Faking Errors to Avoid Making Errors: Machine Learning for Error Detection in Writing

Jonas SJÖBERGH
KTH Nada
SE-100 44 Stockholm, Sweden
jsh@nada.kth.se

Ola KNUTSSON
KTH Nada
SE-100 44 Stockholm, Sweden
knutsson@nada.kth.se

Abstract

This paper describes a method to detect errors in written text which requires no manual work. The method used is to simply annotate a lot of errors in written text and train an off-the-shelf machine learning implementation to recognize such errors. To avoid manual annotation synthetically created errors are used for training. The method is evaluated on erroneously split compounds and word order errors. Results are comparable to a state of the art grammar checker based on manually created rules. The evaluation is performed on real (not synthetic) errors.

1 Introduction

Automatic grammar checking is traditionally done by manually written rules, constructed by a computer linguist. We present a method that saves a lot of work by training a machine learning algorithm on automatically created errors.

Methods for detecting grammatical errors without using manually constructed rules have been presented before. Atwell (1987) uses the probabilities in a statistical part-of-speech tagger, detecting errors as low probability part-of-speech sequences. A similar method is presented by Bigert and Knutsson (2002), where new text is compared to known correct text and deviations from the "norm" are flagged as suspected errors. Chodorow and Leacock (2000) present a method based on mutual information measurements to detect incorrect usage of difficult words.

Training a machine learning algorithm to detect errors has been tried by Izumi et al. (2003), who manually annotated errors in (transcribed) spoken language to detect new errors. Mangu and Brill (1997) used machine learning to detect when one word has been confused with another. Golding (1995) combines several methods to solve the same problem.

Unlike most of the methods mentioned our method is applicable to a wide range of error types. Our method is similar to the one presented by Izumi et al. (2003). A big difference is that our method does not require manual annotation of errors (time consuming and expensive), nor access to large amounts of human produced (unintentional) errors. Machine learning generally requires many training examples, which might be hard to find for rare error types.

2 Method to Detect Errors

Our method works by automatically adding errors to a lot of (mostly error free) unannotated text. When adding an error, the words that now make up the error are marked "error". All other words are marked "correct". The resulting annotated text is then used as training data for a machine learning algorithm. The machine learner is thus trained to detect new errors of the same type that was automatically generated.

It is possible to train many machine learning modules, each on a different type of error, by simply generating different types of errors. It is also possible to generate many different types of errors and training a single machine learner to recognize all types. We use the first approach. One big benefit of this is that it is often possible to suggest corrections if only one type of error is detected. For instance, if word w and the following word u are both marked as errors by a module that detects word order errors the correct word order is likely achieved by putting u before w.

Our method is most useful for those types of errors which are easy to generate but result in "unpredictable" changes to the text. Examples include split compounds and word order errors. It is easy to split a compound or change the word order, but it is not obvious what the resulting sentence structure would be when these

errors occur. Thus it is hard to create rules for them manually.

Since our method uses synthetically generated errors no manual work is required to build the error detecting modules. The machine learner would probably be better at detecting errors if it was trained on real errors, since it would then have a proper view of what kind of errors actual writers make. This would require quite a lot of manual work to find and annotate the errors (and to produce text with unintentional errors in), though, so we have not tried this. Collecting real errors should be feasible, though. Students, for instance, write a lot of text that a teacher later checks for correctness. Publishers and newspapers also do manual proofreading of a lot of text.

The transformation based rule learner fnTBL (Ngai and Florian, 2001) was used as the machine learning implementation in our experiments. It produces rules that are very easily understood by humans. This means that the generated rules could be handed over to a linguist with no computer skills. Possibly the linguist could tweak the automatically learned rules (using linguistic knowledge) to get better performance. The learner generates a lot of rules (several hundred) though.

3 Evaluation

We evaluate this error detection method on two different problems, erroneously split compounds (a bit like writing "a wet nurse" when "a wetnurse" was intended; a common problem in for instance Swedish and German) and word order errors (relevant for any fixed order language). Both error types were evaluated on Swedish texts.

The results are compared to three grammar checkers for Swedish.

- Granska (Domeij et al., 2000), a state of the art grammar checker based on manually constructed rules for many error types.
- Probgranska (Bigert and Knutsson, 2002), an extension to Granska, based on statistics, which detects errors by looking for things that are "different" from known correct text.
- The Swedish grammar checker in Microsoft Word 2000, which uses a grammar checker developed by Lingsoft (Arppe, 2000; Birn, 2000).

The automatic method, based on synthetically created errors, generally has lower precision but higher recall than the manual rules. The automatic method can also be tailored (somewhat) for high recall (at the cost of precision) or high precision (at the cost of recall). The manual rules and the automatic method complement each other somewhat, so combining them gives good results.

