Discriminating similar languages: experiments with linear SVMs and neural networks

Çağrı Çöltekin
Department of Linguistics
University of Tübingen, Germany
ccoltekin
@sfs.uni-tuebingen.de

Taraka Rama
Department of Linguistics
University of Tübingen, Germany
taraka-rama.kasicheyanula
@uni-tuebingen.de

Abstract

This paper describes the systems we experimented with for participating in the discriminating between similar languages (DSL) shared task 2016. We submitted results of a single system based on support vector machines (SVM) with linear kernel and using character ngram features, which obtained the first rank at the closed training track for test set A. Besides the linear SVM, we also report additional experiments with a number of deep learning architectures. Despite our intuition that non-linear deep learning methods should be advantageous, linear models seem to fare better in this task, at least with the amount of data and the amount of effort we spent on tuning these models.

1 Introduction

Automatic language identification if often considered a solved task. Very high levels of accuracies in automatic identification of languages from text had been reported in studies over two decades ago (Beesley, 1988; Cavnar and Trenkle, 1994; Dunning, 1994). For example, Dunning (1994) reports over $99\,\%$ accuracy for test strings of 100 characters, the reported accuracy goes up to $99.90\,\%$ for 500-character strings. Even short strings of 20 characters were enough for over $90\,\%$ accuracy. The results above were obtained when a training set of $50\mathrm{k}$ characters were considered 'large' training data, and many of the machine learning methods were in their infancy. Considering the amount of data, computation power and the methods we have at hand today, the automatic language identification task is, indeed, an almost solved problem. However, there are at least two cases where we are not close to the solution yet. The first is when the languages to be discriminated are closely related (Tiedemann and Ljubešić, 2012; Zampieri et al., 2014; Zampieri et al., 2015), and the second is when the documents of interest contain multiple languages, including code mixing or code switching (Nguyen and Dogruöz, 2013; Lui et al., 2014). Discriminating between Similar Languages (DSL) shared task (Malmasi et al., 2016) aims to address the first issue.

This paper describes the models we experimented with for participating in the DSL shared task. In this work, we describe and report results from two families of models. The first family is the linear models with character-ngram features, including the linear support vector machine (SVM) model which obtained the first rank at the closed training track for test set A, and obtained fair results in other test sets despite the fact that it was not particularly optimized for them. In our experiments, the (simple) linear models with character ngram features were proven difficult to beat.

The second family of models we experimented with are a number of deep neural network architectures. These models has been our initial motivation for participating in the shared task. Besides their recent success in many natural language processing methods, these models are interesting for discriminating between similar languages because of (at least) two reasons. First, it seems success in discriminating between distant/different languages and discriminating between similar languages require different types of models and/or features. This observation is supported by the fact that one of the most popular and successful approaches in earlier DSL shared tasks has been hierarchical systems that use different models

Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects,
pages 15–24, Osaka, Japan, December 12 2016.

for discriminating language groups and individual languages within each group (Goutte et al., 2014; Goutte and Léger, 2015; Fabra Boluda et al., 2015; Ács et al., 2015). The potential benefit of a deep learning model here is the possibility of building a single (hierarchical) model that addresses both issues jointly.

The second potential benefit of deep learning architectures comes from the fact that, unlike linear models, they can capture non-additive non-linear interactions between input features. For example, although none of the features marked with boldface in (1) below are conclusive for 3-way discrimination between Bosnian (1a), Croatian (1b) and Serbian (1c), a non-linear combination of these features are definitely useful.[1]

(1) a. *Član 3.* ***Svako*** *ima pravo na život, slobodu i* ***ličnu sigurnost****.*

b. *Članak 3.* ***Svatko*** *ima pravo na život, slobodu i* ***osobnu sigurnost****.*

c. *Član 3.* ***Svako*** *ima pravo na život, slobodu i* ***ličnu bezb(j)ednost****.*

Article 3. Everyone has the right to life, liberty and security of person.

As a result, given enough training data and appropriate model architecture, we expect deep networks to perform well in discriminating both languages across the language groups and languages within the language groups. Below, we describe both family of models in detail (Section 2), report results of both on the shared task training data (Section 3), and discuss the results obtained (Section 4).

