Language Identification using Classifier Ensembles

Conference Paper · September 2015

CITATIONS
4

READS
247

2 authors, including:

Shervin Malmasi
Macquarie University
25 PUBLICATIONS **115** CITATIONS

SEE PROFILE

Language Identification using Classifier Ensembles

Shervin Malmasi
Centre for Language Technology
Macquarie University
Sydney, NSW, Australia
shervin.malmasi@mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
Sydney, NSW, Australia
mark.dras@mq.edu.au

Abstract

In this paper we describe the language identification system we developed for the Discriminating Similar Languages (DSL) 2015 shared task. We constructed a classifier ensemble composed of several Support Vector Machine (SVM) base classifiers, each trained on a single feature type. Our feature types include character 1–6 grams and word unigrams and bigrams. Using this system we were able to outperform the other entries in the closed training track of the DSL 2015 shared task, achieving the best accuracy of 95.54%.

1 Introduction

Language Identification (LID) is the task of determining the language of a given text, which may be at the document, sub-document or even sentence level. Although the task is generally considered to be a solved problem, recently attention has turned to discriminating between close languages or variants. This includes pairings such as Malay-Indonesian and Croatian-Serbian (Ljubesic et al., 2007), or even varieties of one language (British *vs.* American English).

This has motivated the organization of the Discriminating Similar Languages (DSL) 2015 shared task where the aim is to build systems for distinguishing such pairs. The 2015 edition included 14 language classes.

LID has a number of useful applications including lexicography, authorship profiling, machine translation and Information Retrieval. Another example is the application of the output from these LID methods to adapt NLP tools that require annotated data, such as part-of-speech taggers, for resource-poor languages.

2 Related Work

Work in LID dates back to the seminal research of Beesley (1988), Cavnar and Trenkle (1994) and Dunning (1994). Automatic LID methods have since been widely used in NLP. Although LID can be extremely accurate in distinguishing languages that use distinct character sets (e.g. Chinese or Japanese) or are very dissimilar (e.g. Spanish and Swedish), performance is degraded when it is used for discriminating similar languages or dialects. This has led to researchers turning their attention to the sub-problem of discriminating between closely-related languages and varieties.

This issue has been researched in the context of confusable languages, including Malay-Indonesian (Bali, 2006), Farsi-Dari (Malmasi and Dras, 2015a), Croatian-Slovene-Serbian (Ljubesic et al., 2007), Portuguese varieties (Zampieri and Gebre, 2012), Spanish varieties (Zampieri et al., 2013), and Chinese varieties (Huang and Lee, 2008). The task of Arabic Dialect Identification has also drawn attention in the Arabic NLP community (Malmasi et al., 2015a).

This issue was also the focus of the first "Discriminating Similar Language" (DSL) shared task[1] in 2014. The shared task used data from 13 different languages and varieties divided into 6 sub-groups and teams needed to build systems for distinguishing these classes. They were provided with a training and development dataset comprised of 20,000 sentences from each language and an unlabelled test set of 1,000 sentences per language was used for evaluation. Most entries used surface features and many applied hierarchical classifiers, taking advantage of the structure provided by the language family memberships of the 13 classes. More details can be found in the shared task report by Zampieri et al. (2014).

[1] This was part of the Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects, which was co-located with COLING 2014

Language	Code	Train	Dev	Test
Bulgarian	BG	18,000	2,000	1,000
Bosnian	BS	18,000	2,000	1,000
Czech	CZ	18,000	2,000	1,000
Spanish (Argentina)	ES_AR	18,000	2,000	1,000
Spanish (Spain)	ES_ES	18,000	2,000	1,000
Croatian	HR	18,000	2,000	1,000
Indonesian	ID	18,000	2,000	1,000
Malaysian	MY	18,000	2,000	1,000
Macedonian	MK	18,000	2,000	1,000
Portuguese (Brazil)	PT_BR	18,000	2,000	1,000
Portuguese (Portugal)	PT_PT	18,000	2,000	1,000
Slovak	SK	18,000	2,000	1,000
Serbian	SR	18,000	2,000	1,000
Other	XX	18,000	2,000	1,000
Total		252,000	28,000	14,000

Table 1: The languages included in the corpus and the number of sentences in each set.

