A Naive Bayes classifier for automatic correction of preposition and determiner errors in ESL text

Gerard Lynch, Erwan Moreau and Carl Vogel
Centre for Next Generation Localisation
Integrated Language Technology Group
School of Computer Science and Statistics
Trinity College Dublin, Ireland
`gplynch,moreaue,vogel@scss.tcd.ie`

Abstract

This is the report for the CNGL ILT team entry to the HOO 2012 shared task. A Naive-Bayes-based classifier was used in the task which involved error detection and correction in ESL exam scripts. The features we use include n-grams of words and POS tags together with features based on the external Google N-Grams corpus. Our system placed 11th out of 14 teams for the detection and recognition tasks and 11th out of 13 teams for the correction task based on F-score for both preposition and determiner errors.

1 Introduction

The HOO 2012 shared task seeks to apply computational methods to the correction of certain types of errors in non-native English texts. The previous year's task, (Dale and Kilgarriff, 2011), focused on a larger scale of errors and a corpus of academic articles. This year's task focuses on six error types in a corpus of non-native speaker text. The scope of the errors is as follows:[1]

Error Code	Description	Example
RT	Replace Preposition	*When I arrived at London*
MT	Missing preposition	*I gave it John*
UT	Unnecessary preposition	*I told to John that*
RD	Replace determiner	*Have the nice day*
MD	Missing determiner	*I have car*
UD	Unnecessary determiner	*There was a lot of the traffic*

Table 1: Error types for HOO 2012 Shared Task

In Section 2, we give a brief summary of the data for the shared task and in Section 3 we explain the individual steps in the system. Section 4 details the different configurations for each of the runs submitted and finally, Section 5 presents the results.

2 Training data

The training data for this shared task has been provided by Cambridge University Press and consists of scripts from students sitting the Cambridge ESOL First Certificate in English (FCE) exams. The topics of the texts are comparable as they have been drawn from two consecutive exam years. The data is provided in XML format and contains 1000 original exam scripts, together with a standoff file containing edits of the type described in Section 1 above, also in XML format. These edits consist of offset information, edit type information and before and after text for correction. The results for the shared task were presented in this format.

The test data consists of 100 exam scripts drawn from a new corpus of exam scripts.

Some extra metadata is present in the source files, including information about the student's mother tongue and the age-range of the student, however the mother tongue data is not present in the test set.

3 Approach

The approach we have chosen for this task involves the use of supervised machine-learning algorithms in a four-part classification task.

3.1 Overview of the system

The first part of the task involves *identification* of edits in the training data, perhaps the most challeng-

[1] `http://correcttext.org/hoo2012/errortypes.html` last verified, May 10, 2012

The 7th Workshop on the Innovative Use of NLP for Building Educational Applications, pages 257–262,
Montréal, Canada, June 3-8, 2012. ©2012 Association for Computational Linguistics

ing given the large imbalance of edits vs non-edits in the data.

The next step concerns *classification* of edits into the six types described above, and the final task involves *correction* of edits, replacing or adding prepositions and determiners, and possibly in some cases removal of same.

There is a fourth step involved which reassesses the classification and correction based on some simple heuristics, using POS tags of the head word of each instance. If the headword is not a preposition and the system has marked a replace preposition error at that position, this error will be removed from the system. Likewise when the headword is not a determiner and a replace determiner error has been marked. If the replacement suggested is the same as the original text (in some cases this occurs), the edit is also removed. Another case for removal in this fashion includes an error type involving a missing determiner error where the head word is neither a noun or an adjective. In some cases the system reported and corrected an error suggesting the same text as was originally there, i.e no change. These cases are also removed from the end result.

3.2 Classification

We utilise the freely available Weka machine learning toolkit (Hall et al., 2009), and the algorithm used for classification in each step is Naive Bayes.

3.2.1 Representing the data

We represent each word in the training data as a vector of features. There are 39 basic features used in the detection process, and 42 in the classification and training step. The first 7 features contain information which is not used for classification but is used to create the edit structures, such as start offset, end offset, native language, age group and source filename and part information. These features include the current word plus the four preceding and following words, POS and spell-checked versions of each, together with bigrams of the two following and two preceding words with spell-checked and POS versions for these. Information on speaker age and native language is also included although native language information is not present in the test set.

