Probabilistic Detection of Context-Sensitive Spelling Errors

Johnny Bigert

Department of Numerical Analysis and Computer Science
Royal Institute of Technology, Stockholm, Sweden
johnny@kth.se

Abstract

This article focuses on the evaluation of a novel algorithm for the detection of context-sensitive spelling errors. We present a fully automatic evaluation procedure with no requirements of manual work or resources annotated with spelling errors. The evaluation method is applicable to any language and tag set, and is easily adaptable to other NLP systems such as taggers and parsers.

1. Introduction

Algorithms for the detection of misspelled words have been known since the early days of computer science. The program would simply look up a word in a dictionary, and if not present there, it was probably misspelled. Unfortunately, not all misspelled words result in an unknown word. Misspelled words resulting in existing words are called context-sensitive spelling errors, since a context is required to detect an error. Clearly, these errors are much more difficult than normal spelling errors since they require at least a basic analysis of the surrounding text.

Several approaches have been proposed to address context-sensitive spelling errors. To detect commonly confused words (e.g. *there, they're, their*), methods using confusion sets have been proposed (Golding and Roth, 1996). These use a limited set of errors, either manually constructed or obtained automatically. Context-sensitive spelling errors due to words outside the confusion set will not be processed. The algorithm proposed here is able to process and detect any misspelled word.

Other methods use statistical information, such as part-of-speech (PoS) trigram frequencies (Golding and Schabes, 1996) or transition probabilities and error likelihoods from PoS taggers (Atwell, 1987). These suffer from the problem of sparse data. Since the size of the corpus is limited, we will always face previously unseen grammatical constructions. The proposed method mitigates the effect of sparse data by preprocessing the corpus and extracting extra information on PoS tags.

Full parsing would be the ideal solution to detect context-sensitive spelling errors. The words that do not fit into the grammar are misplaced. To achieve reasonable accuracy for a full parser, extensive amounts of manual work is required. Furthermore, the processing of the text will be difficult if there are several errors in the same region since the parser will have little or no context to base its analysis upon. The method proposed here requires much less manual work and is extremely robust to multiple errors.

The main contribution of this article is a new approach to fully automated evaluation procedures. Furthermore, we provide some refinements of a novel detection algorithm for context-sensitive spelling errors (Bigert and Knutsson, 2002). Nevertheless, the focus here is the evaluation procedure.

2. Detection of Spelling Errors

This section describes the algorithm detecting context-sensitive spelling errors. The algorithm is divided into two parts: a statistical part and a transformation part. For additional details, we refer to the discussion concerning a preliminary version of the algorithm (Bigert and Knutsson, 2002). Nevertheless, a few improvements and simplifications have been made (e.g. the inclusion of different measures) and those are described here.

2.1. Statistical Information

The first, statistical part of the algorithm uses PoS tag n-gram frequency information, gathered from a corpus of the target language. To simplify the exposition of the algorithm, we will use PoS tag trigrams in this discussion, although the discussion is applicable to PoS tag n-grams of any size.

The general observation is that a grammatical construction is probably malformed if it contains previously unseen trigrams. Unfortunately, human language is very productive, and new, unseen grammatical constructs will arise. To address this problem, we broaden the concept of a PoS tag trigram.

Two PoS tags that are used in similar syntactic contexts are said to be close. We want to use this closeness, or *distance*, between tags to mitigate the effect of rare trigrams due to the productivity of the language.

We calculate the distance between two tags by using the frequencies of PoS tag trigrams obtained from the corpus. Given two tags t and r, we look up the frequencies for the trigrams (t_1, t, t_2) and (t_1, r, t_2). Naturally, if either t or r is more frequent than the other, the trigram frequencies will be higher and thus, we have to compensate for the tag frequencies. We obtain $P_t(t_1, t_2) = \mathit{freq}(t_1, t, t_2)/\mathit{freq}(t)$.

From this, we can apply a number of similarity measures from the work of Lee (1999), e.g. the L1 norm: $L1(P_t, P_r) = \sum_{t_1,t_2} |P_t(t_1, t_2) - P_r(t_1, t_2)|$, where the sum is over all tag pairs t_1, t_2 in the tag set. We see that the distance increases when the trigrams differ in frequencies and that the maximum distance 2 is obtained when the uses of the tags are disjoint. The minimum distance 0 is obtained when the PoS tag contexts are identical, which occurs when $t = r$.