3 Data

The data for the shared task comes from the DSL Corpus Collection (Tan et al., 2014). The task is performed at the sentence-level and the corpus consists of 294,000 sentences distributed evenly between 14 language classes. The corpus is subdivided into training, development and test sets. The languages and the number of sentences in each set are listed in Table 1.

An interesting addition to this year's data is the inclusion of an "other" class which contains data from various additional languages. The motivation here is to emulate a realistic language identification and see how the systems perform in classifying previously unseen languages.

More details about the data can be found in the shared task overview paper (Zampieri et al., 2015).

4 Method

In this section we describe the general methodology used to construct our system. We use a supervised learning approach based on discriminative classifiers.

4.1 Features

We use two basic classes of surface features: character n-grams (n = 1–6) and word n-grams (n = 1–2).

4.2 Classifier

We use a linear Support Vector Machine to perform multi-class classification in our experiments. In particular, we use the LIBLINEAR[2] package (Fan et al., 2008) which has been shown to be efficient for text classification problems such as this. For example, it has been demonstrated to be a very effective classifier for the task of Native Language Identification (Malmasi and Dras, 2015b; Malmasi et al., 2013) which also relies on text classification methods.

5 Classifier Ensembles

Classifier ensembles are a way of combining different classifiers or experts with the goal of improving accuracy through enhanced decision making. They have been applied to a wide range of real-world problems and shown to achieve better results compared to single-classifier methods (Oza and Tumer, 2008). Through aggregating the outputs of multiple classifiers in some way, their outputs are generally considered to be more robust. Ensemble methods continue to receive increasing attention from researchers and remain a focus of much machine learning research (Woźniak et al., 2014; Kuncheva and Rodríguez, 2014).

Such ensemble-based systems often use a parallel architecture, as illustrated in Figure 1, where the classifiers are run independently and their outputs are aggregated using a fusion method. Other, more sophisticated, ensemble methods that rely on meta-learning may employ a stacked architecture where the output from a first set of classifiers is fed into a second level meta-classifier and so on.

[2]http://www.csie.ntu.edu.tw/%7Ecjlin/liblinear/

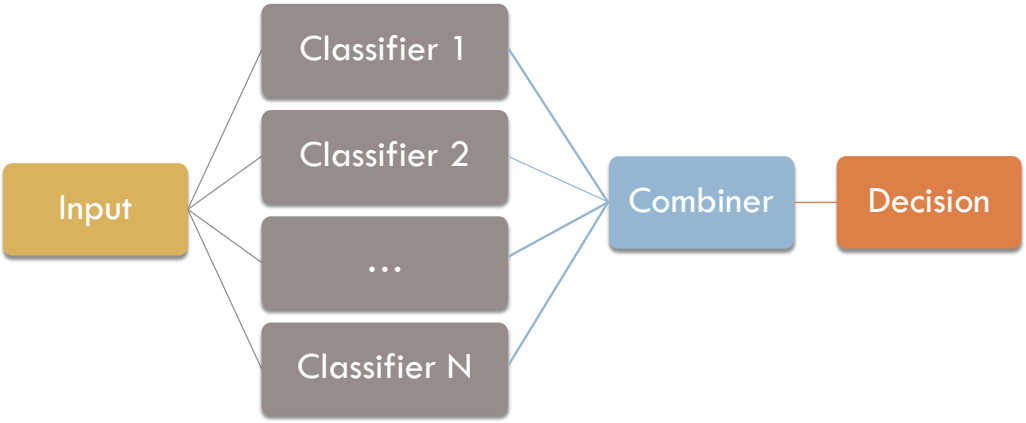

Figure 1: An example of parallel classifier ensemble architecture where N independent classifiers provide predictions which are then fused using an ensemble combination method.

The first part of creating an ensemble is generating the individual classifiers. Various methods for creating these ensemble elements have been proposed. These involve using different algorithms, parameters or feature types; applying different preprocessing or feature scaling methods and varying (*e.g.* distorting or resampling) the training data.

For example, *Bagging* (bootstrap aggregating) is a commonly used method for ensemble generation (Breiman, 1996) that can create multiple base classifiers. It works by creating multiple bootstrap training sets from the original training data and a separate classifier is trained from each one of these sets. The generated classifiers are said to be diverse because each training set is created by sampling with replacement and contains a random subset of the original data. *Boosting* (*e.g.* with the AdaBoost algorithm) is another method where the base models are created with different weight distributions over the training data with the aim of assigning higher weights to training instances that are misclassified (Freund and Schapire, 1996).

