for what the authors call "suspicion," and a maximum f-score of 0.25 for what they call "detection."

## 6. Conclusions and future work

This work showed that a more empirically grounded construction of confusion sets, together with careful implementation details, helps improve the performance of a context sensitive spell checker based on a mixed trigram model.
A major point still open is the tuning and evaluation on ecologically valid test data. In fact, a test set created artificially is not necessarily representative of the distribution of mistakes that real people make. Unfortunately, to the best of our knowledge, no ecologically valid data sets are available to the research community. We are investigating ways to inexpensively collect such data.
Another problem is that the test set we used overestimates the performance of the first version of the system, because each replaced word lies in at least one confusion set of that system (by definition), but this is not true with respect to the second system, which may result penalized. The reader should be aware of this fact when reading the results.
Experimentation with different features for the creation of confusion sets should be done. Again, having the possibility to inspect some ecologically valid data could point us to the selection of more effective features.

| Parameter | Type | Typical values | Meaning | Practical effect |
|---|---|---|---|---|
| Corpus size | Training | WSJ sections 00-19 (manually tagged), other text (automatically tagged) | This is the text used to determine the vocabulary and estimate bigrams and trigrams | A larger corpus tends to increase the coverage (less unknown words) |
| Vocabulary threshold | Training | 3 - 7 | This is the minimum number of occurrences in the training corpus for a word to be included in the vocabulary | A higher threshold reduces the size of the vocabulary |
| Bigrams smoothing threshold | Training | 5 - 15 | The frequency of those bigrams occurring a number of times less or equal than this threshold is discounted using the Good-Turing formula | With a higher threshold, more observed bigrams are discounted |
| Trigrams smoothing threshold | Training | 5 -15 | The frequency of those trigrams occurring a number of times less or equal than this threshold is discounted using the Good-Turing formula | With a higher threshold, more observed trigrams are discounted |
| Features threshold | Training | 0.001 - 0.1 | When computing the confusion sets, only the pairs of words with a probability (calculated according to the features) greater or equal than the threshold are included | Smaller values will produce larger confusion sets |
| Gamma | Runtime | 0.02 - 0.1 | It is P(M), the prior probability of making a spelling mistake | A higher value will cause more hits |
| Confidence | Runtime | Zero for the maximum sensitivity, 0 - 3 for a more tolerant system | It is the minimum value of confidence such that the system will notify a hit | A higher value will cause less hits |

Table 1: Relevant parameters of the system

| | | Training parameters | | | | Runtime parameters | | Performance | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| System | Corpus | Vocabulary threshold | Bigrams threshold | Trigrams threshold | Features threshold | Gamma | Confidence | Coverage | Precision | Recall | F-score |
| Old | wsj00-19 | 4 | N/A | N/A | N/A | N/A | N/A | 0.71 | 0.26 | 0.81 | 0.40 |
| New | wsj00-04 | 3 | 10 | 10 | 0.001 | 0.05 | 0.0 | 0.86 | 0.32 | 0.53 | 0.40 |
| | wsj00-04 | 3 | 10 | 10 | 0.001 | 0.05 | 0.5 | 0.86 | 0.34 | 0.46 | 0.39 |
| | wsj00-04 | 3 | 10 | 10 | 0.001 | 0.05 | 1.0 | 0.86 | 0.37 | 0.42 | 0.39 |
| | wsj00-04 | 3 | 10 | 10 | 0.001 | 0.10 | 0.0 | 0.86 | 0.28 | 0.61 | 0.38 |
| | wsj00-19 | 5 | 10 | 15 | 0.001 | 0.05 | 0.0 | 0.91 | 0.39 | 0.54 | 0.45 |
| | wsj00-19 | 5 | 10 | 15 | 0.001 | 0.05 | 0.5 | 0.91 | 0.43 | 0.49 | 0.46 |
| | wsj00-19 | 5 | 10 | 15 | 0.001 | 0.05 | 1.0 | 0.91 | 0.46 | 0.44 | 0.45 |
| | wsj00-19 | 7 | 10 | 10 | 0.001 | 0.05 | 0.0 | 0.90 | 0.40 | 0.54 | 0.46 |
| | wsj00-19 | 7 | 10 | 10 | 0.001 | 0.05 | 1.0 | 0.90 | 0.47 | 0.44 | 0.46 |
| | big | 7 | 10 | 15 | 0.001 | 0.05 | 0.0 | 0.92 | 0.39 | 0.49 | 0.44 |
| | big | 7 | 10 | 15 | 0.001 | 0.05 | 0.5 | 0.92 | 0.43 | 0.44 | 0.43 |
| | big | 7 | 10 | 15 | 0.001 | 0.05 | 1.0 | 0.92 | 0.47 | 0.38 | 0.42 |
| | big | 7 | 10 | 15 | 0.001 | 0.10 | 0.0 | 0.92 | 0.35 | 0.55 | 0.43 |
| | verybig | 7 | 10 | 10 | 0.001 | 0.05 | 0.0 | 0.93 | 0.42 | 0.46 | 0.44 |
| | verybig | 7 | 10 | 15 | 0.001 | 0.05 | 0.0 | 0.93 | 0.43 | 0.46 | 0.44 |
| | verybig | 7 | 10 | 15 | 0.001 | 0.05 | 1.0 | 0.93 | 0.48 | 0.35 | 0.41 |