2 Approach

We originally intended to participate in the shared task using deep neural network models, with the motivations that are discussed in Section 1. However, our efforts did not yield better results than the 'baseline' models that we initially implemented just for the sake of comparison. As a result, we participated in the shared task with our best baseline, which also obtained good results among the other participated systems. In this section, we first briefly described the 'baseline' models that constitute our official participation to the DSL shared task. We also describe some of the deep learning architectures we have experimented with in this section, and report results obtained by both models – linear and deep learning – and compare them in Section 3. The implementations of all models described below are available at `https://doi.org/10.5281/zenodo.163812`.

2.1 Linear models

The results we submitted to the shared task use a multi-class (one-vs-one) support vector machine (SVM) model with linear kernel. Although we experimented with various features, our final model included character ngrams of length one to seven. The features are weighted using sub-linear tf-idf scaling (Jurafsky and Martin, 2009, p.805). The models we describe here are almost identical to the model of Zampieri et al. (2015) in the DSL 2015 shared task. Similar to them, we also experimented with logistic regression as well, using both one-vs-rest and one-vs-one multi-class strategies. In our experiments, different linear models performed comparably. However, the SVM models always performed slightly better than logistic regression models. In this paper, we only describe the SVM models and discuss the results obtained using them.

We did not apply any filtering (e.g., case normalization, tokenization) except truncating the input documents to 70 white-space-separated tokens. In all experiments reported in this paper, we fixed the single model parameter (SVM margin or regularization parameter, C) at 1.00. Unlike results reported by Purver (2014), in our experiments, the model accuracy was not affected drastically with the changes in the regularization parameter within a reasonable range (not reported here). Although stronger regularization was useful for models employing larger number of features, the effect was not noteworthy.

All linear models were implemented with scikit-learn (Pedregosa et al., 2011) and trained and tested using Liblinear back end (Fan et al., 2008).

[1]The example (article 3 of the Universal Declaration of Human Rights) is taken from `https://en.wikipedia.org/wiki/Comparison_of_standard_Bosnian,_Croatian,_Montenegrin_and_Serbian`.

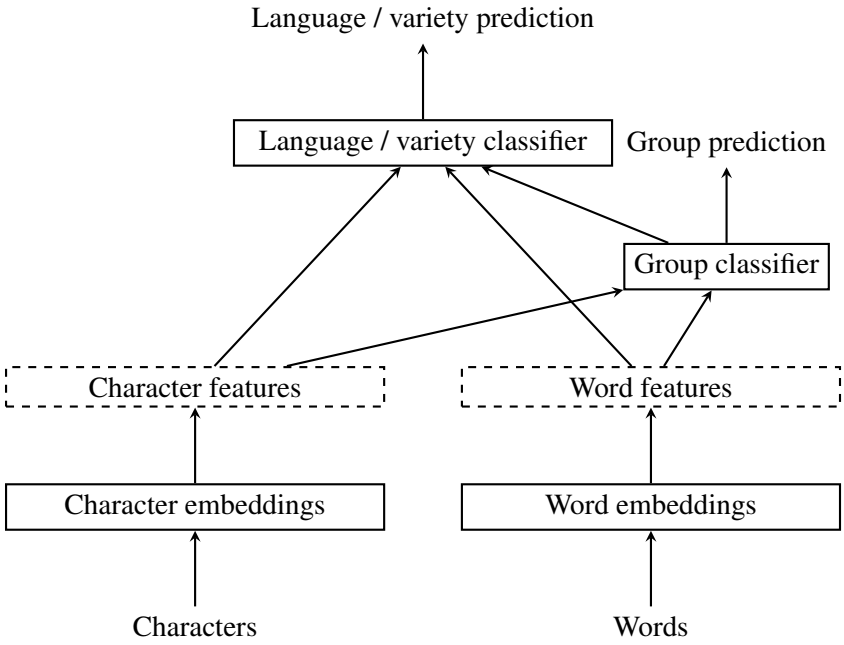

Figure 1: The schematic representation of general neural network architecture.

2.2 Deep learning models

FastText. The FastText model is a recent model proposed by Joulin et al. (2016). The main idea behind this model is to represent a sentence as a sequence of word embeddings (dense vectors) and then employ average the vectors across the sentence to yield a single dense representation of a sentence. Formally, given a sentence of length N, each word w_n is represented as a vector $e_n \in \mathbb{R}^K$ where K is the dimension of the dense representation. At this step, the sentence is now represented as matrix $E \in \mathbb{R}^{k \times n}$. The average pooling step transforms the matrix into a single vector $X \in \mathbb{R}^k$ and $X_k = \frac{\sum E_k}{n}$ where, X_k is the k^{th} element in X. We tested with both character embeddings and word embeddings in our experiments.