As we describe in section 7, each of the base classifiers in our ensemble is trained on a different feature space, as this has proven to be effective.

The second part of ensemble design is choosing a fusion rule to aggregate the outputs from the various learners, this is discussed in the next section.

6 Ensemble Combination Methods

Once it has been decided how the set of base classifiers will be generated, selecting the classifier combination method is the next fundamental design question in ensemble construction.

The answer to this question depends on what output is available from the individual classifiers. Some combination methods are designed to work with class labels, assuming that each learner outputs a single class label prediction for each data point. Other methods are designed to work with class-based continuous output, requiring that for each instance every classifier provides a measure of confidence probability[3] for each class label. These outputs for each class usually sum to 1 over all the classes.

Although a number of different fusion methods have been proposed and tested, there is no single dominant method (Polikar, 2006). The performance of these methods is influenced by the nature of the problem and available training data, the size of the ensemble, the base classifiers used and the diversity between their outputs.

The selection of this method is often done empirically. Many researchers have compared and contrasted the performance of combiners on different problems, and most of these studies – both empirical and theoretical – do not reach a definitive conclusion (Kuncheva, 2014, p 178).

In the same spirit, we experiment with several information fusion methods which have been widely discussed in the machine learning literature. Our selected methods are listed below. Various other methods exist and the interested reader can refer to the exposition by Polikar (2006).

[3] *i.e.* an estimate of the posterior probability for the label. For non-probabilistic classifiers the distance to the decision boundary is used for estimating the decision likelihoods.

6.1 Plurality voting

Each classifier votes for a single class label. The votes are tallied and the label with the highest number[4] of votes wins. Ties are broken arbitrarily. This voting method is very simple and does not have any parameters to tune. An extensive analysis of this method and its theoretical underpinnings can be found in the work of (Kuncheva, 2004, p. 112).

6.2 Mean Probability Rule

The probability estimates for each class are added together and the class label with the highest average probability is the winner. This is equivalent to the probability sum combiner which does not require calculating the average for each class. An important aspect of using probability outputs in this way is that a classifier's support for the true class label is taken in to account, even when it is not the predicted label (*e.g.* it could have the second highest probability). This method has been shown to work well on a wide range of problems and, in general, it is considered to be simple, intuitive, stable (Kuncheva, 2014, p. 155) and resilient to estimation errors (Kittler et al., 1998) making it one of the most robust combiners discussed in the literature.

6.3 Median Probability Rule

Given that the mean probability used in the above rule is sensitive to outliers, an alternative is to use the median as a more robust estimate of the mean (Kittler et al., 1998). Under this rule each class label's estimates are sorted and the median value is selected as the final score for that label. The label with the highest median value is picked as the winner. As with the mean combiner, this method measures the central tendency of support for each label as a means of reaching a consensus decision.

6.4 Product Rule

For each class label, all of the probability estimates are multiplied together to create the label's final estimate (Polikar, 2006, p. 37). The label with the highest estimate is selected. This rule can provide the best overall estimate of posterior probability for a label, given that the individual estimates are accurate. A trade-off here is that this method is very sensitive to low probabilities: a single low score for a label from any classifier will essentially eliminate that class label.

[4]This differs with a *majority* voting combiner where a label must obtain over 50% of the votes to win. However, the names are sometimes used interchangeably.

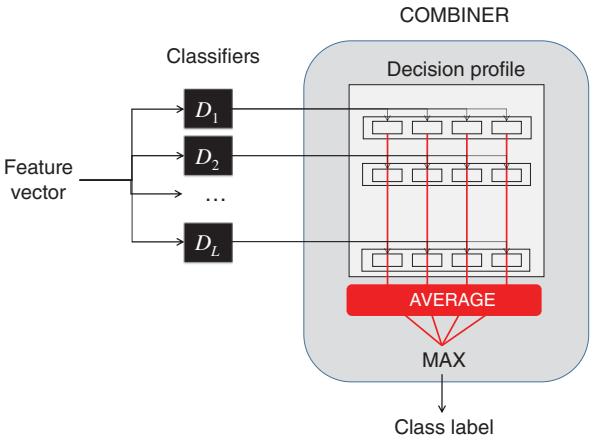

Figure 2: An example of a mean probability combiner. The feature vector for a sample is input to L different classifiers, each of which output a vector of confidence probabilities for each possible class label. These vectors are combined to form the decision profile for the instance which is used to calculate the average support given to each label. The label with the maximum support is then chosen as the prediction. Image reproduced from (Kuncheva, 2014).

