Memory-based text correction for preposition and determiner errors

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 Valkuil.net team entry for the HOO 2012 Shared Task. Our systems consists of four memory-based classifiers that generate correction suggestions for middle positions in small text windows of two words to the left and to the right. Trained on the Google 1TB 5-gram corpus, 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 determines which is the most likely correction given a masked determiner or preposition. The hyperparameters that govern the classifiers are optimized on the shared task training data. We point out a number of obvious improvements to boost the medium-level scores attained by the system.

1 Introduction

Our Valkuil.net team entry, known under the abbreviation 'VA' in the HOO 2012 Shared Task (Dale et al., 2012), is a simplistic text correction system based on four memory-based classifiers. The goal of the system is to be lightweight: simple to set up and train, fast in execution. It requires a (preferably very large) corpus to train on, and a closed list of words which together form the category of interest—in the HOO 2012 Shared Task context, the two categories of interest are prepositions and determiners.

As a corpus we used the Google 1TB 5-gram corpus (Brants and Franz, 2006), and we used two lists, one consisting of 47 prepositions and one consisting of 24 determiners, both extracted from the HOO 2012 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; yet, we were able to make use of the entire Google corpus.

Memory-based classifiers have been used for confusible disambiguation (Van den Bosch, 2006) and agreement error detection (Stehouwer and Van den Bosch, 2009).[1] In both studies it is argued that fast approximations of memory-based discriminative classifiers are effective and efficient modules for spelling correction, particularly because of their insensitivity to the number of classes to be predicted. They can act as simple binary decision makers (e.g. for confusible pairs: given this context, is *then* or *than* more likely?), and at the same time they can handle missing word prediction with up to millions of possible outcomes, all in the same model. Van den Bosch (2006) also showed consistent log-linear performance gains in learning curve experiments, indicating that more training data continues to be better for these models even at very large amounts of training data. The interested reader is referred to the two studies for more details.

2 System

Our system centers around four classifiers that all take a windowed input of two words to the left of the focus, and two words to the right. The focus may either be a position between two words, or a determiner or a preposition. In case of a position

[1]A working context-sensitive spelling checker for Dutch based on these studies is released under the name Valkuil.net; see `http://valkuil.net` – hence the team name.

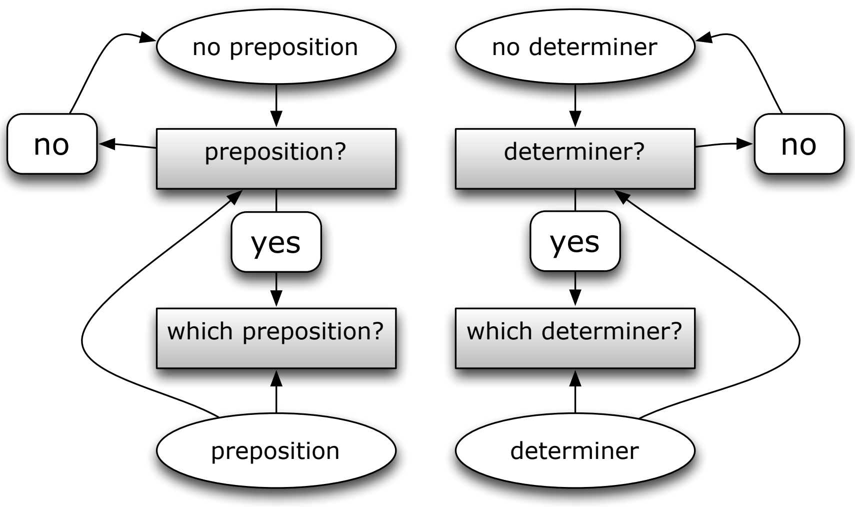

Figure 1: System architecture. Shaded rectangles are the four classifiers.

between two words, the task is to predict whether the position should actually be filled by a preposition or a determiner. When the focus is on a determiner or preposition, the task may be to decide whether it should actually be deleted, or whether it should be replaced.

The main system architecture is displayed in Figure 1. The classifiers are the shaded rectangular boxes. They are all based on IGTree, an efficient decision tree learner (Daelemans et al., 1997), a fast approximation of memory-based or k-nearest neighbor classification, implemented within the TiMBL[2] software package (Daelemans et al., 2010).

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 118,105,582 positive cases of contexts in the Google 1 TB 5-gram corpus in which one of the 47 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 235,730,253 cases. In the participating system we take each n-gram as a single token, and ignore the Google corpus token counts. 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 89.1% of the 10% heldout cases.
- Analogously, the **determiner?** classifier takes all 132,483,802 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 252,634,322 cases. On a 90%–10% split, the classifier makes the correct decision in 88.4% 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, these classifiers are trained on the entire Google 1TB 5-gram corpus:

- The **which preposition?** classifier is trained on the aforementioned 118,105,582 cases of any of the 47 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 47 possible outcomes. In a 90%-10% split experiment on the complete training set, this classifier labels 59.6% of the 10% heldout cases correctly.

