A Mixed Trigrams Approach for Context Sensitive Spell Checking

Davide Fossati and Barbara Di Eugenio

Department of Computer Science
University of Illinois at Chicago
Chicago, IL, USA
dfossa1@uic.edu, bdieugen@cs.uic.edu

Abstract. This paper addresses the problem of real-word spell checking, i.e., the detection and correction of typos that result in real words of the target language. This paper proposes a methodology based on a mixed trigrams language model. The model has been implemented, trained, and tested with data from the Penn Treebank. The approach has been evaluated in terms of hit rate, false positive rate, and coverage. The experiments show promising results with respect to the hit rates of both detection and correction, even though the false positive rate is still high.

1 Introduction

Spell checking is the process of finding misspelled words in a written text, and possibly correct them. This problem has been widely studied, and spell checkers are probably among the first successful NLP applications widely used by the general public.

We can classify spelling errors in two main groups: *non-word errors* and *real-word errors*.

– *Non-word* errors are spelling errors that result in words that do not exist in the language. For example,
  * The *bok* is on the table.
  The word *bok* does not exist in English, and it probably derives from a typo of the noun *book*:
  The *book* is on the table.
– *Real-word* errors are errors that by chance end up to actual words. For example,
  * I saw *tree* trees in the park.
  The noun *tree* exists in English, but in this context it is most likely a typo of the numeral *three*:
  I saw *three* trees in the park.

According to Kukich, the problem of spell checking can be classified in three categories of increasing difficulty: non-word error detection, isolated-word error correction, and context-dependent word correction [1]. The real-word errors detection and correction task, focus of this paper, belongs to the third category.

Such errors are the most difficult to detect and correct, because they cannot be revealed just by a dictionary lookup, but can be discovered only taking context into account.

Different approaches to tackle the issue of real-word spell checking have been presented in the literature. Symbolic approaches [2] 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 [3, 4], POS tagging [5–7], Bayesian classifiers [8], decision lists [9], Bayesian hybrid methods [10], a combination of POS and Bayesian methods [7], and Latent Semantic Analysis [11].

The main problem with word n-grams is data sparseness, even with a fairly large amount of training data. In fact, a recent study [4] reported better performances using word bigrams rather than word trigrams, most likely because of the data sparseness problem. POS based methods suffer less of sparseness problems, but such approaches are unable to detect misspelled words that are of the same part of speech. Bayesian methods, on the other hand, are better able to detect this cases, but have worse general performance. These last two methods give better results when combined together [7].

A slightly different application area in which statistical contextual spell checking have been also studied is Optical Character Recognition (OCR). For this application, Markov Model based approaches using letter n-grams have been shown to be quite successful [12].

2 A Mixed Trigrams Approach

This paper proposes a statistical method based on a language model that is a combination of the word-trigrams model and the POS-trigrams model, called *mixed trigrams model*. The main linguistic motivation behind this model is to represent fine-grained lexical information at a local level, and summarize the context with syntactic categories. The main advantage of this model is a great reduction of the data sparsity problem. The following subsections formally define the model and the method to apply it to the spell checking problem.

2.1 Mixed Trigrams

Given a sentence, a *mixed trigram* is a sequence of three elements (e_i, e_{i+1}, e_{i+2}), where a generic element e_k is either the k-th word of the sentence or its part of speech. The particular type of mixed trigrams used in this work has the additional property that at most one of the elements can be a word. For example, consider the sentence:

The/DET kids/NOUN eat/VERB fresh/ADJ apples/NOUN

A complete set of mixed trigrams deriving from this sentence is the following:

(The, NOUN, VERB)
(NOUN, VERB, ADJ)
(VERB, ADJ, NOUN)
(DET, kids, NOUN)
(kids, VERB, ADJ)
(DET, NOUN, eat)
(NOUN, eat, ADJ)
(eat, ADJ, NOUN)
(DET, NOUN, VERB)
(NOUN, VERB, fresh)
(VERB, fresh, NOUN)

Additional trigrams can be considered by adding two "start of sentence" special words and two "end of sentence" special words, to capture the distinctions between words at the beginning, in the middle, or at the end of a sentence.

