The NRC System for Discriminating Similar Languages

Cyril Goutte, Serge Léger and Marine Carpuat
Multilingual Text Processing
National Research Council Canada
Ottawa, ON K1A0R6
`Firstname.Lastname@nrc.ca`

Abstract

We describe the system built by the National Research Council Canada for the "Discriminating between similar languages" (DSL) shared task. Our system uses various statistical classifiers and makes predictions based on a two-stage process: we first predict the language group, then discriminate between languages or variants within the group. Language groups are predicted using a generative classifier with 99.99% accuracy on the five target groups. Within each group (except English), we use a voting combination of discriminative classifiers trained on a variety of feature spaces, achieving an average accuracy of 95.71%, with per-group accuracy between 90.95% and 100% depending on the group. This approach turns out to reach the best performance among all systems submitted to the open and closed tasks.

1 Introduction

Language identification is largely considered a solved problem in the general setting, except in frontier cases such as identifying languages from very little data, from mixed input or when discriminating similar languages or language variants.

The "Discriminating between similar languages" (DSL) shared task proposes such a situation, with an interesting mix of languages, as can be seen in Table 1. Three groups contain similar languages (Bosnian+Croatian+Serbian, Indonesian+Malaysian, Czech+Slovakian); three groups contain variants of the same language (Portuguese, Spanish and English). In addition, instances to classify are single sentences, a more realistic and challenging situation than full-document language identification.

Our motivation for taking part in this evaluation was threefold. First, we wanted to evaluate our in-house implementation of document categorization on a real and useful task in a well controlled experimental setting.[1] Second, classifiers that can discriminate between similar languages can be applied to tasks such as identifying close dialects, and may be useful for training Statistical Machine Translation systems more effectively. For instance, Zbib et al. (2012) show that small amounts of data from the right dialect can have a dramatic impact on the quality of Dialectal Arabic Machine Translation systems. Finally, we view the DSL task as a first step towards building a system that can identify code-switching in, for example, social media data, a task which has recently received increased attention from the NLP community[2] (Elfardy et al., 2013).

The next section reviews the modeling choices we made for the shared task, and section 3 describes our results in detail. Additional analysis and comparisons with other submitted systems are available in the shared task report (Zampieri et al., 2014).

2 Modeling

Our approach relies on a two-stage process. We first predict the language group, then discriminate the languages or variants within the group. This approach works best if the first stage (i.e. group) classifier

[1]A previous version of our categorization tool produced good results on a Native Language Identification task in 2013 (Tetreault et al., 2013; Goutte et al., 2013).

[2]http://emnlp2014.org/workshops/CodeSwitch/

has high accuracy, because if the wrong group is predicted, it is impossible to recover from that mistake in the second stage. On the other hand, as most groups only comprise two languages or variants, our two-stage process makes it possible to rely on a simple binary classifier within each group, and avoid the extra complexity that comes with multiclass modeling.

We were able to build a high-accuracy, generative group classifier (Section 2.2) and rely on Support Vector Classifiers within each group to predict the language or variant (Section 2.3). Group F was treated in a slightly different way, although the underlying model is identical (Section 2.4). Before describing these classifiers, we briefly describe the features that we extract from the textual data.

2.1 Feature Extraction

The shared task uses sentences as basic observations, which is a reasonable granularity for this task. As we want to extract lexical as well as spelling features, we focus on two types of features:

- Word *ngrams*: Within sentence consecutive subsequences of n words. In our experiments we considered unigrams (bag of words) and bigrams (bag of bigrams); performance seems to degrade for higher order *ngrams*, due to data sparsity. For bigrams, we use special tokens to mark the start and end of sentences.

- Character *ngrams*: Consecutive subsequences of n characters. In our experiments we use $n = 2, 3, 4, 5, 6$. We use special characters to mark the start and end of sentences.

For each type of feature, we index all the *ngrams* observed at least once in the entire collection. Although it may seem that we risk having a combinatorial explosion of character *ngram* features for large values of n, the number of actually observed *ngrams* is clearly sub-exponential and grows roughly as $\mathcal{O}(n^6)$.

2.2 Language Group Classifier

Predicting the language group is a 6-way classification task, for which we use the probabilistic model described in (Gaussier et al., 2002; Goutte, 2008). We consider this model because it is more convenient in a multiclass setting than the multiclass SVM approach described below: only one model is required and training is extremely fast. We ended up choosing it because it provided slightly better estimated performance on the group prediction task.