3.2.2 Additional processing

All tokens have been lower-cased and punctuation has been removed. POS information for each token has been added. The open-source POS tagger from the OpenNLP tools package (OpenNLP, 2012) has been used to this end. Spell correction facility has been provided using the basic spellchecker in the Lucene information retrieval API(Gospodnetic and Hatcher, 2005) and the top match string as provided by this spell correcting software is used in addition to each feature. The basic maximum entropy model for English is used for the POS tagger.

We had also planned to include features based on the Google Books n-gram corpus, (Michel et al., 2011) which is freely available on the web, but unfortunately did not get to include them in the version submitted due to errors which were found in the scripts for generating the features late in the process. Nevertheless, we describe these features in Section 3.3 and present some cross-validation results from the training data for the detection step in Section 5.1.

3.3 Google N-grams Features

3.3.1 Motivation

The Google Books N-Grams[2] is a collection of datasets which consist of all the sequences of words (*n-grams*) extracted from millions of books (Michel et al., 2011). The "English Million" dataset contains more more than 500 millions distinct n-grams[3], from size 1 to 5. for every n-gram, its frequency, page frequency (number of pages containing it) and book frequency (number of books containing it) are provided.

In this Shared Task, we aim to use the Google N-grams as a reference corpus to help detecting the errors in the input. The intuition is the following: if an error occurs, comparing the frequency of the input n-grams against the frequency of other possibilities in the Google N-grams data might provide useful indication on the location/type of the error. For example, given the input *"I had to go in a library"*, The Google N-grams contain only 36,716 occurrences of the trigram *"go in a"*, but 244,098 occurrences of *"go to a"*, which indicates that the latter is more likely.

[2] `http://books.google.com/ngrams/datasets`

[3] The least frequent n-grams were discarded.

However there are several difficulties in using such a dataset:

- *Technical limitations.* Extracting information from the dataset can take a lot of time because of the size of the data, thus the range of approaches is restricted by efficiency constraints.
- *Quality of the data.* The Google N-grams were extracted automatically using OCR, which means that the dataset can contain errors or unexpected data (for example, the English dataset contains a significant number of non-English words).

This is why the Google N-grams must be used cautiously, and only as an indication among others.

3.3.2 Method

Our goal is to add features extracted from the Google N-grams dataset to the features described above, and feed the supervised classification process with these. Before computing the features, a list L of "target expressions" is extracted from the training data, which contains all the words or sequences of words (determiners and prepositions) which occur in a correction. Then, given an input sentence $A_1 \ldots A_m$ and a position n in this sentence, two types of information are extracted from the Google data:

- Specific indications of whether an error exists at this position:
  1. No change: the frequency of the input sequence $A_{n-1}A_n$ and $A_{n-1}A_nA_{n+1}$;
  2. Unnecessary word(s): the frequency of the sequence $A_{n-1}A_{n+1}$ if $A \in L$;
  3. Missing word(s): the frequency of the sequence XA_n (resp. $A_{n-1}XA_n$ for trigrams) for any target expression $X \in L$;
  4. Replacement: if $A \in L$, the frequency of XA_{n+1} (resp. $A_{n-1}XA_{n+1}$ for trigrams) for any target expression $X \in L$;
- Generic indications taking the context into account: for length N from 1 to 5 in a window $A_{n-4} \ldots A_{n+4}$, 16 combinations are computed based only on the fact the n-grams appear in the Google data; for example, one of these combinations is the normalized sum for the 4 5-grams in this window of 0 or 1 (the n-gram occurs or does not).

Additionally, several variants are considered:

- bigrams or trigrams for "specific" features;
- binary values for "specific" features: 1 if the n-gram appears, 0 otherwise;
- keep only the "generic" features and the first three features.

4 Run configurations

Ten runs were submitted to the organisers based on different configurations. Modification of the data was carried out using both instance reduction and feature selection techniques. The system facilitated the use of different training data for each of the three main classification steps.

