Memory-based grammatical error correction

Antal van den Bosch
Radboud University Nijmegen
P.O. Box 9103
NL-6500 HD Nijmegen, The Netherlands
a.vandenbosch@let.ru.nl

Peter Berck
Tilburg University
P.O. Box 90153
NL-5000 LE Tilburg, The Netherlands
p.j.berck@tilburguniversity.edu

Abstract

We describe the 'TILB' team entry for the CONLL-2013 Shared Task. Our system consists of five memory-based classifiers that generate correction suggestions for center positions in small text windows of two words to the left and to the right. Trained on the Google Web 1T corpus, the first two classifiers determine the presence of a determiner or a preposition between all words in a text. The second pair of classifiers determine which is the most likely correction of an occurring determiner or preposition. The fifth classifier is a general word predictor which is used to suggest noun and verb form corrections. We report on the scores attained and errors corrected and missed. We point out a number of obvious improvements to boost the scores obtained by the system.

1 Introduction

Our team entry, known under the abbreviation 'TILB' in the CONLL-2013 Shared Task, is a simplistic text and grammar correction system based on five memory-based classifiers implementing eight different error correctors. The goal of the system is to be lightweight: simple to set up and train, fast in execution. It requires a preferably very large but unannotated corpus to train on, and closed lists of words that contain categories of interest (in our case, determiners and prepositions). The error correctors make use of information from a lemmatizer and a noun and verb inflection module. The amount of explicit grammatical information input in the system is purposely kept to a minimum, as accurate deep grammatical information cannot be assumed to be present in most real-world situations and languages. The system described in this article takes plain text as input and produces plain text as output.

Memory-based classifiers have been applied to similar tasks before. (Van den Bosch, 2006) describes memory based classifiers used for confusible disambiguation, and (Stehouwer and Van den Bosch, 2009) shows how agreement errors can be detected. In the 2012 shared task 'Helping Our Own' (Dale et al., 2012) memory based classifiers were used to solve the problem of missing and incorrect determiners and prepositions (Van den Bosch and Berck, 2012).

The CONLL-2013 Shared Task context limited the grammatical error correction task to detecting and correcting five error types:

ArtOrDet	Missing, unnecessary or incorrect article or determiner;
Prep	Incorrect preposition used;
Nn	Wrong form of noun used (e.g. singular instead of plural);
Vform	Incorrect verb form used (e.g. *I have went*);
SVA	Incorrect subject-verb agreement (e.g. *He have*).

The corrections made by the system are scored by a program provided by the organizers (Ng, 2012). It takes a plain textfile as input (the output generated by the system) and outputs a list with correctly rectified errors followed by precision, recall and F-score.

As training material we used two corpora. The Google Web 1T corpus (Brants and Franz, 2006) was used to train the classifiers for the ArtOrDet and Prep error categories. The GigaWord Newspaper text corpus[1] was used to create the data for the classifier for the noun and verb-related error categories. To make the classifiers more compatible

[1] http://www.ldc.upenn.edu/

Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task, pages 102–108,
Sofia, Bulgaria, August 8-9 2013.

with each other, future versions of the system will all be trained on the same corpus. We also used two lists, one consisting of 64 prepositions and one consisting of 23 determiners, both extracted from the CONLL-2013 Shared Task training data. Using the Google corpus means that we restricted ourselves to a simple 5-gram context, which obviously places a limit on the context sensitivity of our system; on the other hand, we were able to make use of the entire Google Web 1T corpus. The context for the grammatical error detectors was kept similar to the other classifiers, also 5-grams.

2 System

Our system is based on five memory-based classifiers that all run the IGTree classifier algorithm (Daelemans et al., 1997), a decision-tree approximation of k-nearest neighbour classification implemented in the TiMBL software package.[2] The first two classifiers determine the presence of a determiner or a preposition between all words in a text in which the actual determiners and prepositions are masked. The second pair of classifiers determine which is the most likely correction given a masked determiner or preposition. The fifth classifier is a general word predictor that is used for suggesting noun and verb form corrections.

All classifiers take a windowed input of two words to the left of the focus position, and two words to the right. The focus may either be a position *between* two words, or be *on* a word. In case of a position between two words, the task is to predict whether the position should actually be filled by an determiner or a preposition. When the focus is on the word in question, the task is to decide whether it should be deleted, or whether it should be corrected.

