Memory-Based Context-Sensitive Spelling Correction at Web Scale

Andrew Carlson
Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
acarlson@cs.cmu.edu

Ian Fette
Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
icf@cs.cmu.edu

Abstract

We study the problem of correcting spelling mistakes in text using memory-based learning techniques and a very large database of token n-gram occurrences in web text as training data. Our approach uses the context in which an error appears to select the most likely candidate from words which might have been intended in its place. Using a novel correction algorithm and a massive database of training data, we demonstrate higher accuracy on correcting real-word errors than previous work, and very high accuracy at a new task of ranking corrections to non-word errors given by a standard spelling correction package.

1 Introduction

In this paper, we apply memory-based learning techniques to the problem of correcting spelling mistakes in text, using a very large database of token n-gram occurrences in web text as training data. Our approach uses the context in which a word appears to select the most likely candidate from words which might have been intended in its place, as determined by the sequence of words most frequently observed in the n-gram training data. Our novel correction algorithm extends previous work by considering each possible position where the word in question can occur in an n-gram and selecting the most discriminative position.

We evaluate the performance of our algorithm on both *real-word errors* and *non-word errors*. Real-word errors are lexically valid words that are not noticed by conventional dictionary-based spelling correction packages when used incorrectly (e.g. using *piece* instead of *peace*). We exceed the accuracy of previous memory-based approaches to this task for both newswire and general English text. Non-word errors are errors where a word not found in a dictionary was used instead of an intended dictionary word. We show that our algorithm can accurately recommend corrections for non-word errors by reranking a list of spelling suggestions from a widely used spell checking package. The correct word is usually selected by our algorithm.

We view this application and our results as useful in their own right, but we also as a case study of applying relatively simple machine learning techniques that leverage statistics drawn from a very large corpus. The data set we use as training data is the Google n-gram data [3], which was released by Google Research in 2006. It contains statistics drawn from over a trillion words of web text. It was released with the aim of enabling researchers to explore web-scale applications using relatively modest computing resources. We believe that our application is one of many possibilities enabled by this data set.

The rest of the paper is organized as follows. Section 2 discusses related work, while Section 3 explains the details of the dataset we use in our work. Section 4 describes our methodology, while Section 5 presents results of applying that methodology on various datasets. Finally, we present our concluding remarks in Section 6.

2 Related Work

Approaches to finding real-word errors have frequently taken as input a list of *confusion sets*, where a confusion set is a set of dictionary words that are often incorrectly used in place of one another. Typical types of confusion sets include homophones (e.g. *piece* and *peace*), words with low edit distance (e.g. *form* and *from*), and words like *among* and *between* where the correct word is determined by the rules of grammar. When correcting spelling errors in an input text, whenever a member of a confusion set occurs in the text, the problem is to decide if a different member of that confusion set was intended in its place.

Banko and Brill [2] use a memory-based learning approach to select the correct member of a confusion set by collecting n-gram statistics from training data and choosing the member most frequently observed in the context of the word before and word after the target word. They show a roughly log-linear improvement in accuracy relative to the size of the training corpus. Their largest corpus has 10^9 words, three orders of magnitude smaller than the Google n-gram corpus. We report higher accuracy than their results, in line with their observed log-linear trend in improvement.

Liu and Curran [8] build a 10 billion word corpus from a web crawl and use the memory-based learner approach of Banko and Brill. They also give results using the Gigaword corpus, a 2 billion word corpus of relatively "clean" text from newswire and similar sources. They achieve the highest reported accuracies for true cross-domain experiments where the training data is not drawn from the same distribution as the test data. Our methods use much more training data than these previous memory-based methods, and a more sophisticated algorithm which chooses the most discriminative context to select the correct word, to exceed the previously stated best performance.