6.5 Highest Confidence

In this simple method, the class label that receives the vote with the largest degree of confidence is selected as the final prediction (Kuncheva, 2014, p. 150). In contrast to the previous methods, this combiner disregards the consensus opinion and instead picks the prediction of the expert with the highest degree of confidence.

6.6 Borda Count

This method works by using each classifier's confidence estimates to create a ranked list of the class labels in order of preference, with the predicted label at rank 1. The winning label is then selected using the Borda count[5] algorithm (Ho et al., 1994). The algorithm works by assigning points to labels based on their ranks. If there are N different labels, then each classifiers' preferences are assigned points as follows: the top-ranked label receives N points, the second place label receives

[5]This method is generally attributed to Jean-Charles de Borda (1733–1799), but evidence suggests that it was also proposed by Ramon Llull (1232–1315).

$N-1$ points, third place receives $N-2$ points and so on with the last preference receiving a single point. These points are then tallied to select the winner with the highest score.

The most obvious advantage of this method is that it takes into account each classifier's preferences, making it possible for a label to win even if another label received the majority of the first preference votes.

6.7 Oracle

We use an "Oracle" combiner as one possible approach to estimating the upper-bound for classification accuracy. This method has previously been used to analyze the limits of majority vote classifier combination (Kuncheva et al., 2001). The oracle will assign the correct class label for an instance if at least one of the constituent classifiers in the ensemble produces the correct label for that data point. Oracles are usually used in comparative experiments and to gauge the performance and diversity of the classifiers chosen for an ensemble (Kuncheva, 2002; Kuncheva et al., 2003). They can help us quantify the *potential* upper limit of an ensemble's performance on the given data and how this performance varies with different ensemble configurations (Malmasi et al., 2015b).

7 Systems

We test three different systems in our submissions to the shared task, as outlined here.

7.1 System 1

We train a single model based on a simple combination of all our feature types into a single feature space. The model has approximately 13.6 million features. This was the first system that we built and it achieved very good results of 94-95% during testing. It was selected as our first submission.

7.2 System 2

The second system is an ensemble classifier, as described in section 5. The aim here was to improve over the single classifier system described in section 7.1. Each base classifier in the ensemble is trained on a separate feature type, resulting in a total of eight classifiers in the system.

During the development of our system we tested the six ensemble fusion methods described in section 6. Our experiments with the training and development data showed that the mean probability combiner yielded the best accuracy.

We achieved an accuracy of 95.5% on the development set against an oracle accuracy of 99%, showing that the combiner was very close to the upper-bound of possible classification accuracy. This result was slightly better than that of System 1, so this method was selected for our second submission. The results from the other combiners were also in a similar range, but we used the mean probability combiner for our second system.

7.3 System 3

Our final system is identical to the second system in its method and setup with the exception that some weak and redundant features were removed. We suspected that there may be some redundancy in the large number of character n-gram features and removing these might increase the diversity, and thus accuracy, of the ensemble.

Using the feature analysis methodology outlined by Malmasi and Cahill (2015), we analyzed the feature interactions using the training and development sets. This methodology uses Yule's Q-coefficient statistic (Yule, 1912), which can be a useful measure of pairwise dependence between two classifiers (Kuncheva et al., 2003). This notion of dependence relates to complementarity and orthogonality, and is an important factor in combining classifiers (Lam, 2000). The calculated Q-coefficient ranges between -1 to $+1$, where -1 signifies negative association, 0 indicates no association (independence) and $+1$ means perfect positive correlation (dependence). We apply this method to our ensemble to calculate the dependence between the classifiers. The results for the analysis are shown as a heat map in Figure 3.

We see that the predictions obtained using character unigrams are very diverse to the other features, as noted by the low Q-coefficient. This diversity is a result of character unigrams being a weak feature: they only achieve around 76% accuracy whereas most other feature types can obtain $> 90\%$ accuracy. As a result we removed this feature from the ensemble.

Character bigrams are diverse and also have a higher accuracy, so they were retained. Character trigrams are very similar to 4-grams and their accuracies are close, so we remove the trigrams. The same applies to character 5- and 6-grams, and we decided to remove the 5-grams. Character 4-grams were retained since they had good accuracy and diversity, *e.g.* with word bigrams.

