Experiments in Discriminating Similar Languages

Cyril Goutte, Serge Léger
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 (NRC) Canada for the 2015 shared task on *Discriminating between similar languages*. The NRC system uses various statistical classifiers trained on character and word ngram features. Predictions rely on a two-stage process: we first predict the language group, then discriminate between languages or variants within the group. This year, we focused on two issues: 1) the ngram generation process, and 2) the handling of the anonymized ("blinded") Named Entities. Despite the slightly harder experimental conditions this year, our systems achieved an average accuracy of 95.24% (closed task) and 95.65% (open task), ending up second or (close) third on the closed task, and first on the open task.

1 Introduction

Although language identification is largely considered a solved problem in the general setting, a number of frontier cases are still under study. For example, when little data is available (eg single twitter post), when the input is mixed or when discriminating similar languages or language variants.

The *Discriminating between similar languages* (DSL) shared task offers precisely such a situation, by offering an interesting mix of close languages and variants, and relatively short, one-sentence texts. This year, four groups contain similar languages:

- Bosnian, Croatian and Serbian;
- Indonesian and Malaysian;
- Czech and Slovakian;
- Bulgarian and Macedonian.

Two groups contain variants of the same language:

- Portuguese: European vs. Brazilian;
- Spanish: European vs. Argentinian.

In addition, instances to classify are single sentences, a more realistic and challenging situation than full-document language identification.

There are two interesting additions to the 2015 challenge. A second test set, with Named Entity anonymized, was added to evaluate the influence of local information on the predictions. In addition, sentences from "other", unknown languages were added to the test sets. This means that group/language prediction is not limited to the 13 languages in the training set.

Following some good results at last year's evaluation (Goutte et al., 2014; Zampieri et al., 2014), we took part to this years evaluation in order to see how our system would handle the additional language pair, and the two challenges of anonymized named entities and more varied test data. In addition, we wanted to further explore the way character ngrams should be more efficiently extracted from the raw text.

The overall longer term motivation is to use language and variant detection to help natural language processing, for example in machine translation (Zbib et al., 2012). Discriminating similar languages may also be a first step to identify code switching in short messages (Elfardy et al., 2013).

The following section describes the models we used, and the features we extracted from the data. We then briefly describe the data we trained on (Section 3), and summarize our experimental results in Section 4.

2 Models

Our approach relies on a two-stage process. We first predict the language group, then discriminate the languages or variants (which for convenience

we simply all call "language" from now on) within the group. This approach works best if the first stage (i.e. group) classifier 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, 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 first describe the features we extract from the data (Section 2.1). We then provide a quick overview of how the probabilistic group classifier works (Section 2.2). Finally, we describe the within-group language predictors in Section 2.3.

2.1 Features

Based on previous experience, we focus on character and word ngrams as features. We generate several feature spaces, depending on the basic sequence unit (character or word), size of the ngram (N) and way to extract it from the sentence to classify.

bowN: Within-sentence consecutive subsequence of N words. We focus on unigrams (bag of words) and bigrams, as higher orders seem to degrade performance. Bigrams use special tokens to mark begining and end of sentences.

charN: Character ngrams, extracted from the complete sentence including whitespaces and punctuation.

pcharN: Character ngrams, extracted from the complete sentence including whitespaces, but removing the punctuation.

scharN: Within-word character ngrams, after removing punctuation. Word boundaries are included, but ngrams are not extracted over two or more words. Words smaller than the ngram size are include, e.g. "I" yields the ngram "_I_" for $N \geq 3$.

For character ngrams, we generated all feature spaces corresponding to $N = 2 \ldots 6$, while we limit word ngrams to `bow1` and `bow2`. We index all ngrams observed at least once in the entire collection.

2.2 Group Prediction

Predicting the language group, including **X**, is a 7-way classification task. For this first stage, we use the same probabilistic model as last year (Gaussier et al., 2002; Goutte et al., 2014). This model offers a convenient and fast way to handle the group prediction, and its performance on last years' data proved excellent. 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:

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

where $P(w|c)$ is the profile for class c, ie the probability that each word[1] 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. This is a supervised version of the *Probabilistic Latent Semantic Analysis* model (Hofmann, 1999), similar to *Naïve Bayes* (McCallum and Nigam, 1998), except that instead of sampling the class once per document and generating all words from that class, this model can resample the class for each word in the document. This results in a much more flexible model, and higher performance.