Golding and Roth explore a classification-based approach to this problem [6]. Given text that demonstrates correct usage of the words, classifiers are trained to discriminate the intended member of a confusion set from the context words near a member of that confusion set, their part-of-speech tags, and conjunctions of those features. Their work applies mistake-based classification algorithms to this problem, which can be relatively expensive to train and require large amounts of memory for the large feature spaces used. Later work explores methods of pruning features to avoid some of these problems [4]. Our work uses simpler methods with much more training data, and training our models consists of simply counting n-gram statistics in a corpus. Additionally, their approaches tend to perform poorly on true cross-domain experiments, where the training data differs significantly in domain from the test data. We train on a general Web corpus and evaluate on two domains, giving a more realistic evaluation of our system.

Church and Gale present an approach to correcting non-word errors using a noisy channel model to predict a probable intended word given a misspelled word [5]. Their use of contextual information from n-gram statistics showed a small improvement in accuracy. They found that simple MLE-based approaches to using the n-gram statistics actually degraded performance. We find that a simple MLE-based algorithm for estimating probabilities of intended words from their context works very well. Their work used much less training data (about 100 million words versus our 1 trillion), which explains why our simpler approach works well.

An alternative method of using web-scale data for natural language tasks is to use search engine counts, which removes the need to store a large database of aggregated data. Lapata and Keller used this approach with results from the search engine Altavista on the task of spelling correction [7]. However, Liu and Curran showed that this approach gives much lower accuracy than using accurate counts from a collection of web pages. This is because hit counts returned by search engines are typically estimates, and not well-suited for use in NLP systems.

3 Data Set

Our experiments use the Google n-gram data set [3], which was released in 2006, and is available through the Linguistic Data Consortium. The data collection gives the number of times an ordered sequence of tokens occurs in a collection of public web pages containing over one trillion words. Sequences of lengths one, two, three, four, and five are reported. Words that occurred fewer than 200 times were replaced by the token '<UNK>', and n-grams that occurred fewer than 40 times are not reported. Punctuation and capitalization are preserved in the data set. Punctuation is generally split into a single token, with the exception of apostrophes, which start a new token but letters after them are kept intact (e.g. *they're* becomes *they 're*).

This dataset is 87GiB on disk, without any database indexes. This may limit uses, as the data is far too large to distribute with client applications. However, the dataset is still useable in an online scenario where clients are contacting centralized servers, as the query load is light and the queries themselves can be executed quickly. To enable fast queries for our application, we loaded the data set into a MySQL database and indexed each table, which brought the total size of the database up to roughly 250GiB.

4 Methods

Our method for context-sensitive spelling correction chooses the most likely word from a list of candidates, given the context of adjacent words from the sentence in which the word in question appears. This requires us to determine which sequence of words (the context, with each of the candidates substituted in turn) is most probable. To estimate these probabilities we use the Google n-gram corpus previously described. Our procedure is described generally in Section 4.1, and specifically for correcting real-word errors and non-word errors in Sections 4.2 and 4.3.

4.1 General Methodology

Our general algorithm takes as input a set of candidate words $C = \{c_1, c_2, ..., c_k\}$, a database of n-gram statistics for sizes 1 to m, and context words less than m words to the left and right of the target word being corrected, where

w_{-i} is the ith word before, and w_j is the jth word after. For example, C could be $\{peace, piece\}$, m could be 2, and the context words could be w_{-1} = *a* and w_1 = *of*. Our algorithm recommends a word $c \in C$ using the procedure:

Consider each position in which each candidate word c could appear within an n-gram of size m. Select the position with the highest ratio between the highest count and the second highest count, and return the word c that corresponds to the highest count. If all queries had counts of zero, decrement the maximum n-gram size and repeat the procedure. In our example, we would consider the 2-grams *a peace, a piece, peace of,* and *piece of*. If no matches were found any of those 2-grams, we would then consider the 1-grams *peace* and *piece* and select the more frequent word.

Probabilistically, this algorithm selects the candidate with highest ratio between its joint probability with the context and the second highest probability, where we use the maximum likelihood estimate of the joint probability of a word in context from an n-gram count database.

Only n-grams that occurred at least 40 times were reported in the training data, so if we do not find a match for a query, we optimistically assume a count of 39.