It is important to note that not just one classification is returned for a given context by the IGTree classifier, but a *distribution* of results and their respective occurrence counts. The classifier matches the words in the context to examples in the tree in a fixed order, and returns the distribution stored at that point in the tree when an unknown word is encountered. This is analogous to the back-off mechanisms often used in other n-gram based language modeling systems. When even the first feature fails to match, the complete class distribution is returned. The output from the classifiers is filtered by the error correctors for the correct answers. Filtering is done based on distribution size, occurrence counts and ratios in occurrence counts (in the remainder of the text, where we say frequency we mean occurrence count), and in the case of the noun and verb-related error types, on part-of-speech tags.

The system corrects a text from left to right, starting with the first word and working its way to the end. Each error corrector is tried after the other, in the order specified below, until a correction is suggested. At this point, the correction is stored, and the system starts processing the next word. The other classifiers are not tried anymore after a correction has been suggested by one of the classifiers.

The first two classifiers, **preposition?** and **determiner?**, are binary classifiers that determine whether or not there should be a preposition or a determiner, respectively, between two words to the left and two words to the right:

- The **preposition?** classifier is trained on all 120,711,874 positive cases of contexts in the Google Web 1T corpus in which one of the 64 known prepositions are found to occur in the middle position of a 5-gram. To enable the classifier to answer negatively to other contexts, roughly the same amount of negative cases of randomly selected contexts with no preposition in the middle are added to form a training set of 238,046,975 cases. We incorporate the Google corpus token counts in our model. We performed a validation experiment on a single 90%-10% split of the training data; the classifier is able to make a correct decision on 88.6% of the 10% heldout cases.

- Analogously, the **determiner?** classifier takes all 86,253,841 positive cases of 5-grams with a determiner in the middle position, and adds randomly selected negative cases to arrive at a training set of 169,874,942 cases. On a 90%–10% split, the classifier makes the correct decision in 90.0% of the 10% heldout cases.

The second pair of classifiers perform the multi-label classification task of predicting which preposition or determiner is most likely given a context of two words to the left and to the right. Again,

[2] http://ilk.uvt.nl/timbl

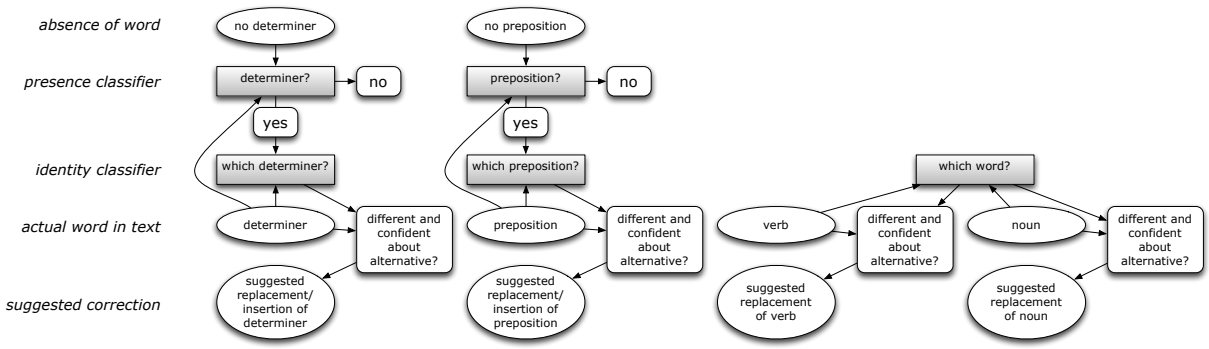

Figure 1: System architecture. Shaded rectangles are the five classifiers.

these classifiers are trained on the entire Google Web 1T corpus, including its token counts:

- The **which preposition?** classifier is trained on the aforementioned 120,711,874 cases of any of the 64 prepositions occurring in the middle of 5-grams. The task of the classifier is to generate a class distribution of likely prepositions given an input of the four words surrounding the preposition, with 64 possible outcomes. In a 90%-10% split experiment on the complete training set, this classifier labels 63.3% of the 10% heldout cases correctly.
- The **which determiner?** classifier, by analogy, is trained on the 86,253,841 positive cases of 5-grams with a determiner in the middle position, and generates class distributions composed of the 23 possible class labels (the possible determiners). On a 90%-10% split of the training set, the classifier predicts 68.3% of all heldout cases correctly.

The fifth classifier predicts the most likely word(s) between a context of two words to the left and two to the right.

- The general word predictor, **which word?**, for the grammatical error types, was trained on 10 million lines of the GigaWord English Newspaper corpus. This amounts to 66,675,151 5-grams. It predicts the word in the middle between the two context words on the left and on the right.