Model estimation is done by maximum likelihood and is identical to *Naïve Bayes*:

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

Model behaviour depends solely on this set of class profile vectors. They provide lexical probabilities for each class. For predicting class assignment for a new document, we 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 (Demp-

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

ster et al., 1977). The resulting iterative, fixed-point equation is:

$$P(c|\widetilde{d}) \leftarrow P(c|\widetilde{d}) \sum_w \frac{n(w,\widetilde{d})}{|\widetilde{d}|} \frac{P(w|c)}{\sum_c P(c|\widetilde{d})P(w|c)}, \quad (3)$$

where $|\widetilde{d}| = \sum_w n(w,\widetilde{d})$ 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, model parameters are estimated using Eq. 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. Although this phase is slower than training, it may be easily and efficiently parallelized on, e.g. multicore architecture.

Note that the 7-way group prediction task is, in practice, handled using a 14-class model predicting the languages (including "Other"), and mapping the predictions to the 7 groups.

2.3 Language Prediction

Once the group is predicted from the previous stage, within-group language prediction becomes a binary classification problem in most groups, or a 3-way classification problem in group **A**.

For groups B to G, we rely on Support Vector Machines, powerful binary discriminative classifiers which typically perform very well on text. In our experiments, we use the SVM$_{light}$ (Joachims, 1998) implementation. We trained a binary SVM on each of the feature spaces described in Section 2.1. The training examples are limited to the two languages within the group under consideration. Each SVM is therefore trained on a smaller document set, making training more manageable. 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 for the model.

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.

2.4 Voting

Each feature space (Section 2.1) yields a set of group and language classifiers. Each may yield different performance and make different mistakes. There are several ways to combine different feature sets.

One method is to concatenate features into a larger, single feature space on which classifiers are trained. Choosing the feature spaces to combine and concatenating them poses some (mild) computational and combinatorial problems. In addition, we did not find this approach particularly effective in our 2014 experiments, so we did not consider it this year.

Another approach is to combine outputs from models trained on different feature spaces. This can be done by training a combination model using these outputs as its inputs, as in, e.g., stacked generalization (Wolpert, 1992). Of course, this requires training an additional model, setting aside data to estimate it, etc. Previous investigations (Goutte et al., 2013) suggest that a simpler approach is actually more effective. We simply combine the output of models by voting: for a given set of models trained on different feature spaces, each model votes for the class it predicts, and the final prediction goes to the class with the most votes. In order to simplify the choice of the set of models, we rank all feature spaces according to their cross-validated prediction error, and pick the number of models that yields the highest performing vote, again according to cross-validation. This simple approach is surprisingly and consistently effective. One explanation is that prediction errors for different models tend to be *independent*, so that these errors usually don't conspire towards the wrong prediction.

3 Data

For the **closed task** experiments, we used the DSLCC v.2.0 corpus provided by the organizers (Zampieri et al., 2015). This corpus covers 13 languages in 6 groups, plus "Other". Although the

number of groups and languages is similar to last year's shared task (Zampieri et al., 2014), the English group was dropped in favour of a new group for Bulgarian and Macedonian, called **G** in Table 1. For the closed task experiment, no other resource of any sort was used.

For the **open task**, we combined the DSLCC v.2.0 data used for the closed task with the relevant portion of the DSLCC v.1.0 provided last year (Tan et al., 2014). Specifically, we ignore the two variants of English from the 2014 collection, and added the rest to the 2015 training data. We actually tried to use the DSLCC v.2.1 collection, with the additional Mexican Spanish and Macanese Portuguese data, to check whether this would help improve the group prediction (first stage), but that did not help according to the cross-validation estimator.

Note that the data is nicely balanced across classes. We used a stratified 10-fold cross-validation estimator, respecting the class proportions across folds, in order to estimate prediction performance. The data provided as *development* set was used as one fold in this estimator.