In some cases, we do not have enough context words. We use only context from within the same sentence, so when the candidates appear near the beginning or the end of a sentence, we might not have a full set of context words. In this case, we restrict the positions in which a word can appear to those with sufficient context.

It should also be noted that the actual query issued to the database might not always directly line up with the context words. This is caused by tokenization issues, namely the fact that a word like “they’re” becomes two separate tokens- “they” and “’re” to match the tokenization used in the Google n-gram database. In this case, one of the tokens from the context will be ignored, as the word with the apostrophe takes up two words in the n-gram query.

4.2 Real-Word Error Methodology

One application of this methodology is to correct real-word errors where a real word was used while a different word was intended. Sets of such words are called *confusion sets*, and include sets of words such as $\{they're, their, there\}$, and $\{hear, here\}$. The application of our method to this problem is straightforward. Given some text to correct, whenever we find a word that is a member of a (pre-defined) confusion set, we use the method described in Section 4.1 to choose the correct word from the confusion set given its context.

In this setting, the left and right context are the sets of words appearing before and after the word that belongs to a confusion set, and the set of candidate words are those words that belong to the same confusion set. E.g. if we were presented with “Johnny is going to their house for dinner tonight”, the left context would be “Johnny is going to”, the right context would be “house for dinner tonight”, and the set of candidate words would be $\{their, they're, there\}$.

We tested this method on the Brown corpus, which is a collection of American English drawn from a variety of sources, and the Wall Street Journal corpus, a collection of articles from the Wall Street Journal. Both corpora are subsets of the text from the Penn Treebank corpus. Since these are widely used corpora, we were concerned that their text might occur enough on the web to bias the results, as the n-gram data came from web pages. However, we tested many queries and found only one complete copy of the Brown corpus and a few excerpts from each corpus online. We believe that the threshold of 40 occurrences used in the n-gram data is enough to prevent these pages from biasing the results. For each corpus, we looked for any occurrence of a word in a confusion set in the text. We assumed that the text was grammatical, meaning that the word used in the sentence was the correct member of the confusion set, and tested whether our method predicted the same member of the confusion set as being correct. This same methodology was used on the same corpus in the results reported by Liu and Curran [8], which we compare to in Section 5.1.

4.3 Non-Word Error Methodology

A second application of this method is towards re-ranking suggestions made by a spell checking program when a word not found in the dictionary is accidentally used instead of a dictionary word. Spell checkers are often used to correct such errors, but it’s not always clear what word is the correct replacement for a mis-typed word. For instance, should someone type “funn”, should this be interpreted as “fun”, “funny”, “funk”, or something else? To handle this, spell checkers often present users with a list of choices of correctly spelled words from which to choose. This list is typically generated based on edit distance from the non-word typed by the user without take context into account. Our goal is to choose the correct word from this list presented by a spell checker using the context in which the error occurs.

To test this approach, we took a novel (Fame and Fortune, by Horatio Alger) that had not previously been indexed by Google, and as such our results would not be biased by the text showing up in search engines. An e-book of this novel was released by Project Gutenberg on May 28, 2007, well after the release date of the n-gram dataset. Searching Google for a number of phrases from the book returned zero results. Our goal then was to introduce errors into the text, and see if we could use our method, in conjunction with a spell checker, to correct these errors.

We introduced three types of errors into the text - we added characters, deleted characters, and subsitituted characters. Each of these errors was introduced into a word with probability 0.05. In the case of added characters, we chose a number of characters by sampling from a Normal(0,.3) and taking the ceiling of its absolute value as the number of chracters to insert. For each of these characters, we chose a position for insertion (defined as the position before which to insert a letter, including the "end" position). We then inserted a character that was "close" to the following character. We did this by creating a distance mapping from each character to each other character, where the distance roughly approximates the L-1 distance on a standard U.S. keyboard (roughly L-1 in the sense that we used an exception for keys that are only one diagonal space apart, which we said have a distance of 1). We then choose a "close" key by choosing a distance by sampling a Normal(0,4) distribution, and again taking the ceiling of its absolute value, and then selecting one of the keys with this distance randomly.