From the predictions of the five classifiers the following eight error correctors are derived. There is no one-to-one correspondence between classifier and corrector. The ArtOrDet and Prep error categories are handled by three separate errors correctors each that handle replacement, deletion, and insertion errors. The three error types Nn, Vform and SVA are handled by just two correctors:

1 missing preposition (Prep)
2 replace preposition (Prep)
3 unnecessary preposition (Prep)
4 missing determiner (ArtOrDet)
5 replace determiner (ArtOrDet)
6 unnecessary determiner (ArtOrDet)
7 noun form (Nn, SVA)
8 verb form (Vform, SVA)

For the latter two error correctors, 7 and 8, we make additional use of a lemmatizer[3] and a singular-plural determiner and generator[4] for noun form errors, and a verb tense determiner and generator[5] for verb form and SVA errors.

The algorithms for the six preposition and determiner correctors will be explained in the rest of this section. The algorithms use the same logic, the difference is in the different lists and parameters used for each error type.

The algorithm for missing preposition (or determiner) is as follows.

1 next word is not a preposition
2 run **positive-negative classifier** P^{+}_{-}
3 if the classification $= +$ (i.e. we expect a preposition), and freq(+):freq($-$) $>$ MP_PNR
4 run the **which preposition?** classifier
5 if length distribution $<=$ MP_DS take answer as missing preposition

The parameters (MP_PNR and MP_DS in the above algorithm) are used to control the certainty we expect from the classifier. Their values were determined in our submission to the 2012 'Helping

[3] http://www-nlp.stanford.edu/software/corenlp.shtml
[4] https://pypi.python.org/pypi/inflect
[5] http://nodebox.net/code/index.php/Linguistics

Our Own' shared task (Dale et al., 2012), which focused on determiner and preposition errors (Van den Bosch and Berck, 2012). Similar classifiers were used in this year's system, and the same parameters were used this time.

In step 3 above, we check the ratio between the frequency of the positive answer and the negative answer. If the ratio is larger than the parameter MP_PNR (set to 20) we interpret this as being certain. In step 5, we prefer a small, sharp distribution of answers. A large distribution indicates the classifier not finding any matches in the context and returning a large distribution with all possible answers. In that case, the majority class tends to be the majority class of the complete training data, and not the specific answer(s) in the context we are looking at. To avoid this we only suggest an answer when the distribution is equal to or smaller than a certain preset threshold, MP_DS, which was set to 20 for this task.

The algorithm for replacing propositions (or determiners) proceeds as follows:

1. word in focus is a preposition p
2. run **which preposition?**, classification is p_{alt}
3. if freq(p_{alt}) $>$ RP_F and
4. if word is in distribution and freq(p_{alt}):freq(p) $>$ RP_R, take p_{alt} as a correction

This algorithm shows another parameter, namely a check on frequency (occurrence count). In order to be generated as a correction, the alternate answer must have a frequency higher than RP_F, set to 5 in our system, and the ratio between its frequency and that of the preposition in the distribution that is the same as in the text must be larger than RP_R. This parameter was set to 20.

The algorithm for unnecessary preposition (or determiner) works as follows:

1. word in focus is a preposition
2. run **positive-negative** classifier P^{+}_{-}
3. if classification $= -$ and freq($-$):freq($+$) $>$ UP_NPR
4. the preposition is unnecessary

The next two algorithms show the Nn and Vf correctors. The parameters these correctors use have not been extensively tweaked, but rather use the same settings as used in the preposition and determiner correctors.

The first list shows the algorithm for the noun type error. This error corrector also makes use of a noun inflection module to turn singular nouns into plural and vice versa. The algorithm first looks for the alternative version of the noun in the distribution returned by the classifier given the context. If it is found, and if it is much more frequent in the distribution than the noun form used in the text, a noun form error may have been found. The alternative form found in the distribution is returned as the correction.

1. word in focus w is a noun
2. check singular or plural, determine alternate version w_{alt}
3. run the **which-word?** classifier, resulting in distribution D
4. check if w is in D
5. check if w_{alt} is in D
6. if freq(w) in D < 10 and w_{alt} is in D use w_{alt} as correction

Finally, the verb form error corrector makes use of a verb-tense determiner and generator, and a lemmatizer. The alternative verb forms are generated from the lemma of the verb and the tense of the verb. To prevent the system changing, for example, *give* to *gave*, the generated alternatives are kept in the same tense as the word in the text. This does, however, mean that it will not be able to correct verb tense errors (*I see him yesterday* versus *I saw him yesterday*).

