# Faking Errors to Avoid Making Errors: Very Weakly Supervised Learning for Error Detection in Writing

**Jonas Sjöbergh**
KTH KOD
SE-100 44 Stockholm, Sweden
jsh@nada.kth.se

**Ola Knutsson**
KTH KOD
SE-100 44 Stockholm, Sweden
knutsson@nada.kth.se

## Abstract

This paper describes a method to create a grammar checker "for free". It requires no manual work, only unannotated text and a few basic NLP tools. 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 artificially created errors are used for training. Recall is comparable to other grammar checkers but precision is lower. Our method also complements traditional grammar checkers, i.e. they do not always find the same errors. The evaluation is performed on real errors.

## 1 Introduction

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

Methods for detecting grammatical errors without using manually constructed rules have been presented before. (Atwell 87) uses the probabilities in a statistical part-of-speech tagger, detecting errors as low probability part-of-speech sequences. A similar method is presented in (Bigert & Knutsson 02), where new text is compared to known correct text and deviations from the "language norm" are flagged as suspected errors. (Chodorow & Leacock 00) present a method based on mutual information measurements to detect incorrect usage of difficult words.

(Mangu & Brill 97) use machine learning to detect when one word has been confused with another. (Golding 95) combines several methods to solve the same problem. (Hardt 01) treats comma placement and determiner-noun agreement in Danish as a confusion set problem in a similar way. He also uses artificial errors as negative examples. Another example of a confusion set problem is English article usage before noun phrases (Han *et al.* 04).

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.* 03), who manually annotated errors in transcribed spoken language.

Our method is based on viewing grammar checking more or less as a tagging task. We simply train an available machine learning algorithm on annotated errors to create a grammar checker. The new idea in our approach is to use only artificial errors for training, and we show that while it might not be as good as training on real errors, it still produces a useful grammar checker. The strength of this approach is that it is very resource lean. No time consuming manual annotation of errors is needed, neither is access to large amounts of human produced (unintentional) errors. Almost no manual work at all is required, only unannotated text and a few basic NLP tools are used.

## 2 Method of Detecting Errors

The basic idea of our method is to treat grammar checking as a tagging task. Collect a lot of text, mark all errors with "ERROR" and all other words with "OK". Train an off-the-shelf tagger on this data and you have a grammar checker. To achieve better feedback it is possible to have different tags for different types of errors, i.e. "SPELLING", "VERB-TENSE", etc. Another way to achieve this is to train a new specialized classifier for each error type, which ignores other types of errors.

Finding these errors and annotating them requires a lot of work. Our method avoids this by using artificial errors. A lot of text without errors is used, and the text is then corrupted by adding errors. Since they are added automatically they can be annotated at the same time. When this is done we automatically annotate the resulting text with part-of-speech (PoS), using TnT (Brants 00). The words, PoS and error anno-

tation is then used as training data for the automatic grammar checker. Almost any machine learning implementation could be used for this. We use fnTBL (Ngai & Florian 01), a transformation based rule learner, which produces rules that are easily understood by humans.

Below is an example of an error generation program, for agreement errors. When implemented in a high level scripting language, the code is not much longer than this pseudo code. Since the main strength of our method is that it is resource lean, the simpler the error generation the better.

```
(1) Read lemma lexicon (or stems)
(2) Read PoS-tags with agreement constraints
(3) Run PoS-tagger
(4) For each tagged sentence:
(5)   Pick random word with agreement constraint
(6)   Get lemma (lexicon)
(7)   Get random word with this lemma (lexicon)
(8)   If not exact same word:
(9)     Change word, mark as error
```

If we run the error generation code on "I bought a car." we could get for instance "I/OK bought/OK a/OK cars/ERR ./OK".