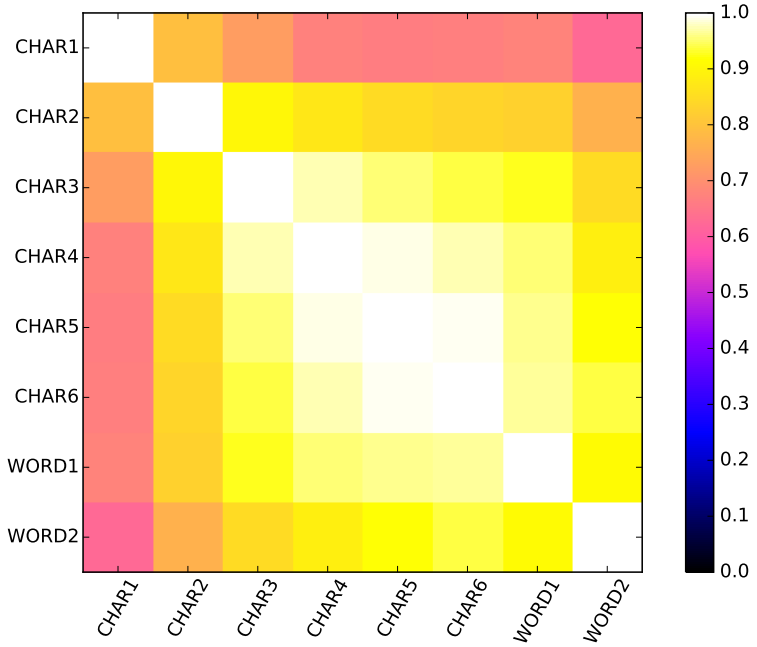

Figure 3: The matrix of pairwise Q-coefficient values between our feature types, displayed as a heat map. Smaller values indicate lower dependence between their predictions.

	Normal		NE Removed	
	Rank	**Accuracy**	**Rank**	**Accuracy**
Random Baseline	—	7.14%	—	7.14%
System 1	3	95.31%	2	93.88%
System 2	2	95.44%	3	93.73%
System 3	**1**	**95.54%**	**1**	**94.01%**

Table 2: Results for our three system on the test set. The accuracy and rank among all systems in the shared task are shown. Our optimized ensemble ranked first in both tasks.

To recap, our third system is a modification of the second system where we remove character 1-, 3- and 5-grams in order to increase the ensemble diversity. This reduced ensemble was chosen as our third submission as it achieved slightly higher results than the full ensemble during development.

8 Results

We entered our systems in both sub-tasks of the closed training track. We did not enter the open training track of the competition. The first subtask (the "normal" task) required our system to classify 14,000 unlabelled sentences. The second task was also similar, but it used a different set of sentences which also had all named entities (NE) removed ("NE Removed" task). This is because it is assumed that features related to NEs can strongly influence the results.

Our systems took the top three places for both subtasks. The results and rankings for each system are shown in Table 2. We note that System 3 — the optimized ensemble — was the winning entry for both tasks. This comports with our initial tests where it was our best system during development.

The confusion matrix for our best results in the normal task are shown in Figure 4. We achieved a perfect 100% accuracy for four classes: Czech (CZ), Macedonian (MK), Slovak (SK) and Other (XX). Bulgarian (BG) was also close with only a single sentence being misclassified as Macedonian. These results suggest that confusion between Bulgarian–Macedonian and Czech–Slovak is not a significant issue here. The greatest confusion is between the Bosnian–Croatian–Serbian[6] group as well as the Spanish and Portuguese dialect pairs. Bosnian is the worst performing language among the 14 classes.

We also analyze the learning rate for the features in our system. The cross-validation accuracy for four different feature types is shown in Figure 5. Higher order character n-grams seem to outperform word n-grams. Word bigrams are lower in accuracy and have a steeper learning rate.

[6]We also observe that Bosnian is the most confused class among the three while Serbian has the least errors.