This is a generative model for co-occurrences of words w in documents d. It models the probability of co-occurrence $P(w, d)$ as a mixture model over classes c:

$$P(w, d) = \sum_c P(w|c)P(d|c)P(c) = P(d) \sum_c P(w|c)P(c|d), \qquad (1)$$

where $P(w|c)$ is the profile for class c, ie the probability that each word[3] w in the vocabulary may be generated for class c, and $P(c|d)$ is the profile for document d, ie the probability that a word from that document is generated from each class.

In essence, this is a supervised version of the *Probabilistic Latent Semantic Analysis* model (Hofmann, 1999). It is similar to the *Naive Bayes* model (McCallum and Nigam, 1998), except that instead of sampling the class once per document and generating all words from that class, this model can re-sample the class for each word in the document. This results in a much more flexible model, and higher performance.

Given a corpus of documents labelled with class information, and assuming that all co-occurrences in a document belong to the class of that document,[4] the maximum likelihood parameter estimates are identical to *Naive Bayes*. From the counts $n(w, d)$ of the occurences of word w in document d, and denoting $|c| = \sum_{d \in c} \sum_w n(w, d)$, the total number of words in class c, the maximum likelihood estimates

[3]In the context of this study, a "word" w may be a (word or character) *ngram*, according to Section 2.1.

[4]This means that for a training document d in class c_d, $P(c_d|d) \equiv 1$.

for the profile parameters are:

$$\widehat{P}(w|c) = \frac{1}{|c|} \sum_{d \in c} n(w, d). \tag{2}$$

Maximum likelihood estimates for parameters $P(d)$ and $P(c|d)$ may be obtained similarly, but they are not useful for predicting new documents. The model is therefore solely represented by a set of class profile vectors giving lexical probabilities in each class.

Note that this is a generative model for the training collection only. In order to predict class assignment for a new document, we need to introduce the new document $\widetilde{d}$ and associated, unknown parameters $P(\widetilde{d})$ and $P(c|\widetilde{d})$. We estimate the posterior assignment probability $P(c|\widetilde{d})$ by *folding in* $\widetilde{d}$ into the collection and maximizing the log-likelihood of the new document,

$$\widetilde{\mathcal{L}} = \sum_{w} n(w, \widetilde{d}) \log P(\widetilde{d}) \sum_{c} P(c|\widetilde{d}) P(w|c),$$

with respect to $P(c|\widetilde{d})$, keeping the class profiles $P(w|c)$ fixed. This is a convex optimization problem that may be efficiently solved using the iterative Expectation Maximization algorithm (Dempster et al., 1977). The resulting iterative, fixed-point equation is:

$$P(c|\widetilde{d}) \leftarrow P(c|\widetilde{d}) \sum_{w} \frac{n(w, \widetilde{d})}{|\widetilde{d}|} \frac{P(w|c)}{\sum_{c} P(c|\widetilde{d}) P(w|c)}, \tag{3}$$

with $|\widetilde{d}| = \sum_{w} n(w, \widetilde{d})$ is the length of document $\widetilde{d}$. Because the minimization is convex w.r.t. $P(c|\widetilde{d})$, the EM update converges to the unique maximum.

Given a corpus of annotated documents, we estimate model parameters using the maximum likelihood solution (2). This is extremely fast and ideal for training on the large corpus available for this evaluation. At test time, we initialize $P(c|\widetilde{d})$ with the uniform distribution and run the EM equation (3) until convergence for each test sentence. This is relatively slow (compared to training), but may be easily and efficiently parallelized on, e.g. multicore architecture.

Note that although group prediction is a 6-way classification task, we ended up using a 13-class model predicting the languages or variants, mapping the predictions from the 13 language classes into the 6 groups. This provided slightly better estimated performance on group prediction, although the prediction on the individual languages was weaker than what we obtained with the models described in the following sections.

2.3 Language Classifiers within Groups A to E

Setting aside Group A for a moment, within each of the other groups, we need to discriminate between two languages or language variants, as summarized in Table 1. This is the ideal situation for a powerful binary discriminative classifier such as the Support Vector Machines. We use a Support Vector Machine (SVM) classifier, as implemented in SVM$_{light}$ (Joachims, 1998).

Note that the probabilistic classifier described in the previous section may provide predictions over all 13 classes (11 without English) of the shared task with one single model. However, preliminary experiments showed that the resulting performance was slightly below what we could achieve using binary SVMs within each groups in the two-stage approach.

We trained a binary SVM on each of the feature spaces described in Section 2.1. We used a linear kernel, and set the C parameter in SVM$_{light}$ to the default value. Prediction with a linear kernel is very fast as it only requires computing the dot product of the vector space representation of a document with the equivalent linear weight vector.