We also deleted characters from words. The number of deletions was calculated in the same manner as the number of additions, and the specific characters to be deleted were selected uniformly at random. Finally, we introduced substitution errors into the text, where again the number of substitutions was calculated in the same way as the number of additions was calculated, the specific positions where errors were to be introduced were selected uniformly at random, and the character to substitute was chosen according to the same "closeness" metric as described in addition of characters.

For each word into which we introduced an error, we used GNU Aspell [1], an open source spell checking library, to generate candidates for spelling corrections. If the correct word did not appear within the first ten candidates, we ignored the instance and moved on to the next. (This often occurred when too severe an error, or combination of errors, was introduced.) We then used our method to determine which of the candidates was the correct suggestion. The left and right context were the (un-perturbed) words surrounding the word in which the error was introduced, and the candidate set was the set of words suggested by Aspell. We calculated both the rank of the correct word in the list returned by Aspell, as well as the rank produced by our method. If our method returned another word as most likely, we removed that word from the candidate set, and repeated the method, and continued this process until the correct word was suggested. The number of times this process was repeated thereby creates a ranking based on frequency of the words given the context.

This method of introducing errors created 3,373 misspelled words where the correct word was present among the top 10 suggestions. 943 of these words were the result of a character deletion only, 1,410 were the result of a character insertion only, 938 were the result of a character substitution only, and the rest were the result of multiple types of errors.

5 Results

5.1 Real-Word Error Correction

Our experiments for real-word error correction, where we choose the member of a confusion set that best fits a given context, aim to answer the following questions: Does using a very large corpus improve accuracy, and does considering each possible position where the target word can occur to select the most discriminative position help?

In the experiments that follow, "N-grams (limit 3)" gives results for training on the Google n-gram data with the same algorithm as Banko and Brill [2] (considering the 3-gram centered on the target word). "N-grams (fixed)" gives results for using 5-grams centered on the target word. "N-grams (sliding)" gives results for the full algorithm, where we consider all possible positions in the 5-gram window for the target word. Comparing to results from Liu and Curran [8], "Gigaword" gives results for the Banko and Brill algorithm on a 2 billion word corpus of newswire text, and "Web corpus" gives results from 10 billion words worth of web page text.

"Average accuracy" reports the accuracy averaged over the 18 confusion sets used in the experiments. "Weighted average accuracy" is the per-token accuracy, where confusion set accuracies are weighted proportional to their frequency in the test corpus.

Table 1. Comparison to previous results for correcting real-word spelling errors on Brown and WSJ corpora. Highest accuracies are in boldface.

Training Data	Test Corpus	Average Accuracy	Weighted Average Accuracy
Gigaword	Brown	90.7	94.6
Web corpus	Brown	91.8	95.4
N-grams (limit 3)	Brown	92.6	95.6
N-grams (fixed)	Brown	94.2	96.3
N-grams (sliding)	Brown	**95.2**	**96.8**
Gigaword	WSJ	93.7	**96.1**
Web corpus	WSJ	93.3	95.1
N-grams (limit 3)	WSJ	93.4	95.4
N-grams (fixed)	WSJ	94.3	95.9
N-grams (sliding)	WSJ	**95.8**	**96.7**

Table 1 compares the accuracy of our method to previous results. Our full method achieves higher accuracy on both the Brown and Wall Street Journal corpora than previous work using memory-based learners. The other rows in the table give results reported by Liu and Curran on the same test data with the same confusion sets. Our accuracy using 3-grams exceeds their results for the Brown corpus, and does not quite match on the Gigaword/WSJ combination. However, the WSJ and Gigaword corpora are both from the newswire domain, which probably helps performance significantly. These results show that using the larger corpus does indeed improve accuracy.

The previous work, when tested on the WSJ data after having been trained on the "web corpus" falls behind the fixed 5-gram method. When we add in the sliding window approach, where we no longer require that the n-grams be centered on the word in question, but rather use the positioning that is most discriminative, our accuracy increases to the point where we have the highest average accuracy on the Brown corpus by a 2.7% margin, and our average accuracy on the WSJ corpus exceeds previous work by 1.3-1.7% (depending on training set). We conclude that using larger n-grams improves performance significantly. Considering all possible positions for the target word appears to be an additional source of improvement.

