Native Language Identification using large scale lexical features

André Lynum
Norwegian University of Science and Technology
Department of Computer and Information and Science
Sem Sælands vei 7-9
NO-7491 Trondheim, Norway
andrely@idi.ntnu.no

Abstract

This paper describes an effort to perform Native Language Identification (NLI) using machine learning on a large amount of lexical features. The features were collected from sequences and collocations of bare word forms, suffixes and character n-grams amounting to a feature set of several hundred thousand features. These features were used to train a linear Support Vector Machine (SVM) classifier for predicting the native language category.

1 Introduction

Much effort in Native Language Identification (NLI) has focused on identifying specific characteristics of the errors in texts produced by English Second Language (ESL) learners, like the work presented in (Bestgen et al., 2012) and (Koppel et al., 2005). This might be specific spelling errors, syntactic or morphological mistakes. One motivation for this approach has been the notion that aspects of the L1 language influences which errors and mistakes are produced by L2 learners, which has guided the model building towards a smaller number of features and models which lend themselves to interpretation in terms of linguistic knowledge.

Research so far has shown mixed support that this notion of *language transfer* is the best indicator of L1 language. While many such features are highly predictive, features that are usually indicative of the text topic has shown strong performance when applied to the NLI task as demonstrated in (Ahn, 2011) and (Koppel et al., 2005). This is largely lexical features such as frequency measures of token, lemma or character n-grams. There has been some effort in identifying if this is an artifact of biases in the available corpora or if it is indeed an indication of a substantial phenomenon in ESL language use by different L1 learners (Ahn, 2011).

The approach in this paper extends the use of lexicalized features and shows that such lexicalized features can by themselves form the basis of a competitive and robust NLI system. This approach entails possibly abondoning interpretability and other linguistic considerations in order to build an as efficient as possible system on the NLI classification tasks itself. It is also motivated by the possibility that simple lexicalized features can be applied efficiently in a task that on the face of it requires the system to on some level learn differences syntactic relations in addition to the differences in morphology found in text produced by the ESL learners.

The experiments presented in this paper are a result of exploring a range of features and machine learning approaches. The best systems found used a combination of bareword features, character n-grams, suffix and bareword collocations with TF-IDF weighting. The resulting feature space contains several hundred thousand features which were used to train a linear Support Vector Machine (SVM) classifier. I will first present the features an how they were extracted in section 2, details of the SVM model is presented in section 3, the different systems submitted to the shared task are described in section 4, along with the results in section 5. I have also included som discussion of issues encountered during the development of features and models in section 6.

Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, pages 266–269,
Atlanta, Georgia, June 13 2013.

2 Model features

This section describes the features used in the submitted systems. All the different text features are derived from the surface form of the training and development corpora without any additional processing or annotation. The provided tokenization was used and no steming, lemmatization or syntactic parsing was performed on the data.

2.1 Bareword features

The frequency of each token by itself was used as a feature, without any processing or normalization. I.e. no stemming was used, and any capitalization was kept.

2.2 Character n-gram features

These features consists of n-grams of length n. Character n-grams includes single spaces between tokens and newlines between lines. The systems presented in this paper uses n-gram orders 3-6 or 1-7.

2.3 Bareword directed collocation features

These are frequencies of the collocations of the bare tokens. The features includes the direction of the collocation, such that a different feature is generated if a token is collocated to the left or right of another token. The collocations are restricted to a window around the target token, and all the systems in this paper uses a window of one token making this feature identical to token bigrams.

2.4 Suffix directed collocation features

These features are constructed in the same manner as the directed bareword collocation features described in 2.3 except that they are based on the 4-character long suffix of each token.

2.5 Feature filtering and TF-IDF weighting

Features that are presumed to be uninformative are filtered out before classifier training and prediction. Features with a document count less than a certain limit varying between the systems were ignored, along with features which appears in more than 50% of the documents, i.e. with a Document Frequency (DF) over 0.5.

All the features based on character n-gram or word counts from the corpus was scaled using sublinear Term Frequency (TF) scaling as described in for exeample (Manning et al., 2008). In addition the IDF was adjusted using add-one smoothing, i.e. one was added to all DF counts[1].