The error generation programs sometimes change a sentence so that the result is still grammatical. One simple example would be a program that inserts word order errors by randomly changing the order of neighboring words. Not all changes will lead to errors, for example "I heard dogs barking" and "I heard barking dogs" are both correct, but "heard I dogs barking" is not. Such sentences will of course still be marked as erroneous. This is not a great problem, since if something is correct there are usually many examples of this which are not the result of changes, and thus marked as correct. This means that the learner will in general only learn rules for those artificial errors that result in text which is incorrect, since the other "errors" will be drowned out by all the correct examples.

Our method can be used on many error types. Some examples of errors that could be generated artificially include: word order errors (reorder randomly selected words), missing words (remove randomly selected words), "hard" spelling errors (replace words with another word with only a one letter difference), split compounds (replace all words that could be made from concatenating two other words in the corpus with these two words), agreement errors and verb tense errors (use a dictionary lookup to replace words with another inflectional form of the same word), prepositional use (change prepositions to other prepositions), etc.

The main strength is errors that are simple to generate, but where the resulting sentence structure is hard to predict. Word order errors and split compounds are examples of such errors. Errors such as repeated words for which it is straightforward to predict the result can also be handled by our method, but is probably better handled by traditional methods.

We have tested our method on two different error types: split compounds, an error type suited to our method, and agreement errors, suited to traditional grammar checking methods. Agreement errors were tested to see how our method holds up where the competition is the hardest. We compared our method to three other grammar checkers, evaluating them on Swedish texts of different genres.

### 2.1 Split Compounds

In compounding languages, such as Swedish and German, a common error is to split compound words, i.e. write "quick sand" when "quicksand" was intended. Two concrete examples from Swedish: (1) "en långhårig sjukgymnast" means "a physical therapist with long hair". Splitting the compounds to make "en lång hårig sjuk gymnast" is still grammatical but the meaning is changed to "a tall, hairy and sick gymnast". (2) If the compound "ett personnummer" ("social security number") is split to "ett person nummer" ("one person number") it would lead to an agreement error and be ungrammatical.

Training data for the erroneously split compounds experiments was a one million words corpus of written Swedish, the Stockholm-Umeå Corpus, SUC (Ejerhed *et al.* 92). A modified spelling checker, Stava (Domeij *et al.* 94; Kann *et al.* 01) was used to automatically split compounds.

While some manual work has been put into creating Stava (and thus in a sense made this type of error generation less independent of manual work), the part used here, i.e. the compound analysis component, was automatically constructed from a dictionary. If however there are tools available that someone already put a lot of manual

effort into creating, our method could use these. Our method would then be a method of creating a grammar checking component from other tools in an unsupervised way.

The training data consisted of the corpus texts, to show correct language use, and another copy of all the corpus texts. The second copy had all compounds recognized by the compound splitter split into their components, with the components marked "error".

The rule learner was given the word n-grams, PoS n-grams and error annotation n-grams. The n-grams were unigrams, bigrams and trigrams. Some combinations of these were also allowed, such as the current word and error annotation trigrams. The initial guess for the learner was that words more common in compounds than as a single word (in the training data) were probably errors and all other words correct. The best rules found by the learner used PoS bigrams and error annotation of one word and PoS of its neighbor.

To improve the precision of the learned rules one can use the fact that if a compound is split it will result in at least two components. Any single word marked error is thus probably a false positive (or one of its neighbors is a false negative), and can be removed. Since there was a spelling checker available we improved this a little by filtering the output through the spelling checker. If a suspicious word could not be combined into a correct compound by using a neighboring word also marked "error" it was considered a false alarm and the error was removed. This improved the precision but also removed many correctly detected split compounds, usually because they were misspelled as well as erroneously split (and would thus be found by the spelling checker instead).

Using the spelling checker gave only a small improvement over just removing errors with no neighboring error, while both methods improved the precision of the original rules significantly.