The measure will give us a list of similarities between every pair of tags t and r. Suitably normalized to values in

$[0, 1]$, where 1 represents the closest distance (e.g. the distance from a tag to itself), we can use the values as probabilities. Thus, if p is $L1(P_t, P_r)$ normalized (i.e. the distance between t and r), p can be seen as the probability of retaining grammaticality when replacing a word having PoS tag t with a word having PoS tag r.

Now, given a rare trigram (t_1, t, t_2), we attempt to replace one (or more) of the tags with another tag close in distance. For example, if replacing t with r, we obtain (t_1, r, t_2). To penalize the tag change, we multiply the frequency of the new trigram with the probability p of the tag change. Now, if the penalized frequency is high (as defined by an arbitrary threshold e), the grammatical construction is most probably correct and the low frequency was originally due to a rare tag. If all attempted tag replacements result in low frequencies, the trigram is probably not grammatical and is marked as an error.

2.2. Phrase Transformations

Most false alarms occur near the beginning or end of a phrase constituent. There, a trigram covers two phrases and the productivity of the language gives rise to almost any combination of PoS tags. Furthermore, rare phrase constructions often produce rare PoS tag trigrams. Normally, the simplest (or shortest) form of a phrase is the most common, e.g. *the men* is a more common type of NP than *the little green men*. Thus, when faced with a potential error as described in the previous subsection, we will identify all adjacent phrases and try to simplify. Hopefully, the rare trigram is due to a rare combination of phrases.

We attempt to transform a phrase to a simpler form by replacing it with a more common phrase of the same type. Since we try to retain the inflectional information of the phrase, the new sentence will most probably be grammatical. For example: an error is detected near the words *are old are* in *All paintings that are old are for sale*. The NP *all paintings that are old* is reduced to *the paintings* and the sentence becomes *The paintings are for sale*, avoiding the rare construction.

The sentence resulting from the phrase transformation is fed to the algorithm in the previous section. If the new sentence is not erroneous, it is probably grammatically correct. If all phrase transformations fail (are reported as errors), there is probably a grammatical error and this is reported to the user. Rare PoS tag trigrams also occur frequently near a clause beginning or end. We decided not to look for errors in trigrams that cross a clause boundary. Hence, the largest unit is not a sentence but a clause.

3. Automatic Evaluation

An important objective in the design of an evaluation of a complex system is to minimize the amount of manual work. Due to the many parameters of the proposed algorithm, we required a fully automatic evaluation process as close as possible to the situation in which the algorithm is normally used. Clearly, we could produce or use an already existing resource with annotated spelling errors. To produce such a resource would be time-consuming and error-prone. Furthermore, vast amounts of data would be necessary to evaluate the many parameters.

The evaluation procedure described here is fully automated and requires no resources annotated with errors. Furthermore, it is portable to any language and tag set (given a dictionary in that language) and produces reproducible evaluations. The procedure can easily be adapted to evaluate other NLP systems, such as robustness in parsers and taggers (Bigert et al., 2003b) or the performance of spelling and grammar checkers.

3.1. Missplel and AutoEval

The normal use of the algorithm is a human writer producing text containing context-sensitive spelling errors. Hence, we wanted to simulate this process.

To produce spelling errors closely resembling those of a human writer, we used a freeware application called MISSPLEL (Bigert et al., 2003a). MISSPLEL was configured to produce keyboard mistype errors resulting in an existing word with a change in PoS tag. For example, *to be or not to be* could be misspelled *to be or not to me*. This results in a PoS tag change from verb to pronoun and clearly, a context-sensitive spelling error difficult to detect.

The results were gathered using another application called AUTOEVAL (Bigert et al., 2003a), devised to simplify the design of evaluations. AUTOEVAL handled input, output, data gathering and processing through a simple ten line script.