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

- The **which determiner?** classifier, by analogy, is trained on the 132,483,802 positive cases of 5-grams with a determiner in the middle position, and generates class distributions composed of the 24 possible class labels (the possible determiners). On a 90%-10% split of the training set, the classifier predicts 63.1% of all heldout cases correctly.

Using the four classifiers and the system architecture depicted in Figure 1, the system is capable of detecting missing and unnecessary cases of prepositions and determiners, and of replacing prepositions and determiners by other more likely alternatives. Focusing on the preposition half of the system, we illustrate how these three types of error detection and correction are carried out.

First, Figure 2 illustrates how a missing preposition is detected. Given an input text, a four-word window of two words to the left and two words to the right is shifted over all words. At any word which is not in the list of prepositions, the binary **preposition?** classifier is asked to determine whether there should be a preposition in the middle. If the classifier says no, the window is shifted to the next position and nothing happens. If the classifier says yes beyond a certainty threshold (more on this in Section 3), the **which preposition?** classifier is invoked to make a best guess on which preposition should be inserted.

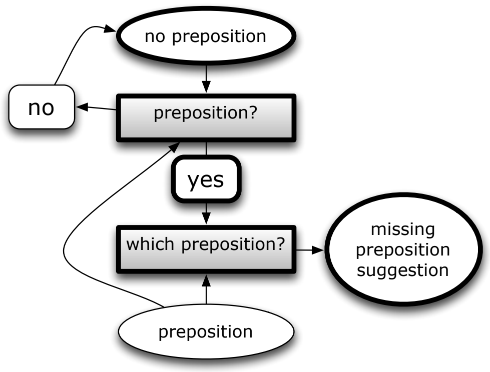

Figure 2: Workflow for detecting a missing preposition.

Second, Figure 3 depicts the workflow of how a preposition deletion is suggested. Given an input text, all cases of prepositions are sought. Instances of two words to the left and right of each preposition are created, and these context windows are presented to the **preposition?** classifier. If this classifier says no beyond a certainty threshold, the system signals that the preposition currently in focus should be deleted.

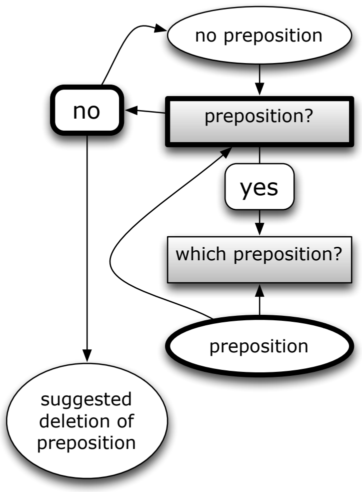

Figure 3: Workflow for suggesting a preposition deletion.

Third, Figure 4 illustrates how a replacement suggestion is generated. Just as with the detection of deletions, an input text is scanned for all occurrences of prepositions. Again, contextual windows of two words to the left and right of each found preposition are created. These contexts are presented to the **which preposition?** classifier, which may produce a different most likely preposition (beyond a certainty threshold) than the preposition in the text. If so, the system signals that the original preposition should be replaced by the new best guess.

Practically, the system is set up as a master process (implemented in Python) that communicates with the four classifiers over socket connections. The master process performs all necessary data conversion and writes its edits to the designated XML format. First, missing prepositions and determiners are traced according to the procedure sketched above; second, the classifiers are employed to find replacement errors; third, unnecessary determiners and prepositions are sought. The system does not iterate over its own output.

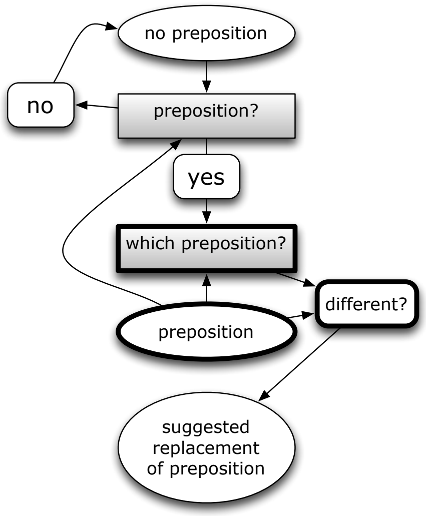

Figure 4: Workflow for suggesting a preposition replacement.

Task	Thresh.	Optimizing on Precision	Recall	F-Score
Prep.	M	30	10	20
	U	30	4	4
	DS	5	50	50
	F	50	5	5
	R	10	20	20
Det.	M	30	10	20
	U	30	2	2
	DS	5	50	20
	F	50	5	20
	R	10	20	20

Table 1: Semi-automatically established thresholds that optimize precision, recall, and F-Score. Optimization was performed on the HOO 2012 Shared Task training data.