### 2.2 Agreement Errors

In Swedish, determiners, adjectives, possessives and nouns must agree in number, gender and definiteness. Agreement errors are quite common, especially when revising text using a computer. The agreement can span long reaches of text, which can make the errors hard to detect. Manually writing good rules for agreement errors is relatively straightforward, and it is one of the more popular error categories to detect among automatic grammar checkers.

To generate artificial errors the SUC corpus was used again. In each sentence a word from any word class with agreement restrictions was randomly selected. This word was then changed to another randomly selected form of the same word. This was done by a simple lexicon lookup were the lemma of the word was found and another word with the same lemma and a different surface form was selected. The selected word was marked as an error and all other words were marked as correct.

When an agreement error occurs, at least two words are involved. We only mark the changed word as an error, although it would be reasonable to mark all words with agreement restrictions related to the changed word. One reason for this is that it is easy to mark the changed word but hard to mark the other words. If we could find them, we would already have an agreement error detection method. Also, since we know which word was changed, we know which word should be corrected to retrieve the intended meaning, even though the agreement error itself could likely be corrected in several ways.

As features for the machine learner the gender, number and definiteness of the word were given (if applicable). All this information is included in the tagset we trained TnT on, and was automatically assigned. The PoS of the word and the error annotation were also included. Unigrams, bigrams, trigrams and combinations of these features were used. The best rules combined PoS and n-grams of the gender features.

The initial guess was that there were no errors in the text. A baseline was constructed by locating every occurrence of two consecutive words that had different gender, number or definiteness and marking the first of these as an error. This baseline could be used as initial guess for the learner, which gives higher precision than the original initial guess, since many rules are learned that remove alarms (mostly spurious alarms from the baseline), but lower recall.

## 3 Evaluation

We compared our method to three different grammar checkers for Swedish, one commercial grammar checker, one state of the art research product and one method not based on manually written rules.

| | MS Word (manual) | ProbGr. (statistical) | Granska (manual) | SnålGr. | SnålGr. + Filter | Baseline | Baseline + Filter | Union | Inter-section |
|---|---|---|---|---|---|---|---|---|---|
| Detected errors | 75 | 225 | 322 | 588 | 535 | 331 | 120 | 582 | 275 |
| False negatives | - | - | 490 | 224 | 277 | 481 | 692 | 230 | 537 |
| False positives | - | - | 6 | 49 | 24 | 162 | 6 | 29 | 1 |
| Precision | - | - | 98% | 92% | 96% | 67% | 95% | 95% | 100% |
| Recall | - | - | 40% | 72% | 66% | 41% | 15% | 72% | 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. "Union" is any word marked "error" by either the manual rules of the Granska grammar checker or the filtered automatic rules of SnålGranska (our method). "Intersection" is any word marked by both. MS Word and ProbGranska do not specifically address the problem of split compounds but find some anyway, but of course with a different diagnosis.

- The Swedish grammar checker in Microsoft Word 2000, which uses a grammar checker developed by Lingsoft (Arppe 00; Birn 00). It is based on manually constructed rules. The rules are tuned for high precision.

- Granska (Domeij *et al.* 00), a state of the art grammar checker, also based on manually constructed rules. Roughly 1 000 hours of manual work have been put into creating the rule set.

- ProbGranska (Bigert & Knutsson 02), a statistically based grammar checker. It detects errors by looking for things that are "different" from known correct text, based on PoS trigrams. ProbGranska is currently used as a complement to the manual rules of Granska in a grammar checking environment.

### 3.1 Evaluation on Collections of Errors

The first evaluation was performed on collections of examples of authentic split compounds and agreement errors. These were all taken from real texts, but since there is at least one error in each sentence it is a quite unrealistic data set, and it is easy for the grammar checkers to achieve high precision with so many errors available. The benefit of these collections is that all errors that occur have been manually annotated, so it is easy to check the precision and recall of the grammar checkers. Since these are real errors a grammar checker with a good result on these texts will likely work well on "real" texts too.