1. word in focus is a verb v
2. determine the lemma of v
3. determine the tense of v
4. generate alternatives in same tense as word, v_{alt}
5. run **which-word?** predictor, resulting in distribution D
6. check if v is in D
7. check which v_{alt} are in D, take highest frequency freq(v_{alt})
8. if freq(v_{alt}):freq(v) > 10: take v_{alt} as a correction of v

3 Results

Table 1 lists the precision, recall and F-score of our system on the test data. The test data (Tetreault, 2013) consisted of 300 paragraphs of English text written by non-native speakers. The system's output is processed by a scorer supplied by the organizers (Ng, 2012). For each sentence, it reports the number of correct, proposed and gold edits, and a running total of the system's precision, recall and F-score.

The system suggested a total of 1,902 edits. Of these, 118 were correct. The total number of correct edits was 1,643. To explain the score obtained by the system, we inspect the kind of errors which it was subjected to, and what kind of errors it did correct and which it missed.

Precision	6.20%
Recall	7.18%
F1	6.66%

Table 1: Summary Score

We see a number of errors which are difficult to correct because they depend on understanding the sentence. Take the following sentence for example:

Surveillance technology such as RFID can be operated twenty-four hours with the absence of operators to track done every detail about human activities .

The gold-edit for this sentence is changing *with* (word 11) to *without*. This edit may be questionable, but questionability aside, it is based on a understanding of what is being talked about in the text. Correcting these kinds of errors falls outside the scope of the system at the moment.

Multi-word edits are also a problem. In *All passengers and pilots were died*, the gold-edit is to change *were died* to *died*. In *The readers are just smiling when they flip the page because it never comes to their mind that one day it might come true*, the gold-edit is to change *are just smiling* to *just smile*. These kind of corrections are missed by our system at the moment due to the rigid one-word, left-to-right checking of the sentence.

Inserting more than one word is also problematic for our system at the moment. Take the following sentence.

Firstly , security systems are improved in many areas such as school campus or at the workplace .

The gold-edit is to insert *on the* before *school*. A potential solution for this problem is to take multiple passes over the sentence, first inserting *on*, followed by *the* in a later pass.

Nevertheless, the system made a number of correct edits as well. The next subsections list examples of each error type and a correction, where applicable.

Missing determiner

In this sentence, the missing determiner before *smart* was corrected by the system.

In spite of that, ***the smart*** *phone is still a device . . .*

In the following sentence however, a determiner is inserted where it is not needed, before *RFID*.

. . . the idea of using ***the RFID*** *to track people . . .*

To illustrate the reasoning of our system, the **determiner?** classifier thinks that it is more than 13 times more likely to find a determiner between *of using* and *RFID to* than not. Of the possible determiners, the determiner *the* has the highest frequency with 38,809 occurrences.

Replace Determiner

Here is an example of a determiner which is corrected:

. . . signal and also ***a⇒the*** *risk that their phone . . .*

It also happens that the right determiner is incorrectly changed into another determiner, as shown in the next example.

. . . this kind of tragedy to happen on ***any⇒the*** *family.*

The determiner *the* had a frequency of more than 6 million in the distribution, compared to only 68,612 for *any*.

Unnecessary Determiner

The system did not detect any unnecessary determiners. It missed, for example, removing the determiner *the* in this setentence:

. . . technology available for the Man 's life .

Replace Preposition

In this example, a preposition was corrected.

. . . to be put ***into⇒under*** *close surveillance . . .*

But in the following sentence

. . . remain functional ***for⇒after*** *a long period of . . .*

the preposition *for* is unfortunately changed to *after*, which in this context is more common.

Unnecessary Preposition

The following is an example of a correct removal of a preposition:

. . . , many ***of*** *things that are regarded . . .*

Prepositions were also incorrectly removed, as shown in the following example. Here

. . . that can be out ***of*** *our imaginations . . .*

of is deemed unnecessary.

Missing Preposition

In this example, the missing preposition *on* was inserted after *live*.

. . . find another planet to live ***on*** *, the earth is . . .*

In the sentence

. . . especially ***in*** *the elderly and the children . . .*

the system inserts the preposition *in* between *especially* and *and*, which in this case was incorrect.

Noun form

The next example shows a noun form correction.

. . . brought harmful side ***effect⇒effects*** *to human body*

This can, of course, also go wrong:

Since RFID ***tags⇒tag*** *attached to the product . . .*

Here the singular form of the noun was deemed correct.

Verb form

Finally, an example of a verb form correction:

People ***needs⇒need*** *a safe environment to live . . .*

And the final example, an incorrect replacement of *been* to *was*.