3.2. Evaluation Method

The parser used here was GTA (Knutsson et al., 2003), a rule-based shallow parser for Swedish. GTA also identified the clause boundaries. The parsing accuracy of GTA is about 88.7% and 88.3% for the clause identification. We used 15 000 words of written Swedish from the SUC corpus (Ejerhed et al., 1992).

Using MISSPLEL, we introduced errors randomly in 1%, 2%, 5%, 10% and 20% of the words. To minimize the influence of chance, we repeated the process 10 times for each error level, resulting in 50 misspelled texts of 15 000 words each.

Since the algorithm is divided into two parts, statistics and transformations, we wanted to assess the individual influence of each part. Thus, each part was turned either on or off, resulting in four different settings. If the statistics part was turned off, we simply considered a trigram ungrammatical if its frequency was below a predetermined threshold e. By turning off both statistics and transformations, we obtained a simple trigram base-line.

Furthermore, there were several viable PoS tag similarity measures to use in the statistical error detection. Lee (1999) gives examples of a few, of which we decided to use Jaccard, Jensen-Shannon, L1, cos and L2.

As stated, the arbitrary threshold e was the limit under which trigram frequencies are considered ungrammatical. By setting e to large values, we obtained higher recall from the algorithm and by setting e to small values, we obtained higher precision. We used $e = 0.25, 0.5, 1, 2, 4$ and so forth up to 512.

The material was scrutinized by the algorithm and the putative errors were marked. Since the minimum resolution of the algorithm is a trigram of words/tags, the algorithm

identified the center of the error, and an error within the trigram was deemed correctly identified. From this, we see that the definition of precision and recall will be as follows:

$$recall = \frac{\text{\# errors overlapped by any detection}}{\text{total \# of introduced errors}},$$

$$precision = \frac{\text{\# of detections overlapping an error}}{\text{total \# of detections}}.$$

4. Results

The characteristics of the similarity measures coincided with the findings of Lee (1999), where Jensen-Shannon, L1 and Jaccard were superior to the other measures and had very similar performance. For the sake of exposition, we chose to limit our findings to Jensen-Shannon, which seemed to have a stable performance over all tests.

The results of the experiments are shown in Figures 1 through 5 corresponding to the percentage of errors in the text, i.e. 1%, 2%, 5%, 10% and 20%. In each figure, four graphs are displayed. These are the four combinations of the statistical method and the transformations turned either on or off. When the error threshold e is increased, the precision drops and the recall increases.

5. Discussion

The algorithm is designed to detect context-sensitive spelling errors. For normal spelling errors and typical grammatical errors, other more suitable algorithms exist. Thus, the proposed algorithm is best used in combination with these to be able to detect all error types in a text. All algorithms will produce false alarms (i.e. correct text marked as an error), and using more algorithms at the same time will produce more false alarms. Hence, to be able to use the proposed method in combination with others, we want to focus on high precision.

Normally, only a small amount of the spelling errors in a text are context-sensitive (that is, result in existing words). Peterson (1986) reports that 16% of the errors may fall into this category (in English, but the results would be similar for Swedish), depending on the size of the dictionary. This is a small fraction of all errors and thus, the 1% and 2% error levels are the most realistic, and the others are shown as comparison.

We see from the figures that using both methods (statistics and transformations) obtains the highest precision at all error levels. Furthermore, both statistics and transformations contribute to this increase.

We also see that the base-line (no statistics, no transformations) obtains the highest recall, although at a very low precision. When the error levels increase, finding errors is less difficult and the precision increases. This also causes the base-line to obtain a precision closer to the other methods. Nevertheless, at the lower (and realistic) error levels, the proposed method achieves a much higher precision at the expense of a loss in recall. See e.g. Figure 1 and compare the highest precision of the base-line (precision 23% at recall 42%) with the proposed method (e.g. precision 50% at recall 26%). The best precision/recall achieved for the proposed method is about 57% precision at 20% recall (at the 1% error level) and about 68% precision at 20% recall (2% errors). Keep in mind here that we cannot expect to achieve high recall while keeping reasonable precision due to the difficult nature of the errors.