4.1 Least frequent words filter

Before classification, the data is preprocessed by replacing all the least frequent words with a default value (actually treated as missing values by the classifier). This is intended to help the classifier focus on the most relevant indications and to prevent over-specification of the classification model.

4.2 Instance reduction filters

4.2.1 POSTrigrams filter

The POS trigrams filter works as follows: during the training stage, the sequences of POS tags for the words *current-1.current.current+1* are extracted for each instance, together with its corresponding class. Every POS trigram is then associated with the following ratio:

$$\frac{\text{Frequency of true instances}}{\text{Frequency of false instances}}$$

Then, when predicting the class, the filter is applied before running the classifier: the sequences of trigrams are extracted for each instance, and are compared against the corresponding ratio observed during the training stage; the instance is filtered out if the ratio is lower than some threshold N%. In Table

Run	Detection	Classification	Correction
0	R1	Normal	Normal
1	R20	Normal	Normal
2	Full	F12	Normal
3	R10	Normal	Normal
4	R30	Normal	Normal
5	F12	F12	Normal
6	R4new	Normal	Normal
7	R4 + F12	F12	Normal
8	R4	Normal	Normal
9	R2	Normal	Normal

Table 2: Run configurations

2, the label RN refers to the percentage (N) used as cut-off in the experiments.

This filter is intended to reduce the impact of the fact that the classes are strongly unbalanced. It permits discarding a high number of false instances, while removing only a small number of true instances. However, as a side effect, it can cause the classifier to miss some clues which were in the discarded instances.

4.2.2 CurrentPlusOrMinusOne filter

The current plusorminus one filter works as follows: A list of all *current.current+1* word bigrams is made from the error instances in the training data, along with all *current-1.current* bigrams. The non-error instances in the training data are then filtered based on whether an instance contains an occurrence of any *current.current+1* or *current-1.current* bigram in the list.

4.3 Feature selection filters

4.3.1 F12

During preliminary experiments, selecting a subset of 12 features produced classification accuracy gains in the detection and classification steps of the process using ten-fold cross validation on the training set. These twelve features were: *current, current+1.current+2, current-1.current-2, currentSC, currentPOS, current-1, current-2, current+1, current+2, current+1SC,* and *current-1SC*. The SC postfix refers to the spell-corrected token, with POS referring to the part-of-speech tag. The F12 configuration filter removes all other features except these.

5 Results

Table 3 displays the results for both preposition and determiner errors which were obtained by the system on the preliminary test set before teams submitted their revisions. Table 4 refers to the results obtained by the system after the revised errors were removed/edited.

Task	Rank	Run	Precision	Recall	F-Score
Detection	11	9	5.33	25.61	8.82
Recognition	11	9	4.18	20.09	6.92
Correction	11	9	2.66	12.8	4.41

Table 3: Overall results on original data: TC

Task	Rank	Run	Precision	Recall	F-Score
Detection	11	8	6.56	26.0	10.48
Recognition	11	8	4.91	19.45	7.84
Correction	11	8	3.09	12.26	4.94

Table 4: Overall results on revised data: TC

5.1 Some detailed results (detection)

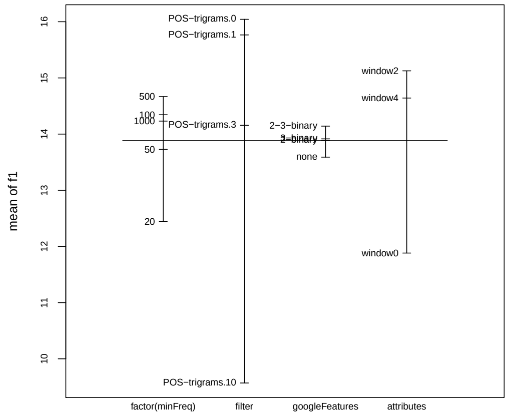
The results reported here were obtained on the training data only, using 5-fold cross-validation, and only for the detection task. We have studied various settings for the parameters; figure 1 shows a global overview of the performance depending on several parameters (we show only a few different values in order to keep the graph readable).