3 Optimizing the system

When run unfiltered, the four classifiers tend to over-predict errors massively. They are not very accurate (the binary classifiers operate at a classification accuracy of 88–89%; the multi-valued classifiers perform at 60–63%). On the other hand, they produce class distributions that have properties that could be exploited to filter the classifications down to cases where the system is more certain. This enables us to tune the precision and recall behavior of the classifiers, and, for instance, optimize on F-Score. We introduce five hyperparameter thresholds by which we can tune our four classifiers.

First we introduce two thresholds for the two binary classifiers **preposition?** and **determiner?**:

M — When the two binary **preposition?** and **determiner?** classifiers are used for detecting missing prepositions or determiners, the positive class must be M times more likely than the negative class.

U — In the opposite case, when the two binary classifiers are used for signalling the deletion of an unnecessary preposition or determiner, the negative class must be U times more likely than the positive class.

For the two multi-label classifiers **which preposition?** and **which determiner?** we introduce three thresholds (which again can be set separately for determiners and prepositions):

DS — the distribution size (i.e. the number of labels that have a non-zero likelihood according to the classifier) must be smaller than DS. A large DS signals a relatively large uncertainty.

F — the frequency of occurrence of the most likely outcome in the training set must be larger than F. Outcomes with a smaller number of occurrences should be distrusted more.

R — if the most likely outcome is different from the preposition or determiner currently in the text, the most likely outcome should be at least R times more likely than the current preposition or determiner. Preferably the likelihood of the latter should be zero.

On the gold training data provided during the training phase of the HOO 2012 Shared Task we found, through a semi-automatic optimization procedure, three settings that optimized precision, recall, and F-Score, respectively. Table 3 displays the optimal settings found. The results given in Section 4 always refer to the system optimized on F-Score, listed in the rightmost column of Table 3.

The table shows that most of the ratio thresholds found to optimize F-Score are quite high; for example, the **preposition?** classifier needs to assign a likelihood to a positive classification that is at least 20 times more likely than the negative classification in order to trigger a missing preposition error. The threshold for marking unnecessary prepositions is considerably lower at 4, and even at 2 for determiners.

4 Results

The output of our system on the data provided during the test phase of the HOO 2012 Shared Task was processed through the shared task evaluation software. The original test data was revised in a correction round in which a subset of the participants could suggest corrections to the gold standard. We did not contribute suggestions for revisions, but our scores slightly improved after revisions. Table 4 summarizes the best scores of our system optimized on F-Score, before and after revisions. Our best score is an overall F-Score of 14.24 on error detection, after revisions. Our system performs slightly better on prepositions than on determiners, although the differences are small. Optimizing on F-Score implies that a reasonable balance is found between recall and precision, but overall our results are not impressive, especially not in terms of correction.

5 Discussion

We presented a preposition and determiner error detection and correction system, the focus task of the HOO 2012 Shared Task. Our system consists of four memory-based classifiers and a master process that communicates with these classifiers in a simple workflow. It takes several hours to train our system on the Google 1TB 5-gram corpus, and it takes in the order of minutes to process the 1,000 training documents. The system can be trained without needing linguistic knowledge or the explicit computation of linguistic analysis levels such as POS-tagging or syntactic analyses, and is to a large extent language-independent (it does rely on tokenization).

This simple generic approach leads to mediocre results, however. There is room for improvement. We have experimented with incorporating the n-gram counts in the Google corpus in our classifiers, leading to improved recall (post-competition). It still remains to be seen if the Google corpus is the best corpus for this task, or for the particular English-as-a-second-language writer data used in the HOO 2012 Shared Task. Another likely improvement would be to limit which words get corrected by which other words based on confusion statistics in the training data: for instance, the training data may tell that 'my' should rarely, if ever, be corrected into 'your', but our system is blind to such likelihoods.

Acknowledgements

The authors thank Ko van der Sloot for his continued 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.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2010. TiMBL: Tilburg memory based learner, version 6.3, reference guide. Technical Report ILK 10-01, ILK Research Group, Tilburg University.

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.

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.

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.

		Before revisions			After revisions		
Task	Evaluation	Precision	Recall	F-Score	Precision	Recall	F-Score
Overall	Detection	12.5	15.23	13.73	13.22	15.43	14.24
	Recognition	10.87	13.25	11.94	11.59	13.53	12.49
	Correction	6.16	7.51	6.77	7.25	8.46	7.8
Prepositions	Detection	13.44	14.41	13.91	14.23	14.75	14.49
	Recognition	11.46	12.29	11.86	12.65	13.11	12.88
	Correction	7.51	8.05	7.77	8.7	9.02	8.85
Determiners	Detection	11.04	15.21	12.79	11.71	15.28	13.26
	Recognition	10.37	14.29	12.02	10.7	13.97	12.12
	Correction	5.02	6.91	5.81	6.02	7.86	6.82

Table 2: Best scores of our system before (left) and after (right) revisions. Scores are reported at the overall level (top), on prepositions (middle), and determiners (bottom).