3.1 Split Compounds

In compounding languages, such as Swedish or German, a common error is to split a compound word. This is especially common with non-native speakers with a non-compounding native language.

To find erroneously split compounds a one million words corpus of written Swedish, the Stockholm-Umeå Corpus, SUC (Ejerhed et al., 1992), was used as training data. A modified spell checker (Domeij et al., 1994; Kann et al., 2001) was used to automatically split compounds. The training data consisted of the corpus texts (to show correct language use) and another copy of all the corpus texts, but with all compounds recognized by the compound splitter split into their components. The splitter marked all components from split compounds with "error" and all other words were marked "correct".

The machine learning implementation used was fnTBL, a transformation based learner. The features for the learner was the word itself and its part of speech (PoS). The PoS was automatically assigned, using a statistical PoS tagger, TnT (Brants, 2000). TnT was automatically trained on the SUC, which is PoS annotated. The text was also parsed using the automatic parser GTA (Knutsson et al., 2003) and chunks were extracted. Giving chunk information to fnTBL was also tried.

The initial guess for each word was its unigram annotation, i.e. the most common of "error" or "correct" as annotation for this word in the training data.

The rule templates used unigrams, bigrams and trigrams of words, PoS, and error annotation. A few combinations of these were also allowed, such as the current word and annotation trigrams. When chunks were used, unigrams, bigrams and trigrams of chunks were also used in the rule templates.

The test data was 5 124 words, of which 812 were components from split compounds. Most

	Granska (manual)	PoS only	Chunk	Filtered	Baseline	Baseline Filtered	One	Both
Detected	322	588	594	545	331	120	593	274
Missed	490	224	218	267	481	692	219	538
False alarms	6	49	49	28	162	6	33	1
Precision	98%	92%	92%	95%	67%	95%	95%	100%
Recall	40%	72%	73%	67%	41%	15%	73%	34%

Table 1: Detection of split compound components. The baseline is simply the most common tag for each word ("error" or "correct"), from the training data. "One" is any word marked "error" by either the manual rules or the filtered automatic rules. "Both" is any word marked by both.

compounds consisted of only two components. All such components were manually annotated with "error" and all other words automatically marked "correct". These sentences were taken from real texts, i.e. these were not synthetic errors.

The results of training on synthetic split compound errors to detect real split compounds is summarized in Table 1. In general the recall is much higher for the automatic learner than for the manual rules of the grammar checker Granska, but the precision is much worse (though still quite good). The test data is possibly to favorable for Granska, since the linguist developing the rules for Granska looked at these sentences during early rule development. The rules have changed a lot since then. Evaluations of Granska on other texts give similar recall but much lower precision.

Granska is one of the few grammar checkers that actually tries to detect split compounds and is likely the best grammar checker currently available for this. Some grammar checkers for Swedish only detect split compounds when they result in some other common error type (such as disagreement between adjective and noun).

A commercial grammar checker for Swedish is included in Microsoft Word. We ran our test data through Microsoft Word 2000. It does not detect split compounds per se but split compounds sometimes look like other types of errors that Word recognizes. On our test data Word indicated an error 66 times related to split compounds. 15 errors were caused by the compound also being misspelled, 18 were caused by a compound component not known by the grammar checker to be an actual word (such as proper nouns, "Rambo"), 17 split compounds looked like spelling errors (caused by the changed morphology of the head), 15 looked like agreement errors and 1 as erroneous usage of a verb. The errors would correspond to 75 detected errors for the other methods.

The Probgranska extension to Granska often finds split compounds. On our test data it generates 117 alarms, most of which are for split compounds. The output of Probgranska is not directly comparable to the data in Table 1 but it would correspond to 171 correct detections (generously counted), 23 false alarms and 6 errors of other types (such as a missing word).

To improve the precision of the automatically learned rules, all detected errors were filtered through the spell checker Stava (Domeij et al., 1994). To report a word as an actual error it had to be possible to combine the word with a neighboring word also suspected to be an error, and the combination should be a compound word recognized by the spell checker. The precision increased a lot through this but it did also remove quite a few actual split compounds. Most of these were compounds that were misspelled as well as erroneously split (there were quite a lot of errors in the test data, not just split compounds).

Since the test data contains a lot of errors, about one erroneously split compound per sentence, it is perhaps too easy to reach a high precision. On the other hand, there are a lot of other errors too in the test texts, which make it harder to detect the split compounds. To see how the method performs on the other end of the spectrum, the trained learner was run on text from a high quality newspaper, which has very few errors. On 10 000 words of newspaper text (not part of the training data) there were 72 false alarms, using filtering but without using chunking which usually improves precision a little. There were also 8 correctly detected split compounds missed by the proofreaders. This indicates that false alarms is not a great problem.

