Feature Space Selection and Combination for Native Language Identification

Cyril Goutte
National Research Council
1200 Montreal Rd,
Ottawa, ON K1A 0R6
Cyril.Goutte@nrc.ca

Serge Léger
National Research Council
100, des Aboiteaux St.,
Moncton, NB E1A 7R1
Serge.Leger@nrc.ca

Marine Carpuat
National Research Council
1200 Montreal Rd,
Ottawa, ON K1A 0R6
Marine.Carpuat@nrc.ca

Abstract

We decribe the submissions made by the National Research Council Canada to the Native Language Identification (NLI) shared task. Our submissions rely on a Support Vector Machine classifier, various feature spaces using a variety of lexical, spelling, and syntactic features, and on a simple model combination strategy relying on a majority vote between classifiers. Somewhat surprisingly, a classifier relying on purely lexical features performed very well and proved difficult to outperform significantly using various combinations of feature spaces. However, the combination of multiple predictors allowed to exploit their different strengths and provided a significant boost in performance.

1 Introduction

We describe the National Research Council Canada's submissions to the Native Language Identification 2013 shared task (Tetreault et al., 2013). Our submissions rely on fairly straightforward statistical modelling techniques, applied to various feature spaces representing lexical and syntactic information. Our most successful submission was actually a combination of models trained on different sets of feature spaces using a simple majority vote.

Much of the work on Natural Language Processing is motivated by the desire to have machines that can help or replace humans on language-related tasks. Many tasks such as topic or genre classification, entity extraction, disambiguation, are fairly straightforward for humans to complete. Machines typically trade-off some performance for ease of application and reduced cost. Equally fascinating are tasks that seem non-trivial to humans, but on which machines, through appropriate statistical analysis, discover regularities and dependencies that are far from obvious to humans. Examples may include categorizing text by author gender (Koppel et al., 2003) or detecting whether a text is an original or a translation (Baroni and Bernardini, 2006). This is one motivation for addressing the problem of identifying the native language of an author in this shared task.

In the following section, we describe various aspects of the models and features we used on this task. In section 3, we describe our experimental settings and summarize the results we obtained. We discuss and conclude in section 4.

2 Modelling

Our submissions rely on straightforward statistical classifiers trained on various combinations of features and feature spaces. We first describe the classifier we used, then give the list of features that we have been combining. Our best performing submission used a combination of the three systems we submitted in a majority vote, which we also describe at the end of this section.

2.1 Classification Model

We decided to use a straightforward and state-of-the-art statistical classifier, in order to focus our attention on the combination of features and models rather than on the design of the classifier.

Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, pages 96–100,
Atlanta, Georgia, June 13 2013. ©2013 Association for Computational Linguistics

We used freely available implementations of Support Vector Machines (SVM) provided in SVM-light (Joachims, 1999) and SVM-perf (Joachims, 2006). SVM performance may be influenced by at least two important factors: the choice of the kernel and the trade-off parameter "C". In our experiments, we did not observe any gain from using either polynomial or RBF kernels. All results below are therefore obtained with linear models. Similarly, we investigated the optimization of parameter "C" on a held-out validation set, but found out that the resulting performance was not consistently significantly better than that provided by the default value. As a consequence our results were obtained using the SVM-light default.

One important issue in this shared task was to handle multiple classes (the 11 languages). There are essentially two easy approaches to handle single label, multiclass classification with binary SVM: one-versus-all and one-versus-one. We adopted the one-versus-all setting, combined with a calibration step. We first trained 11 classifiers using the documents for each language in turn as "positive" examples, and the documents for the remaining 10 languages as negative examples. The output score for each class-specific SVM model was then mapped into a probability using isotonic regression with the pair-adjacent violators (PAV) algorithm (Zadrozny and Elkan, 2002). A test document is then assigned to the class with the highest probability.

2.2 Feature Space Extraction

We extracted the following features from the documents provided for the shared task.

Character ngrams: We index trigrams of characters within each word (Koppel et al., 2005). The beginning and end of a word are treated as special character. For example, the word "at" will produce two trigrams: " at" and "at ". These features allow us to capture for example typical spelling variants. In a language with weak morphology such as English, they may also be able to capture patterns of usage of, e.g. suffixes, which provides a low-cost proxy for syntactic information.

Word ngrams: We index unigrams and bigrams of words within each sentence. For bigrams, the beginning and end of a sentence are treated as special tokens. Note that we do not apply any stoplist filtering. As a consequence, function words, an often-used feature (Koppel et al., 2005; Brooke and Hirst, 2012), are naturally included in the unigram feature space.

Spelling features: Misspelled words are identified using GNU Aspell V0.60.4[1] and indexed with their counts. Some parser artifacts such as "n't" are removed from the final mispelled word index. Although misspellings may seem to provide clues as to the author's native language, we did not find these features to be useful in any of our experiments. Note however, that misspelled words will also appear in the unigram feature space.