Table 1 shows the size of the training and test sets. Two test sets were provided: test set A contains original unmodified sentences, while test set B was modified to replace named entities with a placeholder (`#NE#`). Our closed task system used only the "Train 2015" data, while the open task added the "Train 2014" data to estimate models.

One key difference between the 2014 and 2015 data is the sentence length. In 2015, all sentences have between 20 and 100 words, while in 2014, they had up to 600 words in some groups.

4 Results

4.1 Submitted Systems

We ended up submitting two runs to each of the tasks (close and open), and each of the test sets (A and B), resulting in eight runs in total. The difference between the closed and open tasks runs is simply the data used for training, as explained in the previous section.

For test set A, our first run is the best system using a single feature space, as estimated by 10-fold cross-validation, Our second run is the best voting combination, again estimated by cross-validation.

For test set B, we focused on feature spaces that would be relatively insensitive to the anonymized named entity. We removed the `#NE#` placeholder in test set B before extracting the `bow1` and all `scharN` ngram features. Our first run is the best system using only character ngrams, which ends up being `schar6` in all cases. Our second run is trained on bag of words, `bow1`. No voting combination was used on test set B, resulting as expected in lower performance.

Task	closed				open			
Test set	A		B		A		B	
Run #	1	2	1	2	1	2	1	2
# errors	3	3	5	4	1	2	2	2

Table 2: Number of errors (out of 14000 test sentences) made by the group-level classifier for the eight runs (two runs per task and per test set).

4.2 Group Prediction

We first look at the number of mistakes made by the group-level predictor. As mentioned above, this is important because the language prediction further down the line will be unable to recover from mistakes made at the group level.

The number of group-level errors varies depending on the task (closed or open) and the feature space used (best or vote). On the closed task, the cross-validated accuracy is 99.971% for the best feature space (`pchar6`) and 99.976% for the best vote (`pchar6`, `bow1` and `char6`), corresponding to 80 and 67 errors, respectively, out of 280,000 examples. For the open task, the accuracy is 99.977% for the best and 99.979% for the vote, i.e. respectively 64 and 59 errors. Most of the errors are caused by **X** (Other) documents being predicted in the **E** (Spanish) group.

Table 2 shows the number of actual prediction errors made on the test sets by the group-level classifiers for each of our eight runs. Note that in last year's shared task, a single test document was incorrectly classified at the group level. This year shows a large increase, especially on test set B, for which we did not have any example document before the test phase. The open task, using more data, allows for slightly better performance.

4.3 Language Prediction

Tables 3 and 4 show all results obtained by our runs, as well as the results per group. The overall picture is similar to last year's evaluation. Group **A** is the hardest, followed by the Spanish (**E**) and Portuguese (**D**) varieties groups. Groups **B** and

Group	Language or Variety	# sentences Train 2015	Train 2014	Test sets A	B
A	Bosnian	20,000	20,000	1000	1000
	Croatian	20,000	20,000	1000	1000
	Serbian	20,000	20,000	1000	1000
B	Indonesian	20,000	20,000	1000	1000
	Malaysian	20,000	20,000	1000	1000
C	Czech	20,000	20,000	1000	1000
	Slovak	20,000	20,000	1000	1000
D	Brazil Portuguese	20,000	20,000	1000	1000
	Portugal Portuguese	20,000	20,000	1000	1000
E	Argentine Spanish	20,000	20,000	1000	1000
	Spain Spanish	20,000	20,000	1000	1000
G	Bulgarian	20,000	-	1000	1000
	Macedonian	20,000	-	1000	1000
X	Others ("xx")	20,000	-	1000	1000

Table 1: Number of sentences in the Discriminating Similar Languages Corpus Collections (DSLCC) provided for the 2014 and 2015 shared tasks, plus test sets with (A) or without (B) named entities, across groups and languages. The system for the closed task was trained on 2015 data only, the system for the open task was trained on 2014 and 2015 data.