| | |
|---|---|
| wsj00-04: | (2.2 Mb) WSJ Penn Treebank, sections 00-04 (manually tagged) |
| wsj00-19: | (8.8 Mb) WSJ Penn Treebank, sections 00-19 (manually tagged) |
| big: | (26 Mb) wsj00-19 + New American Bible (automatically tagged) + 4 Mark Twain's novels (automatically tagged) |
| verybig: | (39 Mb) big + The Project Gutemberg Encyclopedia, vol. 1 (automatically tagged) |

Table 2: Experimental results

Finally, the current system uses the Viterbi algorithm to find the most probable sequence of tags (in the sense explained in section 2). A problem revealed by the inspection of the experimental results is that, whenever the system makes a mistake, it is likely to make other mistakes in its surroundings. This is a known problem with the approach of finding the best sequence of tags. As an alternative, a local tag-by-tag maximization could be performed. This alternative approach is usually not adopted in classical part of speech tagging, because it generally leads to less likely sequences.

However, in this context, it might make sense to try it. The effectiveness of this alternate approach has to be assessed in further experiments.

## Acknowledgments

This work is partially supported by award N00014-07-1-0040 from the Office of Naval Research, and additionally by awards ALT-0536968 and IIS-0133123 from the National Science Foundation.

## 7. References

AA.VV. 2007a. Apache commons codec. http://commons.apache.org/codec/userguide.html.

AA.VV. 2007b. The soundex indexing system. http://www.archives.gov/genealogy/census/soundex.html.

AA.VV. 2008a. The Project Gutenberg website. http://www.gutenberg.org/.

AA.VV. 2008b. The United States Conference of Catholic Bishops website. http://www.usccb.org/.

Adam Berlinsky-Schine. 2004. Context-based detection of 'real word' typographical errors using markov models. Technical report, Cornell University, Ithaca, NY. http://adam.politikia.com/documents/typofinal.doc.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Andrew J. Carlson, Jeffrey Rosen, and Dan Roth. 2001. Scaling up context-sensitive text correction. In *AAAI*, pages 45–50.

Davide Fossati and Barbara Di Eugenio. 2007. A mixed trigrams approach for context sensitive spell checking. In *CICLing-2007, Eighth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, Mexico, February.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1993. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439.

Roger Garside, Geoffrey Leech, and Geoffrey Sampson. 1987. *The Computational Analysis of English: a corpus-based approach*. Longman.

Andrew Golding and Dan Roth. 1996. Applying winnow to context-sensitive spelling correction. In *International Conference on Machine Learning*, pages 182–190.

Andrew Golding and Dan 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.

Andrew Golding and Yves Schabes. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *34th Annual Meeting of the Association for Computational Linguistics*.

Andrew Golding. 1995. A bayesian hybrid method for context-sensitive spelling correction. In *The Third Workshop on Very Large Corpora*, pages 39–53.

G. E. Heidorn, K. Jensen, L. A. Miller, R. J Byrd, and M. S. Chodorow. 1986. The EPISTLE text-critiquing system. *IBM Systems Journal*, 21(3):305–326.

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11:87–111.

Graeme Hirst. 2008. An evaluation of the contextual spelling checker of microsoft office word 2007, January. http://ftp.cs.toronto.edu/pub/gh/Hirst-2008-Word.pdf.

Michael P. Jones and James H. Martin. 1997. Contextual spelling correction using latent semantic analysis. In *Fifth Conference on Applied Natural Language Processing*.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19.

Ian Marshall. 1983. Choice of grammatical word-class without global syntactic analysis: tagging words in the LOB corpus. *Computers and the Humanities*, 17:139–150.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 27(5):517–522.

Microsoft Corporation. 2006. Microsoft office word 2007 product guide. http://office.microsoft.com/en-us/word/HA101680221033.aspx.

Roger Mitton. 1986. A collection of computer-readable corpora of English spelling errors. Technical report, Birkbeck College, London, UK. Available online at the Oxford Text Archive website.

Roger Mitton. 1987. Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information processing and management*, 23(5):495–505.

Lawrence Phillips. 2000. The double metaphone search algorithm. *C/C++ Users Journal*, June. http://www.ddj.com/cpp/184401251.

Christoph Ringlstetter, Klaus U. Schulz, and Stoyan Mihov. 2006. Orthographic errors in web pages: Toward cleaner web corpora. *Computational Linguistics*, 32(3):295–340.

Xiang Tong and David A. Evans. 1996. A statistical approach to automatic ocr error correction in context. In *Fourth Workshop on Very Large Corpora*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL 2003*, pages 252–259.

L. Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2008. Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. In *CICLing-2008, 9th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 605–616, Haifa, Israel.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95.