For split compounds examples were taken mostly from web pages and newspapers. There were 5 124 words, of which 812 were components from split compounds. Most compounds consisted of only two components. Sometimes two (but rarely more) adjacent compounds were both split. The results are shown in Table 1.

For split compounds the results are quite good. Compared to the other grammar checkers, the automatically learned rules have lower precision but the highest recall. Detecting split compounds is considered quite hard, and Granska is one of the few grammar checkers that actually tries to detect split compounds. It is likely the best grammar checker currently available for this.

The grammar checker in MS Word 2000 does not look for split compounds but these errors sometimes look like other types of errors that MS Word recognizes. On the test data MS Word classed 75% of the detected split compounds as spelling errors. One third of these were caused by the split compound also being miss-spelled, one third by the compound containing a word which was not recognized (e.g. "Rambo") and one third by the morphological change of the head of the compound. MS Word classed the remaining detected split compounds as agreement errors.

The ProbGranska extension to Granska often finds split compounds. In the test data most of the alarms generated by ProbGranska are caused by split compounds.

For agreement errors the data consisted of 4 556 words, also mostly from newspapers or the Internet. There were 221 agreement errors in the test data, the results are shown in Table 2.

For agreement errors the results are not as impressive, which is to be expected since agreement errors are one of the best covered error types of traditional grammar checkers. While the automatic rules are outperformed by the manually cre-

| | MS Word (manual) | ProbGranska (statistical) | Granska (manual) | SnålGranska | Baseline | Union | Intersection |
|---|---|---|---|---|---|---|---|
| Detected errors | 71 | 17 | 101 | 88 | 100 | 134 | 54 |
| False negatives | 155 | - | 125 | 138 | 126 | 92 | 172 |
| False positives | 1 | - | 5 | 15 | 143 | 19 | 1 |
| Precision | 99% | - | 95% | 85% | 41% | 88% | 98% |
| Recall | 31% | - | 45% | 39% | 44% | 60% | 24% |

Table 2: Detection of agreement errors. The baseline marks the first of any two consecutive words that have different gender, number or definiteness as an error. "Union" is any word marked "error" by either the manual rules of the Granska grammar checker or the automatic rules of SnålGranska (our method). "Intersection" is any word marked by both. ProbGranska does not specifically look for agreement errors.

ated rules, the results are still good enough to be useful.

The main reason for the lower recall of the automatic rules is that they only work in a small local window. Many of the errors detected by the manual rules span tens of words. Since the automatic rules find none of these errors and still manages to find almost as many errors, there are a lot of errors detected by the automatic rules not found by the manual rules. Combining the two methods thus gives better results than either method individually, as shown in Table 2. They also complement each other, though not as much, on split compounds, as shown in Table 1.

### 3.2 Evaluation on Real Texts

To evaluate the performance on real texts a few sample texts were collected. All grammar checkers were then run on the texts. All words suspected to contain errors by any of the grammar checkers were manually checked to see if it was a real error. The texts were not manually checked to find all errors, since that would require a lot of work and the time was not available. This gives the precision of the grammar checkers, but not the recall since there could be many errors not detected by any of the grammar checkers. It is possible to get an upper bound on the recall though, using the errors missed by one grammar checker and detected by another.

The first genre we evaluated the grammar checkers on was old newspaper articles. These were taken from the Swedish Parole corpus (Gellerstam *et al.* 00), which also contains other genres though only newspaper texts were used here. These texts are very hard for the grammar checkers, since they are well proofread and contain almost no errors. The results are shown in Table 3. The results are not impressive, the precision is very low for all grammar checkers. Since there are almost no errors to find, this is to be expected. The number of false positives (false alarms) gives an indication of whether the grammar checkers would be usable for writers who make few errors. 50 false alarms, as for our presented method, in 10 000 words is probably tolerable, considering that the commercial grammar checker produces about twice as many when including spelling error reports, though of course it also tries to capture more error types.