2.2 Confusion Sets

Another key definition is that of *confusion set*.

> Given a dictionary W, a distance function d defining a metric on W, and a word $w \in W$, a confusion set $C(w) \subseteq W$ is a set of words such that $w_c \in C \Leftrightarrow |d(w, w_c)| \leq k$, where k is a constant.

In practice, the confusion set of a word contains all the words "similar enough" to that word. The main issue is then to define a reasonable distance function.

2.3 Levensthein Distance

The *Levensthein distance*, also known as *minimum edit distance* [13], is the minimum number of editing operations necessary to transform one word into another. An editing operation is either a character insertion, deletion, or substitution. The rationale behind the adoption of such a measure is that it reflects quite well some common typing mistakes, like pressing a key twice, typing two keys instead of one, skipping a key, and typing a key instead of another. Such mistakes lead to words with Levensthein distance of 1 from the original word. Another common mistake is switching two characters (e.g., typing *form* instead of *from*). In this case, the wrong word and the correct word will have a Levensthein distance of 2. These kind of mistakes have been found to represent the vast majority of typing errors [14–16]. So, it sounds reasonable to include in the confusion set of a word all the words with Levensthein distance less or equal than 2 from the original word.

2.4 Method

Consider the following problem.

Given a sentence $S = w_1 \dots w_k \dots w_n$, find the most likely sequence of elements $E = t_1 \dots w_k^c \dots t_n$, where:

- $w_i,\ 1 \leq i \leq n$ are words;
- w_k is the *central word* (i.e., the word to be checked);
- $t_i,\ 1 \leq i \leq n,\ i \neq k$ are part of speech tags;
- w_k^c is a word belonging to the confusion set of the central word w_k (ideally, it should be the correct word).

The observed word w_k is likely to be a spelling mistake of w_k^c if:

1. $w_k \neq w_k^c$, and
2. the probability of the sequence E is smaller than the probability of another sequence $\bar{E} = \bar{t}_1 \dots \bar{w}_k^c \dots \bar{t}_n$ such that $\bar{w}_k^c = w_k$. In other words, the sequence $\bar{E}$ is the most likely sequence of elements where the central word w_k is assumed to be correct. In practice, it is calculated by "forcing" the confusion set of the central word to contain only one element, equal to the central word itself.

The criterion above means that a word will be detected as a spelling mistake if another word in the confusion set has higher likelihood of fitting into the same context. In particular, the word in the confusion set belonging to the sequence with the highest probability will be selected by the correction algorithm. This maximization problem can be solved using the Markov Model approach traditionally applied to POS tagging, adopting the same simplifying assumptions, and using mixed trigrams instead of word trigrams. The resulting formula is:

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

In the previous formula, the variables w_i are words, and the variables e_i are either words or POS tags. The Viterbi algorithm [17] can be used to efficiently compute the sequence E. Figure 1 provides an intuitive example of how the detection process works.

2.5 Conditional Probability Estimation for the Central Word

In the previous formula, for $i = k$ (i.e., the index of the central word), the term $P(w_i|e_i) = P(w_k|w_k^c)$ means the probability of getting the word w_k from a misspelling of the word w_k^c. This probability cannot be easily estimated from a corpus. To estimate a value for this probability term, we define it as a function of the distance between the two words, such that the probability is lower if the distance between the words is greater. In other words, the assumption is that it is more likely to get a similar word, rather than a very different word, as result of a spelling mistake.

$$P(w_k|w_k^c) = \frac{\alpha(1-\alpha)^{d(w_k^c, w_k)}}{\text{count}(\lambda \in W : d(\lambda, w_k) = d(w_k^c, w_k))}$$

The value of the parameter α should be tuned with empirical investigation.