Multiclass (Group A)

For group A, we need to handle the 3-way multiclass situation to discriminate between Bosnian, Croatian and Serbian. This is done by first training one linear SVM per class in a one-versus-all fashion. We then apply a calibration step using a Gaussian mixture on SVM prediction scores in order to transform these scores into proper posterior probabilities (Bennett, 2003). We then predict the class with the highest calibrated probability. Once the calibration model has been estimated on a small held-out set, applying the calibration to the three models and picking the highest value is very efficient.

Voting

The different *ngram* feature spaces lead to different models with varying performance. We combine these models using a simple voting strategy. Within each group, we rank the models trained on each feature space by performance, estimated by cross-validation (CV). We then perform a majority voting between predictions, breaking possible ties according to the estimated performance of the individual models.

When voting, adding models of lower performance typically improves the voting performance as long as their predictions are not too correlated with models that are already included in the vote. We therefore need to set the number of models to include in the vote carefully: this is also done by maximizing the performance based on the cross-validation estimator.

2.4 Classifier for Group F (English)

The specific issue of the English data from Group F is discussed in more details in the shared task report (Zampieri et al., 2014) so we only mention a few points that are specific to our system.

Due to the poor cross-validation performance (distinguishing GB and US english is difficult but obviously not impossible) we suspected early on that there was an issue with the data. We asked two native English speakers to perform a human evaluation on a small sample of the training and development data, which confirmed both our suspicion, and the fact that this was a difficult task. On the sentences that our judges confidently tagged GB or US (60% of the sample), they were wrong slightly more often than chance. We therefore suspected that if the test data was more reliable, a statistical model estimated on the training data may also do worse than chance.

We therefore decided to train a straightforward SVM model on bigrams of words. From this, we submitted two runs: one with the SVM predictions (run1), and the second with the same predictions flipped (run2).

3 Experimental Results

3.1 Data

The data provided for the evaluation is described in more detail in (Tan et al., 2014). Table 1 summarizes the size of the corpus across groups and languages for the training (including development) and test sets. Training and test data are balanced across languages and variants.

In order to provide an estimate of performance and guide our modeling choices, we use a 10-fold, stratified cross-validation estimator. We split the training examples for each language into ten equal-sized parts, and test on each fold the models trained on the remaining nine folds. The test predictions obtained on all the folds are then used to compute the cross-validation estimator.

3.2 Group Prediction

Training the group classifier using the probabilistic model described in section 2.2 on the 260,000 sentences using character 4-grams as features takes 133 seconds on a single, 32-core Linux workstation. Predicting the group for the 11,000 test documents (groups A-E) takes just 18 seconds, approximately 1.6ms/sentence.

The performance of the group predictor is near perfect: a single document is predicted incorrectly (Spanish instead of Portuguese) out of the 11,000 test sentences. This matches the excellent performance estimated by 10-fold cross-validation to an error of 0.038%.

3.3 Language Prediction in Groups A to E

For each group from A to E, we submitted:

		# sentences	
Group	Language	Train	Test
A	Bosnian	20,000	1000
	Croatian	20,000	1000
	Serbian	20,000	1000
B	Indonesian	20,000	1000
	Malaysian	20,000	1000
C	Czech	20,000	1000
	Slovak	20,000	1000
D	Brazil Portuguese	20,000	1000
	Portugal Portuguese	20,000	1000
E	Argentine Spanish	20,000	1000
	Spain Spanish	20,000	1000
F	GB English	20,000	800 (∗)
	US English	20,000	800 (∗)

Table 1: Number of training (including development) and test sentences accross groups and languages. (*): the English test data was available separately.

run1: The single best SVM model obtained on a single feature space (no voting), according to the 10-fold cross-validation. Depending on the group, the best feature space is either character 5grams or 6grams.

run2: Same model as run1, with additional tuning of the prediction threshold to ensure balanced predictions on the cross-validated data. On groups B to E, run1 uses a natural threshold of 0 to predict the language or variant. When the SVM score is positive, run1 predicts one class, when it is negative, run1 predicts the other. In constrast, run2 uses the fact that we know that the classes are balanced, and adjusts the threshold to force predictions to be balanced across classes.

run3: The best voting combination. It is obtained by ranking the various feature spaces by decreasing 10-fold CV performance, and picking the number of votes that yields the highest cross-validation estimate for the voting combination. Depending on the group, the best combination involves between 1 and 7 models.

Training the SVM models for group A, including calibration, on the 60,000 training sentences takes 7 minutes and 33 seconds for the best model (character 5grams), and 31 minutes overall for the 7 feature spaces. Prediction for the best model takes 16 seconds, approximately 1.5ms/sentence; for all 7 models used in the vote, prediction requires a total of 1 minute and 16 seconds.