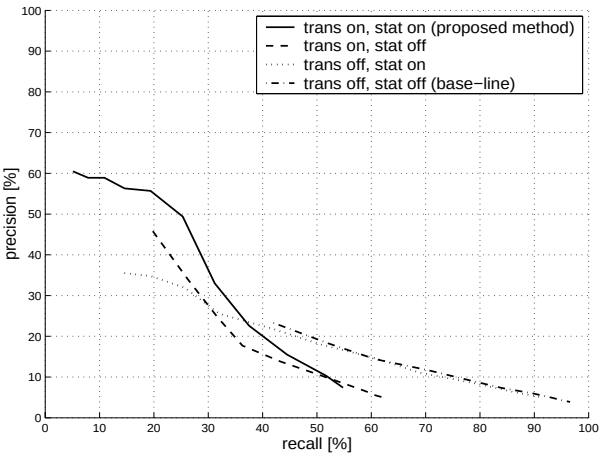

Figure 1: *Precision and recall at the 1% error level. The graphs show the four combinations of the transformations and statistics part of the algorithm turned either on or off.*

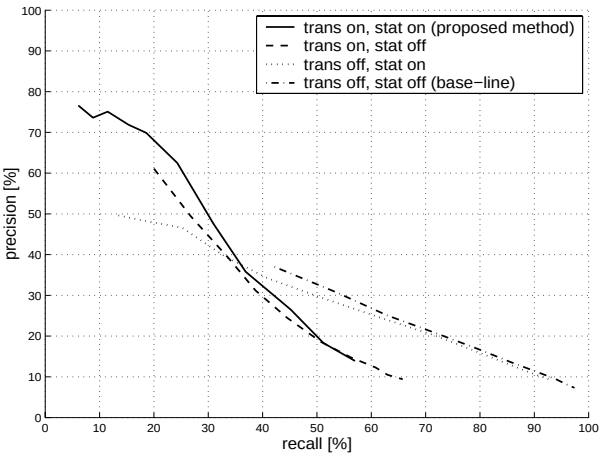

Figure 2: *Precision and recall at the 2% error level.*

We also evaluated the text without errors (0% error level). The text actually contained some spelling and style errors, but these were excluded from the above evaluation. Naturally, the 0% evaluation results depended on the threshold e, but the general observation was that precision ranged from $1/2$ to $2/3$ for reasonable values of e.

Clearly, the performance of the parser and tagger greatly affects the algorithm. Due to the inherent robustness of the PoS tagger, some spelling errors will not result in a change in input to the proposed algorithm. For example, if we introduce 20% errors, only 12.7% of the tags are erroneous from the PoS tagger (Bigert et al., 2003b)! This results in a lowered recall, since some of the errors are just out of reach for an algorithm working on the output of the PoS tagger. Nevertheless, this is also the situation in normal use of the algorithm.