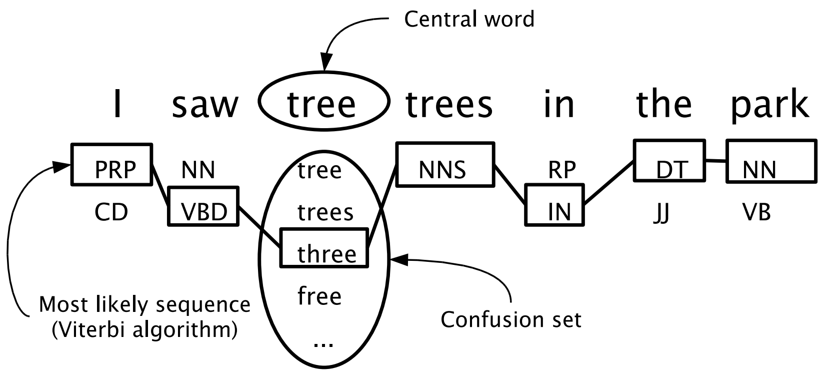

Fig. 1. Example of detection process

3 Experimental Settings

The method described in the previous section has been implemented and empirically evaluated. The following subsections present some details of the experimental settings.

3.1 Algorithm Training

Training the algorithm requires (a) creating a vocabulary, (b) gathering the probability estimations of the mixed trigrams and the conditional probabilities of each word given a POS, (c) calculating the distances among the words in the vocabulary, (d) calculating the conditional probability of each word given another word in its confusion set.

The vocabulary was created from the same corpus used to estimate trigrams and word-POS probabilities. Capitalization was removed, and the least frequent words (tokens with less than four occurrences) were not inserted in the vocabulary and their occurrences in the corpus were replaced with a special symbol for "unknown" words. The reasons are (1) to keep the dictionary and the probabilities table small, and (2) to gather good probability estimations to deal with real unknown words. In fact, even though the spell checking algorithm will check only the words present into the dictionary, the unknown words will be part of the surrounding context, so they have to be managed as well.

Trigrams and word-POS probabilities were estimated from portions of the WSJ section of the Penn Treebank corpus. Probabilities have been computed using the Maximum Likelihood Estimator (MLE) formula. The zero-probabilities were smoothed assigning a very small constant value to them. This smoothing method, however, introduces distortions in the probability space. The impact, hopefully small, of this distortion to the performance of the algorithm should be evaluated and corrected in future work; most likely, a more sophisticated smoothing technique should be used.

Three training data sets of increasing size have been tried. Table 1 shows some statistics about them. The number of tokens pruned by discarding the least frequent words from the vocabulary is noticeable; the effect was expected, and followed Zipf's law.

Table 1. Training corpus chunks used in the experiments

	Sections	Sentences	Words	Tokens	Vocabulary	Trigrams
Small	00-04	9992	274927	19225	5680	232758
Medium	00-09	20386	559315	27976	8857	403993
Large	00-19	41827	1139587	40689	13508	682868

The Levensthein distance measure was calculated for each pair of words in the vocabulary. This step is necessary for the determination of the confusion set of each word, that is computed at run-time looking for the words with Levensthein distance less or equal than 2 to the considered word.

The final training step is the calculation of the conditional probabilities that relate a word with its confusion set. To do that, the formula defined in the previous section was used, with three different values of α (0.25, 0.50, 0.75) for each of the three training sets. The total number of experimental settings was 9.

3.2 Test Data

A testing set of 500 sentences was collected from section 20 of the WSJ Penn Treebank. These sentences were randomly selected among the sentences with number of words between 10 and 30[1]. For each sentence in the test set, one spelling mistake was artificially inserted, by replacing a random word (among those words longer than two characters) with a word in its confusion set. This artificial insertion of spelling mistakes makes the test set ecologically invalid. However, this choice was considered appropriate for this first experimental stage, because choosing the test set in this way gives each "spelling mistake" a chance to be detected. In other words, the detection upper bound would be 100%, which makes it easier to interpret the final results.

3.3 Performance Measures

The following performance measures are considered relevant.

– *Detection hit rate.* It is the ratio between the number of typos detected and the total number of typos. The higher, the better.

[1] Since for each sentence in the WSJ corpus two special "start of sentence" tokens and two "end of sentence" tokens were added, the net number of "real" words in the test sentences actually ranges from 6 to 24 words.