. . . that has currently been⇒was implemented

4 Discussion

We have described a memory-based grammar checker specialized in correcting the five types of errors in the CoNLL-2013 Shared Task. The system is built on five classifiers specialized in the error categories relevant for the task. They are trained to find errors in a small local context of two words to the left and two words to the right. The system scans each word in each sentence in the test data and calls the relevant classifier(s) to determine if a word needs to be replaced, deleted, or inserted. The classifiers take word tokens as input; no deep grammatical information was supplied to them. Even though the training data supplied for the task contained syntax trees, they were not used in creating our system. On the other hand, the part-of-speech information in the training data was used to create the lists of prepositions and determiners. Furthermore, a part-of-speech tagger was used to determine if the noun or verb form error corrector was to be applied.

There are several obvious shortcomings to this approach. The most obvious one is that each corrector is applied to single words, using only a small local context of two words to the left and right. This may work fine for missing prepositions and determiners, but for spotting grammatical errors like subject-verb agreement this limited contextual scope is insufficient. It also means that we are only able to correct "single words to single words". That is, it is not possible to substitute two words for one, and vice versa. One avenue that could be explored is larger contexts. In addition, the classifiers are not limited to words, and contexts with other (contextual) information could be tried as well.

Secondly, the correctors are applied in a strict order one after the other. This should not be a big problem as the classifiers are called separately for their particular part-of-speech category (determiner, preposition, verb, or noun). On the other hand, this puts a lot of weight on the part of speech tagger. Ambiguous or wrong tags could cause the wrong corrector to be tried and even applied, and could miss a potential correct correction.

Furthermore, the corrected words are not fed back into the system. This means that the context after an error still contains that error. This may cause the classifiers to mismatch and miss the next error. It should be noted that the small context of two words to the left and right probably helps to alleviate this problem. However, making the system insert corrections and backtracking a step (or more) could help towards solving the problem of multi-word corrections.

Finally, not all correctors found errors. This may of course depend on the test data, but it seems unlikely that the data contained no 'missing preposition' errors. There is a potential gain in tuning

the parameters controlling the error correctors.

4.1 Update

The organizers of the shared task updated the m2-scorer used to calculate the results, resulting in slightly better scores. Table 2 shows the revised score of our system, with the old score between parentheses.

Precision	7.60%	(6.20%)
Recall	9.29%	(7.18%)
F1	8.36%	(6.66%)

Table 2: Revised Summary Score

And to conclude, we continued working on the system and tweaked some of the parameters controlling the preposition and determiner checkers. By allowing the correctors to be applied more often, we see an increase in the number of proposed and correct edits (2,533 and 178 respectively). The downside to this is of course that the number of false positives increases, which decreases the precision of the system.

The tweaked score is shown in table 3, with the revised score between parentheses.

Precision	7.03%	(7.60%)
Recall	10.83%	(9.29%)
F1	8.52%	(8.36%)

Table 3: Tweaked Summary Score

These improved scores give us good hope that the highest scores have not been reached yet.

Acknowledgements

The authors thank Ko van der Sloot for his sustained improvements of the TiMBL software. This work is rooted in earlier joint work funded through a grant from the Netherlands Organization for Scientific Research (NWO) for the Vici project *Implicit Linguistics*.

References

T. Brants and A. Franz. 2006. LDC2006T13: Web 1T 5-gram Version 1.

W. Daelemans, A. Van den Bosch, and A. Weijters. 1997. IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

R. Dale, I. Anisimoff, and G. Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, Montreal, Canada.

Daniel Dahlmeier & Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 568 – 572.

H. Stehouwer and A. Van den Bosch. 2009. Putting the t where it belongs: Solving a confusion problem in Dutch. In S. Verberne, H. van Halteren, and P.-A. Coppen, editors, *Computational Linguistics in the Netherlands 2007: Selected Papers from the 18th CLIN Meeting*, pages 21–36, Nijmegen, The Netherlands.

Hwee Tou Ng & Siew Mei Wu & Yuanbin Wu & Christian Hadiwinoto & Joel Tetreault. 2013. The conll-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning.*

A. Van den Bosch and P. Berck. 2012. Memory-based text correction for preposition and determiner errors. In *Proceedings of the 7th Workshop on the Innovative Use of NLP for Building Educational Applications*, pages 289–294, New Brunswick, NJ. ACL.

A. Van den Bosch. 2006. All-word prediction as the ultimate confusible disambiguation. In *Proceedings of the HLT-NAACL Workshop on Computationally hard problems and joint inference in speech and language processing*, New York, NY.