This model is similar to the system of Franco-Salvador et al. (2015) in the DSL 2015 shared task, since it represents the documents as an average vectors of their components (characters or words). However, crucially, the embeddings used in this model are learned specifically for the discrimination task rather than being general word vectors capturing syntactic/semantic properties.

Hierarchical character + word models. The general architecture used for our hierarchical network model is presented in Figure 1. In this model, we use both character and word embeddings to train our model. Similar to FastText model discussed above, the embeddings are task-specific. They are trained during learning to discriminate the languages varieties. As a result, the expectation is that the input features (characters words) that are indicative of a particular label (rather than words that are semantically similar) cluster in the same region of the space defined by the embeddings.

The model is an example of multi-label classification. During training, model parameters are optimized to guess both the group, and the specific language variety correctly. Furthermore, we feed the model's prediction of the group to the classifier predicting the specific language variety. For instance, we would use the information that *es-ar* and *es-mx* labels belong to the Spanish group. The intuition behind this model is that, it will use the highly accurate group prediction during test time to tune into features that are useful within a particular language group for predicting individual varieties. In principle, the boxes 'Group classifier' and 'Language / variety classifier' in Figure 1 may include multiple layers for allowing the classifier to generalize based on non-linear combinations in its input features. However, in the experiments reported in this paper, we did not use multiple layers in both classifiers, since it did not improve the results.

The dashed boxes in Figure 1 turn the sequence of word and character embeddings into fixed-size

feature vectors. In principle, any model that extracts useful features from a sequence of embeddings are useful here. The convolutional and recurrent neural networks are typical choices for this step. We have experimented with both methods. However, simple averaging of the embedding as in the FastText model described above performed better. As a result we only report results from a simple feature extraction step where the features are averaged embedding vectors.

The model we use for the results reported in Section 3.3 below has the following components.

1. A character embeddings model and a word embeddings model similar to FastText.

2. A group label classifier is trained on the concatenation of the representation from character and word embeddings.

3. The softmax score of the group classifier is concatenated again with the character and word representations' concatenation to train a final language variety classifier based on softmax classifier.

In the experiments reported below, the documents are padded or truncated to 512 characters for the character embedding input, and they are padded or truncated to 70 tokens for the word embeddings input. We used character embeddings of size 20 and word embeddings of size 32. For both embedding layers, we used dropout with rate 0.20 to prevent overfitting. As noted above, the feature extraction step is only averaging over all embedding vectors. Both classifiers in the figure were single layer networks (with softmax activation function), predicting one-hot representations of groups and varieties. The network was trained using categorical cross entropy loss function for both outputs using Adam optimization algorithm. To prevent overfitting, the training was stopped when validation set accuracy stopped improving. All neural network experiments are realized using Keras (Chollet, 2015) with Tensorflow backend (Abadi et al., 2015).

3 Experiments and results

This section presents the results obtained with the approaches described in Section 2. We first introduce the data sets used briefly. Then we present the result we received from the shared task organizers, followed by the results from the models that we did not submit to the shared task.

3.1 Data

The DSL shared shared task (Malmasi et al., 2016) included two tasks. The data for the first task (*task 1*) comes from Tan et al. (2014), and contains a set of 12 language varieties belonging to 5 groups. The data for the second task (*task 2*) comes from Ali et al. (2016), and contains five varieties of Arabic. The first task included two test sets. Test set A contains in-domain documents including all languages or language varieties listed in Table 1. Test set B contains out-of-domain documents (tweets) in Bosnian, Croatian, Serbian, Brazilian Portuguese and European Portuguese. The test set B comes in two varieties, B1 and B2, differing in the way documents were sampled (Malmasi et al., 2016).