The second genre was essays written by people learning Swedish as a second language. These were taken from the SSM-corpus (Hammarberg 77). These texts contain a lot of errors, which is generally good for the grammar checkers (easier to get high precision). It also leads to problems though, since many errors overlap and there is often very little correct text to base any analysis on. Results are shown in Table 4. There are a lot of errors that no grammar checker detects, in a sample that was manually checked to find all errors less than half the errors were detected.

The grammar checkers using manually constructed rules show much higher precision (about 95%) than our presented method (about 86%). They also detect many more errors, mainly because they also look for spelling errors, which are common and much easier to detect. When it comes to grammatical errors the recall is comparable to the manual rules. On split compound errors, which our method is well suited for and which are hard to describe with rules, our method performs very well. On agreement errors, which are one of the best covered error types using man-

| | MS Word | ProbGranska | Granska | SnålGranska | Total |
|---|---|---|---|---|---|
| All detected errors | 10 | 1 | 8 | 3 | 13 |
| All false positives | 92 | 36 | 35 | 50 | 200 |
| Detected spelling errors | 8 | 0 | 6 | 1 | 9 |
| False positives | 89 | - | 20 | - | 101 |
| Detected grammatical errors | 2 | 1 | 2 | 2 | 4 |
| False positives | 3 | 36 | 15 | 50 | 99 |
| Detected agreement errors | 0 | 0 | 0 | 1 | 1 |
| Detected split compounds | 0 | 0 | 0 | 0 | 0 |

Table 3: Evaluation on proofread newspaper texts, 10 000 words. Since there are very few remaining errors to detect, performance is less than impressive.

| | MS Word | ProbGranska | Granska | SnålGranska | Total |
|---|---|---|---|---|---|
| All detected errors | 392 | 101 | 411 | 122 | 592 |
| All false positives | 21 | 19 | 13 | 19 | 67 |
| Detected spelling errors | 334 | 34 | 293 | 26 | 362 |
| False positives | 18 | - | 5 | - | 21 |
| Detected grammatical errors | 58 | 67 | 118 | 96 | 230 |
| False positives | 3 | 19 | 8 | 19 | 46 |
| Detected agreement errors | 32 | 9 | 49 | 43 | 74 |
| Detected split compounds | 5 | 8 | 20 | 27 | 35 |

Table 4: Evaluation on second language learner essays, 10 000 words. With many errors in the text high precision is to be expected. Less than half of all errors are detected, though.

ual rules, its performance is still quite good, with similar recall but lower precision compared to the manual rules.

It is also interesting to note that the grammar checkers do not overlap very much in which errors they detect. A total of 230 grammatical errors are detected but no individual grammar checker detects more than 118. Combining different methods, for instance by signaling an error whenever at least one grammar checker believes something is wrong, would thus give much higher recall.

The final genre was student essays written by native speakers, Table 5. Again, the results are not impressive for any of the grammar checkers. Many false alarms stem from quotations, law books and old texts such as the Bible are quoted. These contain text that is grammatical but differs a lot from "normal" language use. There are also false alarms when spoken language constructions that are rare in written texts are used. This is especially true for the two automatic methods, which both compare new texts to the "language norm" they were trained on (in this case written language).

## 4 Conclusions and Discussion

We have presented an error detection method that requires almost no manual work. It works quite well for detecting errors. It has lower precision than state of the art grammar checkers based on manually constructed rules, but the precision is high enough to be useful. For some error types the recall of the new method is much higher than the recall of other grammar checkers.

The greatest advantage of this method of creating a grammar checker is that it is very resource lean. A few minutes were spent on generating artificial errors. Some other resources are also needed but only commonly available resources: unannotated text, a part-of-speech tagger and a spelling checker was all that was used.