- *Correction hit rate.* It is the ratio between the number of typos corrected and the total number of typos. This ratio is usually lower than the detection hit rate, because a spelling mistake can be rightly detected but corrected in a wrong way.
- *False positive rate on checked words.* It is the ratio between the number of false positives (i.e., correct words that are wrongly detected as typos) and the number of words checked by the algorithm. The lower, the better.
- *False positive rate on total words.* It is the ratio between the number of false positives and the total number of words in the tested sentences. Since not all the words are checked by the algorithm (see below), this ratio is usually lower than the false positive rate on checked words.
- *checked over unknown words ratio.* It is the ratio between the number of words checked and the number of words skipped by the algorithm because not in the vocabulary.
- *checked over short words ratio.* It is the ratio between the number of words checked and the number of words skipped because shorter than three characters. The reason why the words with less than three characters are skipped is that the confusion sets of short words were unmanageably high. In order to be able to check short words too, further research is needed in order to figure out how to prune the confusion set.
- *checked over skipped words ratio.* It is the ratio between the number of words checked and the total number of words skipped (unknown + short).
- *detected over false positive ratio.* It is the ratio between the number of detected typos and the number of false positives.

4 Experimental Results

4.1 Hit and False Positive Rates

Figure 2, Figure 3, and Figure 4 show the results of detection and correction hit rates (%), false positive rate on checked and total words (%), and the ratio between detected typos and false positives for the 9 possible combinations of the parameters (training corpus size and α value).

To put those numbers in perspective, the only results found in the literature that are somewhat comparable are those reported in [4]. Those experiments 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.

One might expect that the performance of the system would improve as the size of the training corpus increases, because a larger training corpus usually leads to a better estimation of the model's probability tables. In fact, Figure 3 shows a trend of reduction of the false positive rate as the corpus size gets larger. However, the results on hit rate do not display a similar positive trend. Also, both hit rate and false alarm rate show very little sensitivity with respect to the α parameter. This fact is interesting and unexpected.