The results show that the Google features contribute positively to the performance, but only slightly: the F1 score is 0.6% better on average. This overview also hides the fact that some combinations of values work better together; for instance, contrary to the fact that not filtering the POS trigrams per-

Run3	**Recall**	**Precision**	**F**
Detection	9.05	7.42	8.15
Correction	4.19	3.44	3.78
Recognition	9.05	7.42	8.15
Run8	**Recall**	**Precision**	**F**
Detection	22.51	5.44	8.76
Correction	11.25	2.72	4.38
Recognition	22.51	5.44	8.76
Run9	**Recall**	**Precision**	**F**
Detection	25.61	5.33	8.82
Correction	12.80	2.66	4.41
Recognition	20.09	4.18	6.92

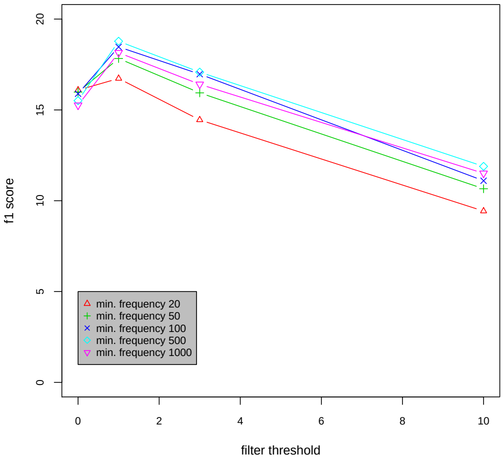
Table 5: Top results on original test data

Figure 1: Average F-score depending on several parameters.



forms better on average, the best performances are obtained when filtering, as shown in figure 2.

Figure 2: F-score (%) w.r.t POS trigrams filter threshold. Parameters: window 2, Google features with bigrams and trigrams.



- *Minimum frequency*[4] (preprocessing, see 4.1). As shown in Figure 2, using a high threshold helps the classifier build a better model.

- *POS trigrams filter* (see 4.2.1.) Even if not filtering at all performs better on average, the best cases are obtained with a low threshold. Additionally, this parameter can be used to balance between recall and precision (when one wants to favor one or the other).

- *Size of the context window.* Results can show important differences depending on the size of the window, but no best configuration was found in general for this parameter.

- *Google features* (see 3.3.2.) The Google features help slightly in general, and are used in the best cases that we have obtained. However there is no significantly better approach between using the original frequencies, simplifying these to binary values, or even not using the list of target expressions.

6 Conclusions

The task of automated error correction is a difficult one, with the best-performing systems managing approx. 40 % F-score for the detection, recognition and correction (Dale et al., 2012). There are several areas where our system's performance might be improved. The spellcheck dictionary which was used

[4] Remark: the values used as "minimum frequencies" reported in this paper can seem unusually high. This is due to the fact that, for technical reasons, the thresholds were applied globally to the data after it had been formatted as individual instances, each instance containing a context window of 9 words. As a consequence a threshold of N means that a given word must occur at least N/9 times in the original input data.

was a general one and this resulted in many spelling corrections which were out of context. A more tailored dictionary employing contextual awareness information could be beneficial for the preprocessing step.

Multi-word corrections were not supported by the system due to how the instances were constructed and these cases were simply ignored, to the detriment of the results.

In the basic feature set, the majority of features were based on word unigrams, however more n-gram features could improve results as these were found to perform well during classification.

There were many different ways to exploit the Google N-Grams features and it may be the case that better combinations of features can be found for each of the classification steps.

Finally, very little time was spent tuning the datasets for the classification and correction step as opposed to the detection phase, this is another part of the system where fine-tuning parameters could improve performance.

Acknowledgments

This material is based upon works supported by the Science Foundation Ireland under Grant No.[SFI 07/CE/I 1142.].

References

Robert Dale and Adam Kilgarriff. 2011. Helping Our Own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, Dublin, Ireland.

Robert Dale, Ilya Anisimoff, and George 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.

O. Gospodnetic and E. Hatcher. 2005. *Lucene*. Manning.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

J.B. Michel, Y.K. Shen, A.P. Aiden, A. Veres, M.K. Gray, J.P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176.

OpenNLP. 2012. Website: http://opennlp. apache. org.