The languages/varieties included in task 1 are presented in Table 1, and the Arabic dialect data for task 2 is presented in Table 2. Besides the codes and short descriptions of the varieties, Table 1 also presents the average document length in number of words and characters for each variety in the training set. Although training set for task 1 is balanced in number of documents, there is a slight amount of imbalance with respect to token and character features available in the training data. The overall average length of the documents in task 1 training data was 34.80 words (sd=14.42) and 185.48 characters (sd=76.52). Besides average number of characters and words, Table 1 also presents the number of documents belonging to each variety in the training set for task 2. The data for task 2 contains ASR transcripts. The lengths of the documents in the task 2 training set vary more. The average length of the documents in task 2 training set is 183.79 (sd=271.81) characters and 41.45 (sd=60.68) tokens. In particular, the task 2 training data consists mainly of short documents ($27\,\%$ of the documents are less than 50 characters, cf. $1.50\,\%$ for task 1 training set). However, there are also occasional very long documents (longest document contains $18\,017$ characters).

Code	Language / variety	Characters	Tokens
bs	Bosnian	168.36	31.13
hr	Croatian	203.56	37.02
sr	Serbian	180.04	34.27
es-ar	Argentinian Spanish	213.11	41.47
es-es	European Spanish	224.49	44.77
es-mx	Mexican Spanish	151.92	30.85
fr-ca	Canadian French	147.22	28.34
fr-fr	European French	181.98	35.06
id	Indonesian	207.62	33.51
my	Malay	157.94	25.51
pt-br	Brazilian Portuguese	202.82	39.53
pt-br	European Portuguese	186.66	36.18

Table 1: The DSL 2016 data set for the task 1. The number of documents in both training (18 000) and development (2 000) sets were balanced. The columns labeled 'characters' and 'tokens' present the average number of non-space characters and white-space-separated tokens for the documents belonging to each language variety in the training set.

Code	Language / variety	Documents	Characters	Tokens
egy	Egyptian	1578	236.63	53.83
glf	Gulf	1672	168.53	38.33
lav	Levantine	1758	163.16	37.67
msa	Modern Standard Arabic	999	231.62	49.04
nor	North-African	1612	140.77	32.01

Table 2: The DSL 2016 data set for the task 2. The column 'documents' number of documents that belong to each language variety in the training set. The columns labeled 'characters' and 'tokens' present the average number of non-space characters and white-space-separated tokens for documents belonging to each language variety.

Test Set	Run	Accuracy	F1 (micro)	F1 (macro)	F1 (weighted)	Rank
A	run1	0.8938	0.8938	0.8938	0.8938	1
A	run2	0.8905	0.8905	0.8904	0.8904	1
B1	run1	0.862	0.862	0.6144	0.8602	5
B1	run2	0.86	0.86	0.6126	0.8576	5
B2	run1	0.822	0.822	0.5839	0.8175	5
B2	run2	0.81	0.81	0.5745	0.8044	5
C	run2	0.4747	0.4747	0.4703	0.4725	9

Table 3: The results submitted to the closed track of DSL shared task 2016. All results are obtained using the same model parameters (an SVM using character ngram features of length one to seven). In *run1*, the model is trained on combined training and development corpus, the model is trained only on the training corpus in *run2*. Since task 2 (test set C) did not have a development set, we only list the results of *run2*.

3.2 Main results

We submitted two sets of results (runs) to the DSL shared task. Both runs were obtained with identical models, the linear SVM described in Section 2.1. The differences between the runs were the data used for the training. For *run1*, we used the all the data available to the closed track participants (training and development sets), while we used only the training set for *run2*. Since test-set C did not have any development set, both runs were identical. We optimized the hyperparameters (the regularization parameter of the SVM model and the range of character n-grams) on the development set of task 1 which is most similar to test set A, and used the same parameters for all sub-tasks of task 1 (test sets A, B1 and B2) and task 2 (test set C). We did not perform any cleanup or filtering on test sets B1 and B2. We only participated in the closed track. The results, as calculated by the shared task organizers, are presented in Table 3.

All results are substantially higher than the trivial (random or majority) baselines, which are 0.08, 0.20 and 0.23 for test sets A, B and C, respectively. The baseline scores are random baselines for test sets A and B, and the majority baseline, which is slightly higher than the random baseline (0.20) for test set C, due to class imbalance.

The differences between *run1* and *run2* are small enough that it does not affect the system's rank in the shared task. However, small but consistent increase in scores of *run1* in comparison to *run2* suggests that more data is useful for this task.

In task 1, our results are better for test set A. Besides the fact that this test set contained in-domain documents, our results are also better here probably because the hyperparameters (regularization constant, and range of character ngrams used) are tuned for this particular task. The performance scores obtained on the gold standard test data is also very close to the scores we have obtained on the development set. We also note that test set A results are, in general, lower than last year's shared task (Zampieri et al., 2015). Despite the fact that there is no 'other' label in this year's data, it seems to be more challenging in some other reasons.