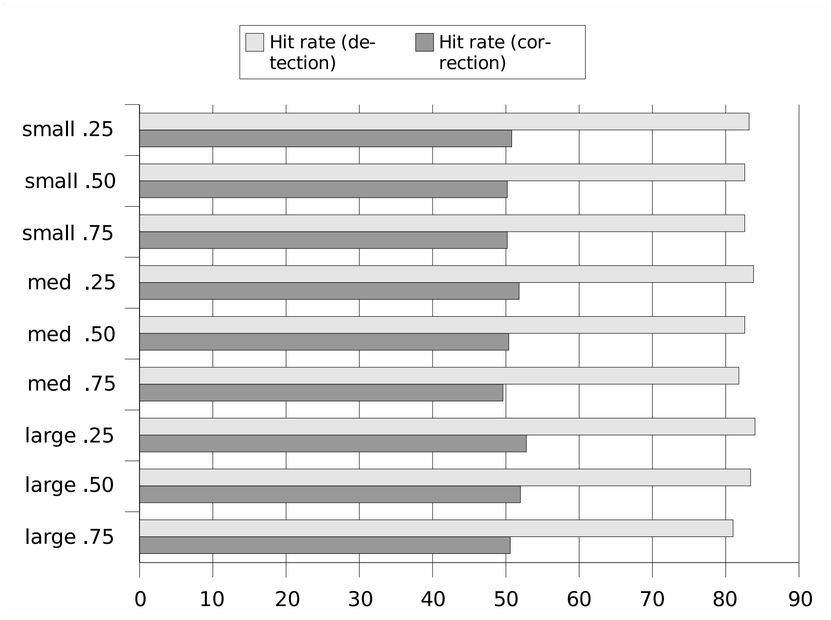

Fig. 2. Hit rates

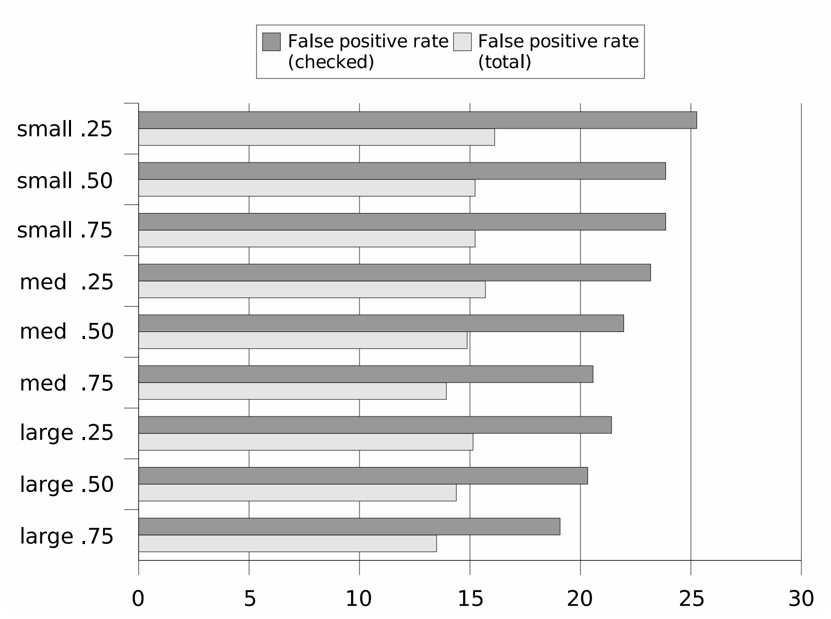

Fig. 3. False positive rates

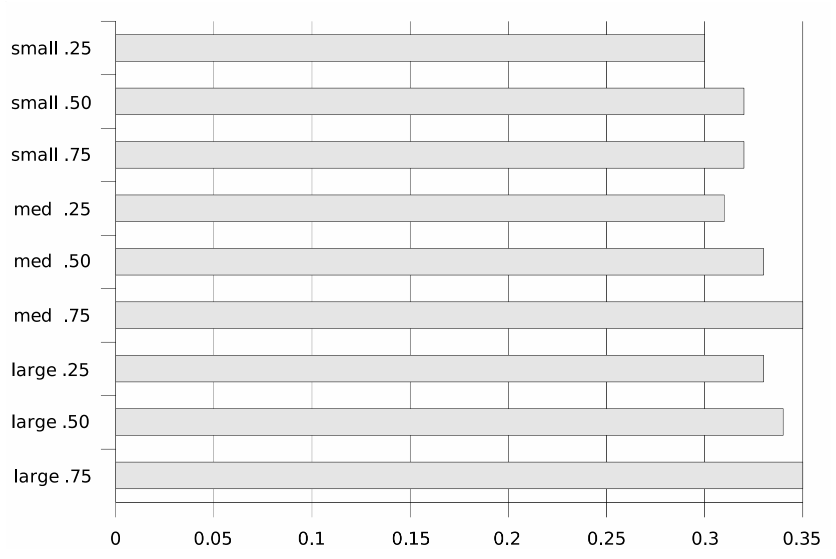

Fig. 4. Detected over false positive ratio

4.2 Coverage

Figure 5 shows the results for the coverage measures, i.e., checked over unknown, checked over short, and checked over skipped ratios. Those measures are independent on the values of α, as they depend only on the training corpus.

The coverage of the unknown words increases noticeably with the corpus size, because with a larger corpus more words are added to the vocabulary. However, the overall coverage is still depressed by the high amount of short words, that are skipped anyway.

5 Conclusions and Future Work

The results of these experiments are promising, and represent a good starting point for future research. Among the others, there are several points that would be worthy of further investigation:

- Improve the estimation of the probability tables, for example by using a more sophisticated smoothing technique.
- Deal with the problem of short words, that represent an important percentage of words that are now skipped.
- Explore the sensitivity of the algorithm to several other parameters, such as the threshold value to prune the vocabulary, and the length of the context to take into account.
- Run experiments on ecologically valid data.