Table 2 gives percentage accuracy for spelling correction on the Brown corpus on a per-confusion set level. The Confusion Set Average is the average accuracy across the confusion sets, and the Weighted Average is the average accuracy on a per-token level. We see that some confusion sets have lower accuracy than others– it is hard to distinguish between *among* and *between*, and relatively easier to distinguish between *affect* and *effect*. This seems sensible because *affect* and *effect* are likely to have different types of words nearby in the text, since *affect* is a verb and *effect* is usually a noun. *Among* and *between* share the same part of speech, and the local context in the text might not be informative about which is correct.

5.2 Non-Word Error Correction

Figure 1 shows the accuracy of our method for reranking suggestions from a spell checker for words where errors have been introduced by inserting, subsituting, or deleting characters. Insertion errors are the easiest to recover from. Aspell ranks the correct word first 43.1% of the time, while our method using 1-grams and 5-grams ranks the correct word first 79.1% and 92.4% of the time, respectively. For deletion errors, Aspell ranks the correct word first 33.1% of the time, compared to 60.1% and 84.9% for our method with 1-grams and 5-grams. Accuracy on substitution errors is similar, with Aspell ranking the correct word first 30.9% of the time, and our method 67.4% and 86.4% of the time for 1- and 5-grams, respectively.

Table 2. Test accuracies by confusion set for correcting real-word errors on the Brown corpus for various n-gram types.

Confusion Set	Examples	N-gram types used: 3-gram (fixed)	5-gram (fixed)	5-gram (sliding)
accept, except	253	96.4	97.2	97.2
affect, effect	249	97.6	97.6	98.8
among, between	1099	82.9	86.1	87.4
amount, number	643	78.2	87.6	89.9
begin, being	805	97.0	97.8	97.8
cite, sight, site	160	73.8	82.5	88.1
country, county	500	92.8	93.0	94.0
its, it's	2158	95.4	95.8	96.3
lead, led	264	87.1	90.2	88.6
fewer, less	478	94.6	94.6	94.4
maybe, may be	592	97.5	97.8	98.0
I, me	6347	98.9	99.0	99.0
passed, past	442	93.4	93.9	95.2
peace, piece	331	91.2	92.7	95.5
principal, principle	201	89.1	93.0	95.0
quiet, quite	358	96.4	96.1	98.3
raise, rise	155	94.2	94.2	95.5
than, then	3175	95.5	95.7	97.3
their, there, they're	5576	97.5	98.0	97.2
weather, whether	361	97.0	97.2	98.1
your, you're	1072	97.2	97.4	98.6
Confusion Set Average		92.6	94.2	95.2
Weighted Average		95.6	96.3	96.8

The accuracy using word frequency alone (1-grams) is better than what Aspell produces, and the addition of context yields an improvement in accuracy of roughly 10-25%, depending on error type. However, as Figure 1 also shows, the improvement between 3-grams and 5-grams is much smaller. It may be worth using only 3-grams to save space, as they offer the best tradeoff between space and accuracy.

Because spelling correction systems return a ranked list of suggestions, we also evaluate how often the correct word is ranked among the top k words. Figure 2 shows the "recall at k" for Aspell, 1-grams, and 5-grams, for k from 1 to 10 (recall that we discarded instances when the correct word did not appear in the top 10 results, hence all methods converge to 100% at k=10). Here, we see that the method using 5-grams usually has the target word in the top 2 or 3 ranked slots. The reranking using 1-grams is a significant improvement over Aspell, but often has candidates ranked fourth or fifth. The ranking given by Aspell is much worse than our methods, with a significant number of examples ranked in the lower half of the recommendations.

Figure 1. Results of reranking spell checker suggestions for non-word errors per error type.