Our results in task 2 are the worse among the results reported in Table 3. The rank we obtained in this task is also relatively low, 9^{th} among 18 participants. However, the scores on this task are rather close to each other, with best accuracy of 0.51. It is also noteworthy that the scores obtained on the gold standard data is substantially lower than the scores we obtained using cross validation on the training data (0.65 accuracy, 0.65 weighted F_1 score). This may be due to differences between the domain (or some other characteristics) of the test set and the gold standard data. Another reason for relatively low score obtained on test set C is due the fact that we truncated the documents to 70 tokens. Since the test set C contains a large number of rather long documents, the truncation choice seems to hurt the performance up to $1\,\%$ in cross validation scores, which may have affected the rank of the system a few steps if the difference was reflected to the testing on the gold standard data.

We also present the confusion tables for each test set in Table 4, Table 5 and Table 6, for test sets A,

B and C, respectively. As expected, most confusion occurs within the language groups. There are very few instances of out-of-group confusions. Among the groups the most confusions occur within *bs-hr-sr* group and the Spanish varieties. This may be due to genuine difficulties of discriminating between these languages, but there may also be some effect of amount of data available for each class. The varieties with least recall often corresponds with the varieties with shorter documents on average, for example, *bs* and *es-mx* which have the smallest number of characters and words within their group (see Table 1).

		Predicted label											
		bs	hr	sr	es-ar	es-es	es-mx	fr-ca	fr-fr	id	my	pt-br	pt-pt
True Label	bs	774	125	98	0	1	0	0	2	0	0	0	0
	hr	138	846	15	0	0	0	0	0	0	0	1	0
	sr	65	12	920	0	0	0	0	2	0	0	1	0
	es-ar	0	0	0	846	43	108	0	2	0	0	1	0
	es-es	0	0	0	43	779	172	0	4	0	0	1	1
	es-mx	0	0	0	90	112	798	0	0	0	0	0	0
	fr-ca	0	0	0	0	0	0	958	42	0	0	0	0
	fr-fr	0	0	0	0	1	0	58	940	0	0	1	0
	id	1	0	0	0	0	0	0	2	977	20	0	0
	my	0	0	0	0	0	0	0	0	14	986	0	0
	pt-br	0	0	0	0	0	0	0	0	0	0	959	41
	pt-pt	0	0	0	0	0	0	0	0	0	0	57	943

Table 4: Confusion table for task 1, test set A. The language labels are explained in Table 1.

The confusion matrices presented in Table 5 also show the same trend. Almost no confusions across the language groups, and the *bs-hr-sr* group also seems to be harder to discriminate here as well. The confusion table between the Arabic dialects, presented in Table 5, shows more confusions overall, as expected from low scores presented in Table 3. Gulf variety seems to be difficult to identify for the system, without a clear pattern. We also observe a relatively poor recall for the North African variety which is mostly confused with, probably not surprisingly, Egyptian Arabic.

		Predicted label											
		Test set B1						Test set B2					
		bs	hr	sr	pt-br	pt-pt	other	bs	hr	sr	pt-br	pt-pt	other
True Label	bs	62	31	5	0	0	2	59	31	8	0	0	2
	hr	2	97	1	0	0	0	0	99	1	0	0	0
	sr	2	1	97	0	0	0	2	1	97	0	0	0
	pt-br	0	0	0	98	2	0	0	0	0	94	6	0
	pt-pt	0	0	0	22	77	1	0	0	0	36	62	2

Table 5: Confusion table for task 1, test sets B1 and B2. The predicted label other refers to all labels that did not occur in the gold standard. This amounts to, *bs–id* and *bs–es-ar* confusion for both tests sets, a single *pt-pt–id* confusion in test set B1, and *bs–id* and *bs–es-ar* confusions in test set B2.

3.3 Results with deep learning architectures

We present the performance scores of FastText and the hierarchical neural network model described in Section 2.2 in Table 7. The models are evaluated on the gold standard test data released after the shared

		Predicted label				
		egy	glf	lav	msa	nor
True Label	egy	171	33	54	27	30
	glf	52	86	44	57	17
	lav	54	65	157	33	35
	msa	27	26	25	179	17
	nor	73	59	45	36	138

Table 6: Confusion table for test set C. The language labels are explained in Table 2.