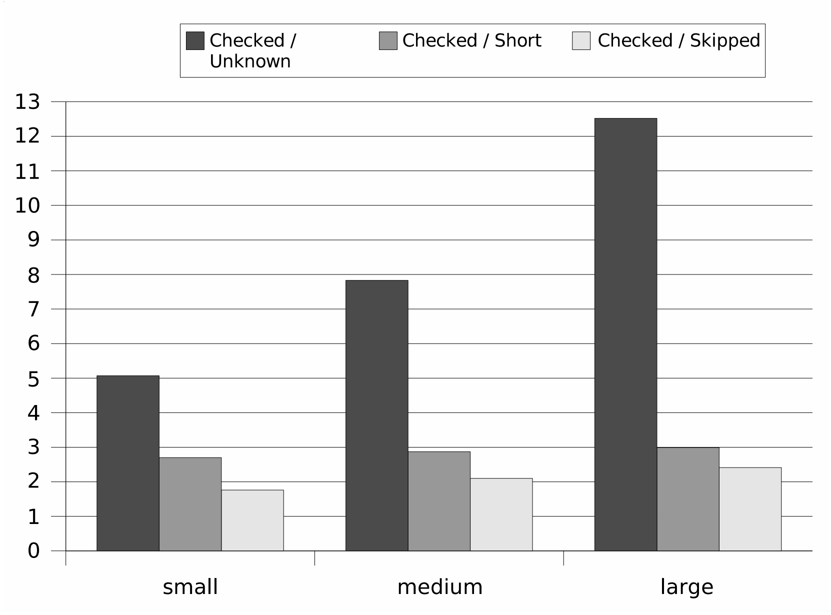

Fig. 5. Coverage of the spell checker

– Explore the usage of alternative word distance measures and conditional probabilities estimations for words with respect to their confusion set.

Acknowledgments

This work is supported by awards N00014-00-1-0640 from the Office of Naval Research, and awards IIS-0133123 and ALT-0536968 from the National Science Foundation. The authors thank Rajen Subba and Joel Booth for useful discussions.

References

1. Kukich, K.: Techniques for automatically correcting words in text. ACM Computing Surveys **24(4)** (1992) 377–439
2. Heidorn, G.E., Jensen, K., Miller, L.A., Byrd, R.J., Chodorow, M.S.: The EPISTLE text-critiquing system. IBM Systems Journal **21(3)** (1986) 305–326
3. Mays, E., Damerau, F.J., Mercer, R.L.: Context based spelling correction. Information Processing and Management **27(5)** (1991) 517–522
4. Berlinsky-Schine, A.: Context-based detection of real word typographical errors using markov models. Technical report, Cornell University, Ithaca, NY (2004) http://www.people.cornell.edu/ pages/arb36/typofinal.doc.

5. Marshall, I.: Choice of grammatical word-class without global syntactic analysis: tagging words in the lob corpus. Computers and the Humanities **17** (1983) 139–150
6. Garside, R., Leech, G., Sampson, G.: The Computational Analysis of English: a corpus-based approach. Longman (1987)
7. Golding, A., Schabes, Y.: Combining trigram-based and feature-based methods for context-sensitive spelling correction. In: 34th Annual Meeting of the Association for Computational Linguistics. (1996) http://acl.ldc.upenn.edu.
8. Gale, W.A., Church, K.W., Yarowsky, D.: A method for disambiguating word senses in a large corpus. Computers and the Humanities **26** (1993) 415–439
9. Yarowsky, D.: 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. (1994) 88–95
10. Golding, A.: A bayesian hybrid method for context-sensitive spelling correction. In: Proceedings of the Third Workshop on Very Large Corpora. (1995) 39–53
11. Jones, M.P., Martin, J.H.: Contextual spelling correction using latent semantic analysis. In: Fifth Conference on Applied Natural Language Processing. (1997) http://acl.ldc.upenn.edu.
12. Tong, X., Evans, D.A.: A statistical approach to automatic ocr error correction in context. In: Fourth Workshop on Very Large Corpora. (1996) http://acl.ldc.upenn.edu.
13. Jurafsky, D., Martin, J.H.: Speech and Language Processing. Prentice Hall (2000)
14. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Communications of the A.C.M. **7** (1964) 171–176
15. Pollock, J.J., Zamora, A.: Automatic spelling correction in scientific and scholarly text. Communications of the A.C.M. **27(4)** (1984) 358–368
16. Mitton, R.: Spellchecking by computer. Journal of the Simplified Spelling Society **20(1)** (1996) 4–11 http://www.les.aston.ac.uk/simplspel.html.
17. Manning, C.D., Schutze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)