Task	closed							
Test set	A				B			
Run #	1		2		1		2	
	Acc.	#err	Acc.	#err	Acc.	#err	Acc.	#err
Group A	88.53	344	89.90	303	87.00	390	84.57	463
Group B	99.30	14	99.35	13	98.80	24	98.90	22
Group C	99.95	1	99.95	1	100.0	0	99.95	1
Group D	92.40	152	92.70	146	86.75	265	86.15	277
Group E	89.40	212	89.95	201	85.20	296	84.35	313
Group G	99.95	1	99.95	1	100.0	0	100.0	0
Overall	94.82	725	**95.24**	666	**93.01**	979	92.30	1078

Table 3: Language prediction test accuracy for the **closed task**, both test sets and all runs. Our best results are in bold (tied for second overall). Overall #err is larger than column sum due to "Other".

Task	open							
Test set	A				B			
Run #	1		2		1		2	
	Acc.	#err	Acc.	#err	Acc.	#err	Acc.	#err
Group A	89.90	303	90.43	287	87.47	376	85.87	424
Group B	99.50	10	99.60	8	98.65	27	98.85	23
Group C	99.95	1	100.0	0	100.0	0	100.0	0
Group D	92.90	142	93.05	139	87.85	243	87.35	253
Group E	90.90	182	91.35	173	86.30	274	85.35	293
Group G	99.95	1	99.95	1	100.0	0	100.0	0
Overall	95.43	640	**95.65**	609	**93.41**	922	92.89	995

Table 4: Language prediction test accuracy for the **open task**, both test sets and all runs. Our best results are in bold (first overall). Overall #err is larger than column sum due to "Other".

especially **C** and **G** are relatively easy, reaching above 99% performance.

Performance is clearly impacted by removing the named entities, as shown by the results on test set B. This degrades the accuracy by around 2%. This clearly shows that named entities help language detection. In our case, this effect is somewhat amplified by the fact that we did not run a voting system on test set B. As shown by the test set A results, voting brings us 0.25-0.5% improvements in accuracy. On the other hand, it is slightly surprising that on the two groups with highest performance (**C** and **G**), we get perfect performance on test set B while we make one mistake on test set A. Of course this could be due to sample variation, as test set A and B are different.

In last year's evaluation (Zampieri et al., 2014), two groups (Lui et al., 2014; King et al., 2014) compiled additional textual material in several languages in order to compete in the open track. Their submission on the open track turned out several points of accuracy *lower* than their performance on the closed track. By contrast, our inclusion of the DSLCC v.1.0 from last year in our open submission helped us consistently reduce the number of errors by about 10% (a boost of at least 0.4% in accuracy). This may be due to the fact that despite the differences in sentence length, DSLCC v.1.0 was fairly similar to this year's corpus, and helped more than independently acquired material. This is a typical domain adaptation effect, often observed in natural language processing.

We also note that the final test accuracy on test set A was very well estimated by the 10-fold cross-validation, always with 0.5% and typically much less.

5 Conclusions

We described the National Research Council's entry to the second shared task on *Discriminating between similar languages*. Our system uses a fairly straightforward processing and modeling approach, building a two stage predictor relying on a probabilistic document classifier to predict the group, and Support Vector Machines to identify the language within each group. We tested various word and character ngram features. Group-level classification was very accurate, making only a handful of mistakes mostly due to the presence of confounding documents from other languages. Our top system yields an average accuracy of 95.65% on test set A (open task), the top result (by a hair) reported on this new collection. Performance on test set B is clearly impacted by the lack of named entities, degrading average accuracy by about 2%. On the closed task, accuracy is 0.4% lower

Acknowledgments

This work was partly supported by the National Research Council programs Multimedia Analytic Tools for Security (MATS) and Learning and Performance Support Systems (LPSS).

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, Serge Léger, and Marine Carpuat. 2014. The NRC system for discriminating similar languages. In *Proceedings of the 1st Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*, Dublin, Ireland.

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.

Ben King, Dragomir Radev, and Steven Abney. 2014. Experiments in sentence language identification with groups of similar languages. In *Proceedings of the 1st Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*, Dublin, Ireland.

Marco Lui, Ned Letcher, Oliver Adams, Long Duong, Paul Cook, and Timothy Baldwin. 2014. Exploring methods and resources for discriminating similar languages. In *Proceedings of the 1st Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*, Dublin, Ireland.

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.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

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)*.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the DSL shared task 2015. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (LT4VarDial)*, Hissar, Bulgaria.

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.