		Variety accuarcy			**Group accuracy**		
Model	**Features**	A	B1	B2	A	B1	B2
FastText	char	55.69					
	word	76.75					
Hierarchical	char+word	86.42	60.20	70.20	99.66	86.40	92.00

Table 7: The accuracy values obtained on *task 1* by the neural network models. The models are tested on the gold standard testing data.

task results were announced. We evaluate the models only on task 1. The hierarchical model clearly performs better than the FastText baseline, both using character or word features. Although the accuracy on the test set A is not as good as the SVM model discussed above, if we had submitted the results with this model it would have obtained a mid-range rank in the shared task. The group accuracy on test set A is almost perfect. The results on test set B1 and B2 are lower than ones obtained by the SVM model. The performance on B is also noticeably bad for the group prediction. Furthermore, the drop of performance on test sets B in comparison to test set A also seem to be more steep in comparison to the linear models, despite the fact that we prevented overfitting using multiple measures (see Section 2.2). This may be an indication that even though the system may not be overfitting in the usual sense, it may be 'overfitting' to the domain.

4 Discussion and conclusions

In this paper we reported on our contribution to the DSL 2016 shared task. We described and reported results from two (families of) models. The linear SVMs, that we originally intended to use as a baseline, and deep learning methods that we expected to perform well in this task. In our experiments the 'baseline' SVM model outperformed a number of neural network architectures we have experimented with, and it also obtained the first rank in the test set A of on closed track of the shared task. Our neural network models, on the other hand, did not perform as well in this task, although they have some attractive features discussed in Section 1. Within alternative neural network architectures, simple ones seem to perform better, despite some apparent shortcomings. Furthermore the neural network models seem to be more sensitive to domain differences during the training and testing time.

Our findings show that linear models, in general simpler models, are quite useful and hard to beat in this particular setup. Our experiments with the neural network architectures are rather preliminary, and can probably be improved, for example, through better architectures, better hyper-parameters and more training data.

Acknowledgements

The second author has been supported by the ERC Advanced Grant 324246 EVOLAEMP, which is gratefully acknowledged.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Judit Ács, László Grad-Gyenge, and Thiago Bruno Rodrigues de Rezende Oliveira. 2015. A two-level classifier for discriminating similar languages. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (LT4VarDial)*, pages 73–77.

Ahmed Ali, Najim Dehak, Patrick Cardinal, Sameer Khurana, Sree Harsha Yella, James Glass, Peter Bell, and Steve Renals. 2016. Automatic dialect detection in arabic broadcast speech. In *Interspeech 2016*, pages 2934–2938.

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.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of the Third Symposium on Document Analysis and Information Retrieval*, pages 161–175.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

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

Raül Fabra Boluda, Francisco Rangel, and Paolo Rosso. 2015. Nlel upv autoritas participation at discrimination between similar languages (dsl) 2015 shared task. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (LT4VarDial)*, pages 52–58, Hissar, Bulgaria.

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.

Marc Franco-Salvador, Paolo Rosso, and Francisco Rangel. 2015. Distributed representations of words and documents for discriminating similar languages. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (LT4VarDial)*, pages 11–16.

Cyril Goutte and Serge Léger. 2015. Experiments in discriminating similar languages. In *Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, page 78.

Cyril Goutte, Serge Léger, and Marine Carpuat. 2014. The nrc system for discriminating similar languages. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 139–145.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall, second edition.

Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics*, 2:27–40.

Shervin Malmasi, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between similar languages and arabic dialect identification: A report on the third DSL shared task. In *Proceedings of the 3rd Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (VarDial)*, Osaka, Japan.

Dong Nguyen and A. Seza Dogruöz. 2013. Word level language identification in online multilingual communication. In *Conference on Empirical Methods in Natural Language Processing*, pages 857–862. Association for Computational Linguistics.

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

Matthew Purver. 2014. A simple baseline for discriminating similar languages. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*, pages 155–160.

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 (BUCC)*, pages 11–15, Reykjavik, Iceland.

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

Marcos Zampieri, Liling Tan, Nikola Ljubešić, and Jörg Tiedemann. 2014. A report on the dsl shared task 2014. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 58–67.

Marcos Zampieri, Liling Tan, Nikola Ljubešic, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the dsl shared task 2015. In *Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 1–9.