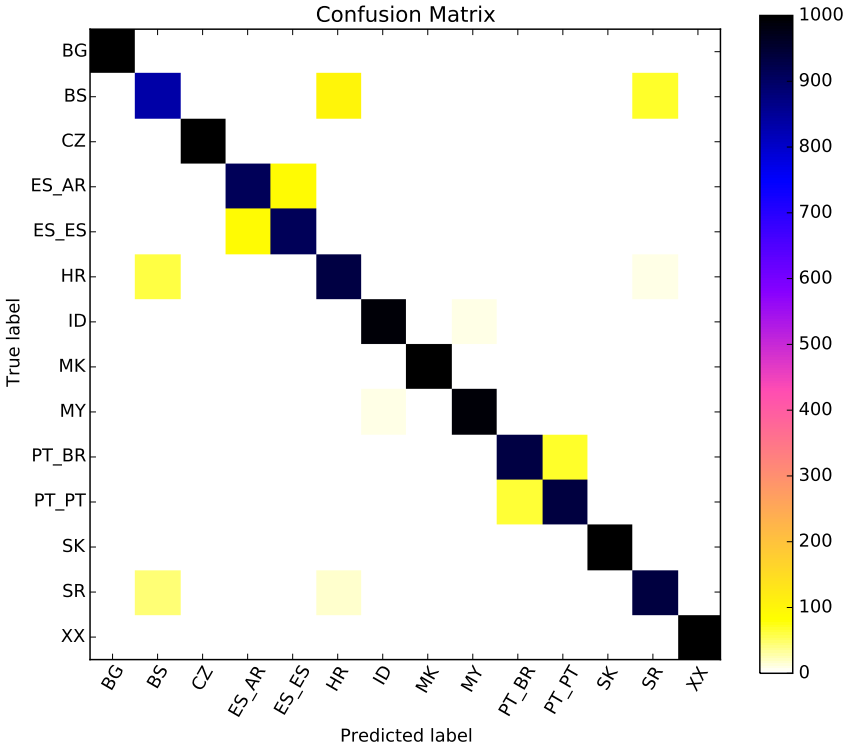

Figure 4: The confusion matrix for our results on the test set (normal task).

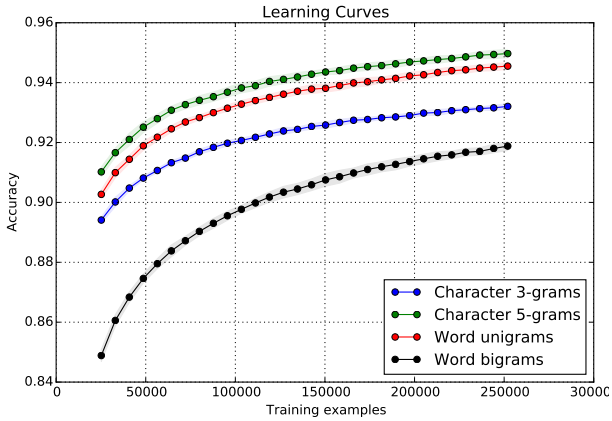

Figure 5: Learning curves for some of our features based on cross-validation accuracy. We observe that character n-grams perform better than word-based n-grams. The accuracy does not plateau with the entire training data used.

We also observe that accuracy increases continuously as the training data is increased. This suggests that despite the already large size of the training set, there is still room for further improvement by adding more data. However, this would also result in an increase in the size of our feature space, which is already quite large due to the prodigious growth rate of the larger order character n-grams.

9 Discussion and Conclusion

In this work we demonstrated the utility of classifier ensembles for text classification. Using an ensemble composed of base classifiers trained on character 1–6 grams and word unigrams and bigrams, we were able to outperform the other entries in the closed track of the DSL 2015 shared task.

A crucial direction for future work is the investigation of methods to reduce the confusion between these three groups of classes.

In this work we did not experiment with feature selection methods to evaluate if this can further enhance performance, or at least efficiency by reducing the dimensionality of the feature space. One weakness of our system may be the very high dimensionality of the feature space with almost 14 million features. Having such a large number of features can be inefficient and may impede the use of our system for real-time applications.

References

Ranaivo-Malançon Bali. 2006. Automatic Identification of Close Languages–Case Study: Malay and Indonesian. *ECTI Transaction on Computer and Information Technology*, 2(2):126–133.

Kenneth R Beesley. 1988. Language identifier: A computer program for automatic natural-language identification of on-line text. In *Proceedings of the 29th Annual Conference of the American Translators Association*, volume 47, page 54. Citeseer.