Part-of-speech ngrams: The texts were tagged with the Stanford tagger v. 3.0[2] using the largest and best (bidirectional) model. Note that the language in a couple of documents was so poor that the tagger was unable to complete, and we reverted to a slightly weaker (left three words) model for those. After tagging, we indexed all ngrams of part-of-speech tags, with $n = 2, 3, 4, 5$. We experimented with the choice of n and found out that $n > 2$ did not bring any significant difference in performance.

Syntactic dependencies: We ran the Stanford Parser v2.0.0 on all essays, and use the typed dependency output to generate features. Our goal is to capture phenomena such as preposition selection which might be influenced by the native language of the writer. In order to reduce sparsity, each observed dependency is used to generate three features: one feature for the full lexicalized dependency relation; one feature for the head (which generalizes over all observed modifiers); one feature for the modifier (which generalizes over all possible heads). For instance, in the sentence "they participate to one 's appearance", the parser extracts the following dependency: "prep$_{to}$(participate,appearance)". It yields three features "prep$_{to}$(participate,appearance)", "prep$_{to}$(participate,X)" and "prep$_{to}$(X,appearance)". We experimented with all three feature types, but the systems used for the

[1] http://aspell.net

[2] http://nlp.stanford.edu/software/tagger.shtml

official evaluation results only used the last two (head and modifier features.) Note that while these features can capture long distance dependencies in theory, they significantly overlap with word ngram features in practice.

For each feature space, we used a choice of two weighting schemes inspired by SMART (Manning et al., 2008):

ltc: log of the feature count, combined with the log inverse document frequency (idf), with a cosine normalization;

nnc: straight feature count, no idf, with cosine normalization.

Normalization is important with SVM classifiers as they are not scale invariant and tend to be sensitive to large variations in the scale of features.

2.3 Voting Combination

Investigating the differences in predictions made by different models, it became apparent that there were significant differences between systems that displayed similar performance. For example, our first two submissions, which perform within 0.2% of each other on the test data, disagree on almost 20% of the examples.

This suggests that there is potentially a lot of information to gain by combining systems trained on different feature spaces. An attempt to directly combine the predictions of different systems into a new predictive score proved unsuccessful and failed to provide a significant gain over the systems used in the combination.

A more successful combination was obtained using a simple majority vote. Our method relies on simply looking at the classes predicted by an ensemble of classifier for a given document. The prediction for the ensemble will be the most predicted class, breaking possible ties according to the overall scores of the component models: for example, for an ensemble of only 2 models, the decision in the case of a tie will be that of the best model.

3 Experiments

We describe the experimental setting that we used to prepare our submissions, and the final performance we obtained on the shared task (Tetreault et al., 2013).

3.1 Experimental Setting

In order to test the performance of various choices of feature spaces and their combination, we set up a cross-validation experimental setting. We originally sampled 9 equal sized disjoint folds of 1100 documents each from the training data. We used stratified sampling across the languages and the prompts. This made sure that the folds respected the uniform distribution across languages, as well as the distribution across prompts, which was slightly uneven for some languages. These 9 folds were later augmented with a 10th fold containing the development data released during the evaluation.

All systems were evaluated by computing the accuracy (or equivalently the micro-averaged F-score) on the cross-validated predictions.

3.2 Experimental Results

We submitted four systems to the shared task evaluation:

1. BOW2^{ltc}+CHAR3^{ltc}: Uses counts of word bigrams and character trigrams, both weighted independently with the ltc weighting scheme (tf-idf with cosine normalization);

2. BOW2^{ltc}+DEP^{ltc}: Uses counts of word bigrams and syntactic dependencies, both weighted independently with the ltc weighting scheme;

3. BOW2^{ltc}+CHAR3^{ltc}+POS2^{nnc}: Same as system #1, adding counts of bigrams of part-of-speech tags, independently cosine-normalized;

4. 3-system vote: Combination of the three submissions using majority vote.

The purpose of submission #1 was to check the performance that we could get using only surface form information (words and spelling). As shown on Table 1, it reached an average test accuracy of 79.5%, which places it in the middle of the pack over all submissions. For us, it establishes a baseline of what is achievable without any additional syntactic information provided by either taggers or parsers.

Model	#	Acc(%)
BOW2^{ltc}+CHAR3^{ltc}	1	79.27
BOW2^{ltc}+DEP^{ltc}	2	79.55
BOW2^{ltc}+CHAR3^{ltc}+POS2^{nnc}	3	78.82
3-system vote	4	81.82
10-system vote	-	84.00

Table 1: The four systems submitted by NRC, plus a more extensive voting combination. System 1 uses only surface information. Systems 2 and 3 use two types of syntactic information and system #4 uses a majority vote among the three previous submissions. The last (unsubmitted) uses a majority vote among ten systems.