Training on groups B to E is faster because we only need one SVM model per feature space. In addition, for group C, only one model is necessary because no vote outperforms the best model. Training the best model on each group (character 6gram) requires between 242 and 721 seconds depending on the group. Training all models used in the vote requires up to 29 minutes. Prediction with the best model takes 1.4 to 2.1ms/sentence, while computing all predictions used in the vote requires up to 8ms/sentence.

Table 2 summarizes the performance for our three runs on the 5 target groups. We give the cross-validation estimator computed before submission, as well as the test error obtained from the gold standard data released with the official results. Although there are small differences between actual test results and the CV estimates, the CV estimates are fairly reliable. They always indicate that run3 is best, which is only incorrect on Group D, where the actual test performance of run1 is only very slightly better.[5]

According to the official results (Zampieri et al., 2014), this allowed our system to get the best per-group accuracy on all groups, as well as the best overall accuracy with 95.71%. This is also higher than the two open submissions.

[5]The difference corresponds to only 2 sentences.

	Group A		Group B		Group C		Group D		Group E	
	CV	Test	CV	Test	CV	Test	CV	Test	CV	Test
run1 (1-best)	6.12	6.70	0.720	0.600	**0.0075**	**0.00**	4.85	**4.40**	10.59	9.85
run2 (thresh.)	6.12	6.70	0.720	0.600	**0.0075**	**0.00**	4.87	4.50	10.57	10.05
run3 (vote)	**5.54**	**6.40**	**0.642**	**0.450**	**0.0075**	**0.00**	**4.47**	4.50	**9.91**	**9.05**

Table 2: Cross-validated (CV) and test error (1-accuracy), in %, on Groups A to E.

(Group F)	CV	Test
run1 (bag-of-bigrams)	**44.67**	52.37
run2 (flipped)	55.33	**47.63**

Table 3: Cross-validated (CV) and test error (1-accuracy), in %, on Group F (English).

3.4 Group F (English)

Because of the data issue in that group, our submission used one of the simpler models. As a consequence, training and test times are of less relevance. Training the bigram model on 40,000 sentences took 2 minutes while prediction on the 1600 English test sentences took 4 seconds, i.e. 2.5ms/sentences.

Table 3 shows the cross-validation and test errors of our two runs on the English data. This illustrates that the cross-validated estimate for accuracy was poor for our system. As suspected, the more reliable test data shows that our system (**run1**) was in fact not learning the right task. As a result, our submission with flipped predictions (**run2**) yields better accuracy on the test set.

In fact it appears from official evaluation results that our run2 was the only close task submission that performed better than chance on the test data.

4 Summary

Using fairly straightforward modeling tools, a probabilistic document classifier and linear Support Vector Machines, we built a two stage system that classifies language variants in the shared task with an average accuracy of 95.71%, providing the best overall performance for both open and closed task submissions. The individual language group performance varies from 91% to 100% depending on the group. This systems seems like a good baseline for experimenting with dialect identification, or code-switching in social media data. We are especially interested in investigating how performance evolves with smaller segments of texts.

References

Paul N. Bennett. 2003. Using asymmetric distributions to improve text classifier probability estimates. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 111–118, New York, NY, USA. ACM.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. 2013. Code switch point detection in arabic. In *18th International Conference on Applications of Natural Language to Information Systems*, pages 412–416.

Éric Gaussier, Cyril Goutte, Kris Popat, and Francine Chen. 2002. A hierarchical model for clustering and categorising documents. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research: Advances in Information Retrieval*, pages 229–247, London, UK, UK. Springer-Verlag.

Cyril Goutte, Serge Léger, and Marine Carpuat. 2013. Feature space selection and combination for native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 96–100, Atlanta, Georgia, June. Association for Computational Linguistics.

Cyril Goutte. 2008. A probabilistic model for fast and confident categorization of textual documents. In Michael W. Berry and Malu Castellanos, editors, *Survey of Text Mining II*, pages 187–202. Springer London.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, pages 289–296.

Thorsten Joachims. 1998. Text categorization with Suport Vector Machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press.

Liling Tan, Marcos Zampieri, Nikola Ljubešić, and Jörg Tiedemann. 2014. Merging comparable data sources for the discrimination of similar languages: The DSL corpus collection. In *Proceedings of the 7th Workshop on Building and Using Comparable Corpora (BUCC)*, Reykjavik, Iceland.

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*, pages 48–57, Atlanta, Georgia, June. Association for Computational Linguistics.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, and Jörg Tiedemann. 2014. A report on the DSL shared task 2014. In *Proceedings of the 1st Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*.

Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richard Schwartz, John Makhoul, Omar F. Zaidan, and Chris Callison-Burch. 2012. Machine translation of arabic dialects. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 49–59, Montréal, Canada, June. Association for Computational Linguistics.