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 produces 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. This kind of false

| | MS Word | ProbGranska | Granska | SnålGranska | Total |
|---|---|---|---|---|---|
| All detected errors | 38 | 23 | 48 | 28 | 90 |
| All false positives | 31 | 45 | 13 | 31 | 111 |
| Detected spelling errors | 24 | 3 | 17 | 1 | 25 |
| False positives | 28 | - | 0 | - | 28 |
| Detected grammatical errors | 14 | 20 | 31 | 27 | 65 |
| False positives | 3 | 45 | 13 | 31 | 83 |
| Detected agreement errors | 5 | 0 | 11 | 8 | 15 |
| Detected split compounds | 0 | 1 | 1 | 1 | 1 |

Table 5: Evaluation on essays written by native speakers, 10 000 words. Frequent use of spoken language style and quotations from for instance legal documents lead to a lot of false alarms in these essays.

alarm might not be a serious problem, since they are caused by real errors and just have the wrong classification. Possibly the module which should find this type of error will also find those errors and the correct classification will also be available. It is also possible to steer the machine learner towards high precision (few false alarms).

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

Another interesting and useful result is that the automatically learned rules complement the manually constructed rules. This means that they do not find the same errors, so combining the two methods to achieve better results than each individual method is possible.

## 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

(Arppe 00) Antti Arppe. Developing a grammar checker for Swedish. In T. Nordgård, editor, *Proceedings of Nodalida '99*, pages 13–27. Trondheim, Norway, 2000.

(Atwell 87) Eric Steven Atwell. How to detect grammatical errors in a text without parsing it. In *Proceedings of the 3rd EACL*, pages 38–45, Copenhagen, Denmark, 1987.

(Bigert & Knutsson 02) Johnny Bigert and Ola Knutsson. 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, 2002.

(Birn 00) Juhani Birn. Detecting grammar errors with lingsoft's Swedish grammar checker. In T. Nordgård, editor, *Proceedings of Nodalida '99*, pages 28–40. Trondheim, Norway, 2000.

(Brants 00) Thorsten Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, pages 224–231, Seattle, USA, 2000.

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

(Domeij *et al.* 94) Rickard Domeij, Joachim Hollman, and Viggo Kann. Detection of spelling errors in Swedish not using a word list en clair. *Journal of Quantitative Linguistics*, 1:195–201, 1994.

(Domeij *et al.* 00) Richard Domeij, Ola Knutsson, Johan Carlberger, and Viggo Kann. Granska – an efficient hybrid system for Swedish grammar checking. In *Proceedings of Nodalida '99*, pages 49–56, Trondheim, Norway, 2000.

(Ejerhed *et al.* 92) Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Magnus Åström. 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, 1992.

(Gellerstam *et al.* 00) Martin Gellerstam, Yvonne Cederholm, and Torgny Rasmark. The bank of Swedish. In *Proceedings of LREC 2000*, pages 329–333, Athens, Greece, 2000.

(Golding 95) Andrew Golding. A bayesian hybrid for context sensitive spelling correction. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 39–53, Cambridge, USA, 1995.

(Hammarberg 77) Björn Hammarberg. Svenskan i ljuset av invandrares språkfel. *Nysvenska studier*, 57:60–73, 1977.

(Han *et al.* 04) Na-Rae Han, Martin Chodorow, and Claudia Leacock. Detecting errors in english article usage with a maximum entropy classifier trained on a large, diverse corpus. In *Proceedings of LREC-2004*, pages 1625–1628, Lisbon, Portugal, 2004.

(Hardt 01) Daniel Hardt. Transformation-based learning of Danish grammar correction. In *Proceedings of RANLP 2001*, Tzigov Chark, Bulgaria, 2001.

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

(Kann *et al.* 01) Viggo Kann, Rickard Domeij, Joachim Hollman, and Mikael Tillenius. 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, 2001.

(Mangu & Brill 97) Lidia Mangu and Eric Brill. Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning*, pages 187–194, 1997.

(Ngai & Florian 01) Grace Ngai and Radu Florian. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, pages 40–47, Carnegie Mellon University, Pittsburgh, USA, 2001.