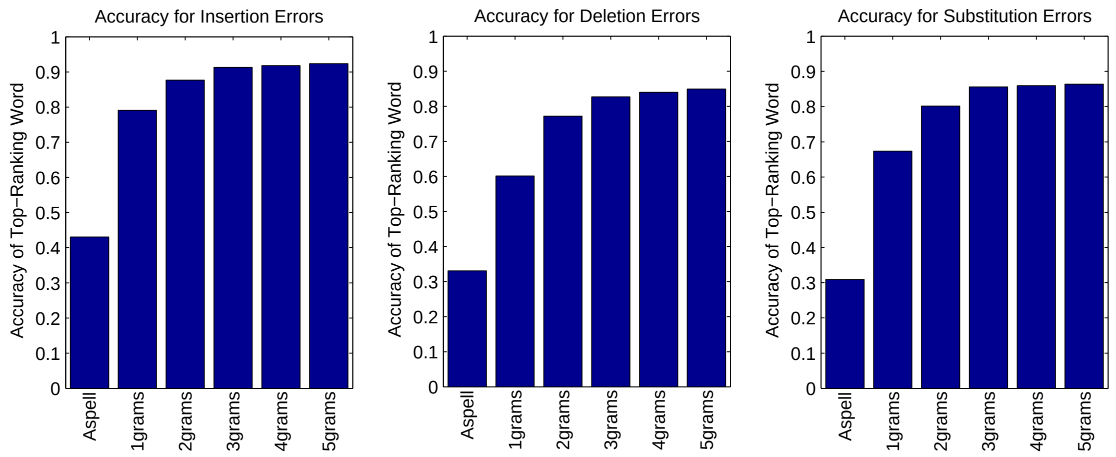
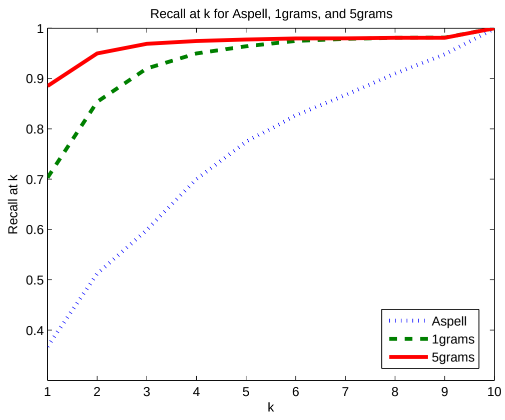


Figure 2. Recall at k for reranking the top 10 recommendations made by GNU Aspell for non-word errors using 1-grams and 5-grams.



6 Conclusions

We have shown that using a massive database for spelling correction with memory-based learning techniques yields high accuracy at correcting both real-word errors and non-word errors. Additionally, we extended previous approaches by considering each possible position where the spelling error could occur inside the n-grams and demonstrated that it improved accuracy.

Acknowledgements The authors would like to acknowledge the support of their advisors, Tom Mitchell and Norman Sadeh. Work reported herein has been supported in part under the NSF Cyber Trust initiative (Grant #0524189), under ARO research grant DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems") to Carnegie Mellon University's CyLab, by DARPA under the CALO/PAL program, and by a Yahoo! Graduate Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] K. Atkinson. *GNU Aspell*, 1998. Software available at `http://aspell.net/`.

[2] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

[3] T. Brants and A. Franz. Web 1t 5-gram version 1, 2006.

[4] A. J. Carlson, J. Rosen, and D. Roth. Scaling up context-sensitive text correction. In *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference*, pages 45–50. AAAI Press, 2001.

[5] K. W. Church and W. A. Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1991.

[6] A. R. Golding and D. Roth. Applying winnow to context-sensitive spelling correction. In *International Conference on Machine Learning*, pages 182–190, 1996.

[7] M. Lapata and F. Keller. Web-based models for natural language processing. *ACM Trans. Speech Lang. Process.*, 2(1):3, 2005.

[8] V. Liu and J. R. Curran. Web text corpus for natural language processing. In *EACL*. The Association for Computer Linguistics, 2006.