Our submissions #2 and #3 were meant to check the effect of adding syntactic features to basic lexical information. We evaluated various combinations of feature spaces using cross-validation performance and found out that these two combinations seemed to bring a small boost in performance. Unfortunately, as shown on Table 1, this did not reflect on the actual test results. The test performance of submission #2 was a mere 0.2% higher than our baseline, when we expected +0.6% from the cross-validation estimate. The test performance for submission #3 was 0.5% *below* that of the baseline, whereas we expected a small increase.

Submission #4 was our majority voting submission. Due to lack of time, we could not generate test predictions for all the systems that we wanted to include in the combination. As a consequence, we performed a majority voting over just the 3 previous submissions. Despite this, the majority voting proved remarkaby effective, yielding a 2.5% performance boost over our baseline, and a 2.3% increase over our best single system.

In order to further test the potential of the majority vote, we later applied it to the 10 best systems in a pool generated from various combinations of feature spaces (*10-system vote* in Table 1). That (unsubmitted) combination outperformed our official submissions by another 2.2% accuracy, and in fact outperformed the best system in the official evaluation results by a small (and very likely not significant) margin.

In comparison with submissions from other groups, our top submission was 1.8% below the top performing system (Table 2). According to the results of significance tests released by the organizers, the difference is slightly below the traditional threshold of statistical significance (0.05).

Model	Accuracy(%)	p-value
Jarvis	83.6	0.082
Oslo NLI	83.4	0.1
Unibuc	82.7	0.361
MITRE-Carnie	82.6	0.448
Tuebingen	82.2	0.715
NRC	**81.8**	
CMU-Haifa	81.5	0.807
Cologne-Nijmegen	81.4	0.665
NAIST	81.1	0.472
UTD	80.9	0.401
UAlberta	80.3	0.194
Toronto	80.2	0.167
MQ	80.1	0.097

Table 2: Resulting accuracy scores and significance vs. NRC top submission (3-system vote).

4 Discussion and Conclusion

Our results suggest that on the shared task, a combination of features relying only on word and character ngrams provided a strong baseline. Our best system ended up being a combination of models trained on various sets of lexical and syntactic features, using a simple majority vote. Our submission #4 combined only our three other submissions, but we later experimented with a larger pool of models. Table 3 shows that the best performance is obtained using the top 10 models, and many of the combinations are competitive with the best performance achieved during the evaluation. Our cross-validation estimate was also maximized for 10 models, with as estimated accuracy of 83.23%. It is interesting that adding some of the weaker models does not seem to hurt the voting combination very much.

One obvious limitation of this study is that it was applied to a well defined and circumscribed setting. There is definitely no guarantee on the performance that may be obtained on a different corpus of documents.

Another limitation is that although the resulting performance of our models seems encouraging, it is not obvious that we have learned particularly

Rank	Model score	Vote score	Feature set
1	79.55	79.55	BOW2+DEP
2	79.36	79.55	BOW1+DEP
3	79.27	82.18	BOW2+CHAR3
4	79.00	82.27	BOW1+DEPL
5	78.91	82.91	BOW2+CHAR3+POS3
6	78.82	83.18	BOW2+CHAR3+POS2
7	78.73	83.45	BOW2+DEPL
8	78.36	83.55	BOW2
9	77.09	**83.82**	BOW1+POS3
10	76.82	**84.00**	BOW2+POS2
11	76.55	**83.64**	BOW2+POS3
12	76.55	**83.82**	BOW1+POS2
13	75.27	83.55	BOW1
14	74.36	**83.73**	BOW1+CHAR3
15	74.27	**83.73**	DEP
16	66.91	**83.91**	DEPL
17	64.18	**83.82**	CHAR3
18	51.64	**83.82**	POS3
19	49.64	83.36	POS2

Table 3: Majority vote among the top-N models. BOWn=word ngrams; CHAR3=char trigrams; POSn=POS ngrams; DEP/DEPL=syntactic dependecies.

useful clues about what differentiates the English written by authors with different native languages. This is of course a side effect of a format where systems compete on a specific performance metric, which encourages using large, well-regularized models which optimize the relevant metric, at the expense of sparser models focusing on a few markers that may be more easily understandable.

During the workshop, we plan to show more complete results using the majority vote strategy, involving a wider array of base models.

References

M. Baroni and S. Bernardini. 2006. A new approach to the study of translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274.

Julian Brooke and Graeme Hirst. 2012. Robust, lexicalized native language identification. In *Proceedings of COLING 2012*.

T. Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*.

T. Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

M. Koppel, S. Argamon, and A. R. Shimoni. 2003. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17:401–412.

M. Koppel, J. Schler, and K. Zigdon. 2005. Determining an authors native language by mining a text for errors. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD 05)*, pages 624–628, Chicago, Ilinois, USA.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. Document and query weighting schemes. In *Introduction to Information Retrieval*. Cambridge University Press.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, Atlanta, GA, USA, June. Association for Computational Linguistics.

B. Zadrozny and C. Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD'02)*.