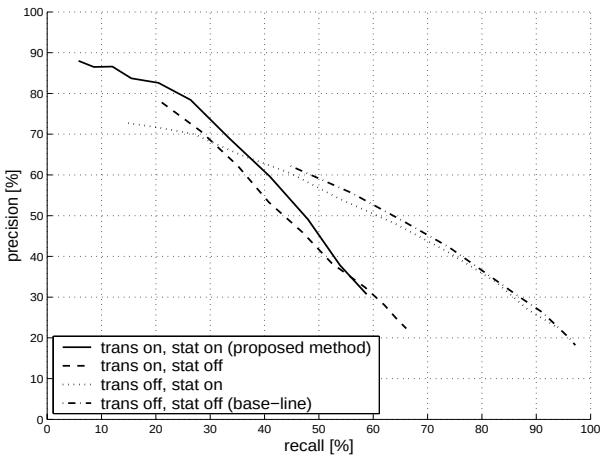

Figure 3: *Precision and recall at the 5% error level.*

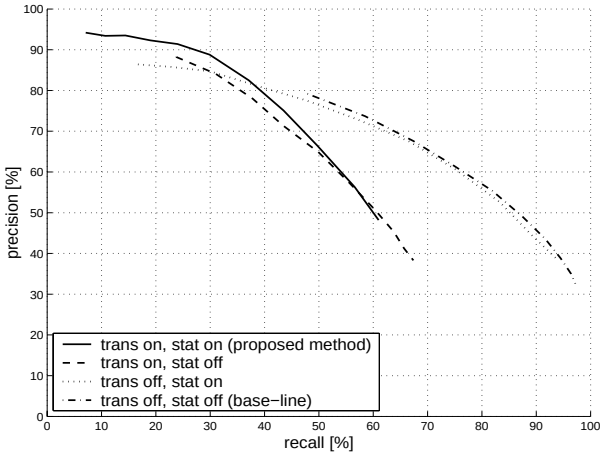

Figure 4: *Precision and recall at the 10% error level.*

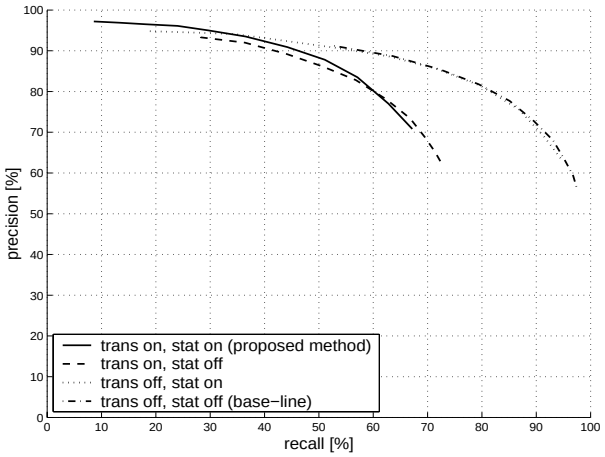

Figure 5: *Precision and recall at the 20% error level.*

6. Conclusions

We have approached the very difficult problem of detecting context-sensitive spelling errors by using statistical information and parser technology. The proposed method requires much less manual work than say a full parser approach. Furthermore, it is much more robust to ill-formed or noisy input.

A thorough evaluation was carried out on context-sensitive spelling errors caused by keyboard mistypes. We saw that the choice of PoS tag distance measures did not affect the results much (as there were at least three measures with similar characteristics). The base-line achieved high recall, but at terrible precision. To achieve high precision, the proposed method was used. As always, higher precision was gained at the expense of recall.

We noted that many of the errors were not possible to detect at all. Having this in mind, the recall and precision of the algorithm were acceptable, while the manual work was kept to a minimum. We conclude that the algorithm could favorably be used with traditional spelling and grammar checkers to more extensively map the spelling errors in a text.

7. Acknowledgments

Many thanks to my supervisor Viggo Kann, the Royal Institute of Technology and the Swedish Agency for Innovation Systems (Vinnova).

8. References

Atwell, E., 1987. How to detect grammatical errors in a text without parsing it. In *Proceedings of the 3rd European ACL Conference*.

Bigert, J., L. Ericson, and A. Solis, 2003a. Missplel and AutoEval: Two generic tools for automatic evaluation. In *Proceedings of Nodalida 2003*. Reykjavik, Iceland.

Bigert, J. and O. Knutsson, 2002. Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proceedings of Robust Methods in Analysis of Natural language Data (ROMAND'02)*. Frascati, Italy.

Bigert, J., O. Knutsson, and J. Sjöbergh, 2003b. Automatic evaluation of robustness and degradation in tagging and parsing. In *Proceedings of RANLP 2003*. Bovorets, Bulgaria.

Ejerhed, E., G. Källgren, O. Wennstedt, and M. Åström, 1992. *The Linguistic Annotation System of the Stockholm-Umeå Project*. Department of Linguistics, University of Umeå, Sweden.

Golding, A. and D. Roth, 1996. Applying winnow to context-sensitive spelling correction. In *Proceedings of the International Conference on Machine Learning*.

Golding, A. and Y. Schabes, 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In A. Joshi and M. Palmer (eds.), *Proceedings of the 34th Annual Meeting of the ACL*. San Francisco.

Knutsson, O., J. Bigert, and V. Kann, 2003. A robust shallow parser for Swedish. In *Proceedings of Nodalida 2003*. Reykjavik, Iceland.

Lee, L., 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the ACL*.

Peterson, J., 1986. A note on undetected typing errors. *Communications of the ACM*, 29(7):633–637.