	Granska (manual)	Random Naive	Random Trigram	Verbs Naive	Verbs Trigram	Verbs Mix	Manual + Random	Manual + Verbs
Detected	9	14	13	47	53	42	21	59
Missed	132	127	128	94	88	99	120	82
False alarms	1	6	6	31	35	24	7	36
Precision	90%	70%	68%	60%	60%	64%	75%	62%
Recall	6%	10%	9%	33%	38%	30%	15%	42%

Table 2: Detection of word order errors. False alarm only means that no word order error occurred at the suggested position. Many false alarms were caused by other types of errors. "Random" means that errors were generated by moving a randomly selected word, "Verbs" means only verbs and the Swedish word for "not" were selected. "Naive" means that the initial guess of the system was that all words were correctly placed, "Trigram" means the initial guess was based on part-of-speech trigrams and "Mix" means that trigrams were used for the training data and the naive method for the test data (to increase precision). The last two columns combine Granska with the automatic rules by reporting any error found by at least one method.

The manual rules of the grammar checker and the automatic rules complement each other somewhat. That means that the precision or the recall can be improved by combining these methods. In Table 1 the results of using the union (i.e. at least one method thinks it is a split compound, high recall) and the intersection (i.e. both methods must believe it is an error, high precision) is shown.

3.2 Word Order Errors

Word order errors are not very common in Swedish for native speakers, but second language learners make many word order errors. Swedish has different word order in different types of clauses, such as reversed word order for questions. In Swedish the placement of adverbial phrases is different in subordinate clauses compared to the main clause, which is a common difficulty for learners of Swedish.

To create a word order error detector the SUC corpus was again used as a reference text and fnTBL as the machine learner. The rule templates used the same features as for split compounds, i.e. words, PoS and the "error/correct" annotation. Since the correct word order in Swedish is different in different types of clauses adding the first word of the clause was also tried. This made almost no difference except demanding more processing time and memory for the training step so it was only tested once. Chunking also had very little impact, so it too was only tested a few times.

Word order errors turned out to be much harder to detect than split compounds, so several different strategies for generating synthetic errors was tried. It was also hard to find a good initial guess for the annotation, so several strategies was tried for this too.

As test data 70 sentences from second language learner essays were manually annotated by marking all word order errors with "error". Usually a word order error resulted in two words being marked "error" and would be corrected if these two switched places. There were 1 241 words, of which 141 words were marked "error". There were also a lot of errors that were not word order errors. These were marked "correct" (in the sense that they are not word order errors). Such words caused many of the false alarms.

To generate synthetic errors the first strategy used was to simply switch places of one randomly selected word and a neighboring word in each sentence. The second strategy was to select only among verbs and the word "inte" (which is Swedish for "not") and switching them and a neighboring word. This was based on the assumption that most word order errors in Swedish concerns the verb of the sentence, and that "inte" is hard to place correctly. This method also generated more than one error per sentence (on average two errors per sentence). Both error generation strategies also added a copy of each sentence without any introduced errors to the training data. By adding more word order errors to the training data the recall can be improved dramatically, but at a high cost in precision.

The first strategy for an initial guess for the learner was that all words are "correct". This is very uninformative and does not give the learner

much to work with. The second strategy was to mark words based on the PoS trigram centered on the word. Trigrams that were more often marked as "error" than "correct" in the training data were given an initial guess of "error". Any trigram seen only one or two times in the training data was also marked "error". The idea was that this forces the learner to learn more rules that correct false alarms, hopefully increasing precision.

In the test data there are a lot of PoS trigrams that do not occur at all in the training data. If these are marked "error" it gives a lot of "false alarms", since these are often not word order errors but some other form of error. These false alarms are somewhat reduced when using the second strategy, but not very much. To avoid false alarms such trigrams were marked "correct" in the initial guess.

The results of automatic detection of word order errors is summarized in Table 2. The results are not as impressive as for split compounds, though the automatic method still has higher recall than the manual rules.

Since there are a lot of errors in the test data, the word order error detector was also run on 10 000 words of newspaper texts (not part of the training data). These are proofread and contain very few errors. Training using the second strategy for generating the training errors resulted in 363 alarms on the newspaper texts. Of these 6 were actual word order errors missed by the proofreaders, 2 were caused by other errors in the text (wrong tense of verbs) and the rest were false alarms. Many false alarms were caused by quotes from interviews using sentence structure rarely used in written text (the training data consisted mostly of written text).