2.6 Proficiency and prompt features

Both proficiency value and prompt value for the document are used as features in the form of $0-1$ indicators for the possible values.[2]

3 SVM classification

The system uses an SVM multiclass classifier. The SVM classifier was trained without a kernel, i.e. linear, and with the cost parameter optimized through cross validation. SVM was used since it can train models with a large number of features efficiently, and has been successsfully used to construct high-dimensional models in many NLP tasks (Joachims, 1998), including NLI (Tsur and Rappoport, 2007; Koppel et al., 2005).

The cost hyperparameter of the SVM models was optimized over 5-fold cross validation on the training set.

4 Systems submitted

Four systems were submitted to the shared task. Of these three share the same feature types and differ in the DF cutoff used to prune individual features. The fourth system adds additional character n-grams to the features found in the other three systems.

The first three systems are based on the following features:

- Weighted token counts.
- Weighted character n-grams of orders 3 through 6.
- Prompt and proficiency values.

[1]The documentation of the software used for feature extraction notes that this smoothing is mainly for numerical considerations, i.e. avoiding division by zero errors (http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html)

[2]While the prompt value is included in the submitted systems it was not found to be an effective feature and did not have any effect on the performance of the systems. Its inclusion in the feature set is an oversight.

- Weighted directed token collocation counts with a window size of one, i.e. token bigrams.

- Weighted directed 4 character suffix collocation counts with a window size of 1, i.e. 4 character suffix bigrams.

The three systems vary in the DF cutoff with no cutoff in systmem 1, a cutoff of 5 in system 2 and a cutoff of 10 in system 3.

System 4 uses different cutoffs for different features; 10 for token and character n-gram frequencies and 5 for the token and suffix collocation features. It also uses character n-grams of order 1 through 7 instead of 3 through 6.

Table 1 show the performance of the four systems on the development data set in addition to the feature count for each of the systems. The table shows both classification accuracy on the development data set in addition to average and standard deviation for 10-fold cross validation scores over the combined training and development data sets.

The software used to generate the systems is available at https://github.com/andrely/NLI2013-submission.

5 Results

The final results shows competitive performance from all the submitted systems with little variation in performance between them. Both test set accuracies and average 10-fold cross validation scores with standard deviation for the shared tasks fixed folds are given in table 2.

6 Some impressions

Performance stability: When developing the various systems the performance was always robust for the features described in this paper and variations on them. There were little variation in 5-fold cross validation scores, or difference between cross validation and held out scores. This was taken as an indication that the system was not being overfitted despite the amount of and specificity of the features.

Feature comparison: All the lexical features used were highly predictive also in isolation, and could be used for a competetive system by themselves.

POS tags and lemmatization: Similar features based on POS tags or lemmatized tokens turned out to be much less predictive than the lexical features. This could be caused by low quality of such annotation on data with many spelling or other errors.

Acknowledgments

The Python software package Scikit-learn[3] (Pedregosa et al., 2011) and libSVM[4] (Chang and Lin, 2011) was used to implement the systems described in this paper.

References

Charles S. Ahn. 2011. Automatically Detecting Authors' Native Language. Master's thesis, Naval Postgraduate School, Monterey, CA.

Yves Bestgen, Sylviane Granger, and Jennifer Thewissen. 2012. Error Patterns and Automatic L1 Identification. In Scott Jarvis and Scott A. Crosley, editors, *Approaching Language Transfer through Text Classification*, pages 127–153. Multilingual Matters.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Thorsten Joachims. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK. Springer-Verlag.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. *Intelligence and Security Informatics*, pages 41–76.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Oren Tsur and Ari Rappoport. 2007. Using Classifier Features for Studying the Effect of Native Language

[3] http://scikit-learn.org/

[4] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

System	# of features	Dev. 10-fold accuracy	Dev. accuracy
1	867479	0.841 ± 0.010	0.827
2	439063	0.839 ± 0.012	0.824
3	282797	0.838 ± 0.012	0.823
4	510191	0.836 ± 0.011	0.824

Table 1: Performance and number of features for the submitted systems. Performance is shown as accuracy on the development data set and 10-fold cross validation on the training and test set. The feature counts shown are for the final systems trained on the training and development data sets. The systems are described in section 4.

System	Accuracy	10-fold accuracy
1	0.833	0.839 ± 0.013
2	0.834	0.837 ± 0.011
3	0.833	0.835 ± 0.012
4	0.830	0.835 ± 0.012

Table 2: Final accuracy scores on the test set and 10-fold cross validation for the submitted systems. The systems are described in section 4.

on the Choice of Written Second Language Words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 9–16, Prague, Czech Republic, June. Association for Computational Linguistics.