Leo Breiman. 1996. Bagging predictors. In *Machine Learning*, pages 123–140.

William B. Cavnar and John M. Trenkle. 1994. N-Gram-Based Text Categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Ted Dunning. 1994. *Statistical identification of language*. Computing Research Laboratory, New Mexico State University.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874.

Yoav Freund and Robert E Schapire. 1996. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156.

Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. 1994. Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(1):66–75.

Chu-Ren Huang and Lung-Hao Lee. 2008. Contrastive Approach towards Text Source Classification based on Top-Bag-Word Similarity.

Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. 1998. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239.

Ludmila I Kuncheva and Juan J Rodríguez. 2014. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, 38(2):259–275.

Ludmila I Kuncheva, James C Bezdek, and Robert PW Duin. 2001. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314.

Ludmila I Kuncheva, Christopher J Whitaker, Catherine A Shipp, and Robert PW Duin. 2003. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications*, 6(1):22–31.

Ludmila I Kuncheva. 2002. A theoretical study on six classifier fusion strategies. *IEEE Transactions on pattern analysis and machine intelligence*, 24(2):281–286.

Ludmila I Kuncheva. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons.

Ludmila I Kuncheva. 2014. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, second edition.

Louisa Lam. 2000. Classifier combinations: implementations and theoretical issues. In *Multiple classifier systems*, pages 77–86. Springer.

Nikola Ljubesic, Nives Mikelic, and Damir Boras. 2007. Language indentification: How to distinguish similar languages? In *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on*, pages 541–546. IEEE.

Shervin Malmasi and Aoife Cahill. 2015. Measuring Feature Diversity in Native Language Identification. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, Denver, Colorado, June. Association for Computational Linguistics.

Shervin Malmasi and Mark Dras. 2015a. Automatic Language Identification for Persian and Dari texts. In *Proceedings of the 14th Conference of the Pacific Association for Computational Linguistics (PACLING 2015)*, pages 59–64, Bali, Indonesia, May.

Shervin Malmasi and Mark Dras. 2015b. Large-scale Native Language Identification with Cross-Corpus Evaluation. In *Proceedings of NAACL-HLT 2015*, Denver, Colorado, June. Association for Computational Linguistics.

Shervin Malmasi, Sze-Meng Jojo Wong, and Mark Dras. 2013. NLI Shared Task 2013: MQ Submission. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 124–133, Atlanta, Georgia, June. Association for Computational Linguistics.

Shervin Malmasi, Eshrag Refaee, and Mark Dras. 2015a. Arabic Dialect Identification using a Parallel Multidialectal Corpus. In *Proceedings of the 14th Conference of the Pacific Association for Computational Linguistics (PACLING 2015)*, pages 209–217, Bali, Indonesia, May.

Shervin Malmasi, Joel Tetreault, and Mark Dras. 2015b. Oracle and Human Baselines for Native Language Identification. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, Denver, Colorado, June. Association for Computational Linguistics.

Nikunj C Oza and Kagan Tumer. 2008. Classifier ensembles: Select real-world applications. *Information Fusion*, 9(1):4–20.

Robi Polikar. 2006. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 6(3):21–45.

Liling Tan, Marcos Zampieri, Nikola Ljubešic, 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*, pages 11–15, Reykjavik, Iceland.

Jörg Tiedemann and Nikola Ljubešić. 2012. Efficient discrimination between closely related languages. In *Proceedings of COLING 2012*, pages 2619–2634.

Michał Woźniak, Manuel Graña, and Emilio Corchado. 2014. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17.

George Udny Yule. 1912. On the methods of measuring association between two attributes. *Journal of the Royal Statistical Society*, pages 579–652.

Marcos Zampieri and Binyam Gebrekidan Gebre. 2012. Automatic identification of language varieties: The case of Portuguese. In *KONVENS2012-The 11th Conference on Natural Language Processing*, pages 233–237. Österreichischen Gesellschaft für Artificial Intelligende (ÖGAI).

Marcos Zampieri, Binyam Gebrekidan Gebre, and Sascha Diwersy. 2013. N-gram language models and POS distribution for the identification of Spanish varieties. In *Proceedings of TALN 2013*, pages 580–587.

Marcos Zampieri, Liling Tan, Nikola Ljubešic, and Jörg Tiedemann. 2014. A report on the DSL shared task 2014. *COLING 2014*, page 58.

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.