The grammar checker Granska only has a few rules for word order errors, which are by no means meant to catch all such errors. The Probgranska extension to Granska can detect some types of word order errors. In our test data it finds five word order errors, six errors of other types (such as using "small" instead of "little") and makes two false alarms. The output of Probgranska is not directly comparable to the data in Table 2 but a reasonable translation would give it 6 correct detections and 2 false alarms.

The grammar checker in Microsoft Word 2000 has the ability to detect some types of word order errors. We ran our test data through Word but it did not detect any of the word order errors present in our test data.

The manual rules of Granska and the automatic rules complement each other much more for word order errors than for split compounds. There is just two or three (depending on which error generation method is used) errors that they both detect. This means that the combination of both methods could be very useful, mainly for high recall. In Table 2 the results of this is presented. They could also be combined for high precision, the precision will be 100% if reporting only those errors that both methods agree on (they have no common false alarms), but the recall will be very low (2%).

4 Conclusions

We presented an error detection method that requires no manual work. It works quite well for detecting errors. It has (much) higher recall than manually constructed rules, but lower precision. Using manually annotated real errors would likely give better performance, but requires manual work. A nice property of the method is that it is easy to suggest a correction.

While using more sophisticated NLP tools such as chunking did improve the results a little, using only part of speech tagging achieved results that are still very useful. This means that not only is no manual work required, only commonly available resources such as unannotated text and a PoS tagger is required.

It is especially interesting that the method works so well for split compounds. This is a very common problem for second language learners of Swedish (and quite common in informal texts by native speakers). It is also a hard problem to write rules for manually. Very few grammar checkers handle these errors.

If several different modules are trained to detect different types of errors they can be combined into one framework that detects many error types. In this case false alarms become a problem, since even if each module only produce a few false alarms the sum of them might be too high. In our tests many false alarms were caused by some other type of error occurring, so this might not be such a big problem. It is also possible to steer the machine learner towards high precision (few false alarms).

5 Acknowledgments

We thank Viggo Kann for contributing useful ideas and helpful suggestions.

This work has been funded by The Swedish Agency for Innovation Systems (VINNOVA).

References

Antti Arppe. 2000. Developing a grammar checker for Swedish. In T. Nordgård, editor, *Proceedings of Nodalida '99*, pages 13–27. Trondheim, Norway.

Eric Steven Atwell. 1987. How to detect grammatical errors in a text without parsing it. In *Proc. of the 3rd EACL*, pages 38–45, Copenhagen, Denmark.

Johnny Bigert and Ola Knutsson. 2002. Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proceedings of Romand 2002, Robust Methods in Analysis of Natural language Data*, pages 10–19.

Juhani Birn. 2000. Detecting grammar errors with lingsoft's Swedish grammar checker. In T. Nordgård, editor, *Proceedings of Nodalida '99*, pages 28–40. Trondheim, Norway.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, pages 224–231, Seattle, USA.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of NAACL'00*, pages 140–147, Seattle, USA.

Rickard Domeij, Joachim Hollman, and Viggo Kann. 1994. Detection of spelling errors in Swedish not using a word list en clair. *Journal of Quantitative Linguistics*, 1:195–201.

Richard Domeij, Ola Knutsson, Johan Carlberger, and Viggo Kann. 2000. Granska – an efficient hybrid system for Swedish grammar checking. In *Proceedings of Nodalida '99*, pages 49–56, Trondheim, Norway.

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Magnus Åström. 1992. The linguistic annotation system of the Stockholm-Umeå Corpus project. Technical report, Department of General Linguistics, University of Umeå (DGL-UUM-R-33), Umeå, Sweden.

Andrew Golding. 1995. A bayesian hybrid for context sensitive spelling correction. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 39–53, Cambridge, USA.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Companion Volume to the Proceedings of ACL '03*, pages 145–148, Sapporo, Japan.

Viggo Kann, Rickard Domeij, Joachim Hollman, and Mikael Tillenius. 2001. Implementation aspects and applications of a spelling correction algorithm. In L. Uhlirova, G. Wimmer, G. Altmann, and R. Koehler, editors, *Text as a Linguistic Paradigm: Levels, Constituents, Constructs. Festschrift in honour of Ludek Hrebicek*, volume 60 of *Quantitative Linguistics*, pages 108–123. WVT, Trier, Germany.

Ola Knutsson, Johnny Bigert, and Viggo Kann. 2003. A robust shallow parser for Swedish. In *Proceedings of Nodalida 2003*, Reykjavik, Iceland.

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning*, pages 187–194.

Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, pages 40–47, Carnegie Mellon University, Pittsburgh, USA.