Discriminating between Similar Languages with Word-level Convolutional Neural Networks

Marcelo Criscuolo and **Sandra Maria Aluísio**
Institute of Mathematical and Computer Sciences
University of São Paulo
{mcrisc,sandra}@icmc.usp.br

Abstract

Discriminating between Similar Languages (DSL) is a challenging task addressed at the VarDial Workshop series. We report on our participation in the DSL shared task with a two-stage system. In the first stage, character *n*-grams are used to separate language groups, then specialized classifiers distinguish similar language varieties. We have conducted experiments with three system configurations and submitted one run for each. Our main approach is a word-level convolutional neural network (CNN) that learns task-specific vectors with minimal text preprocessing. We also experiment with multi-layer perceptron (MLP) networks and another hybrid configuration. Our best run achieved an accuracy of 90.76%, ranking 8th among 11 participants and getting very close to the system that ranked first (less than 2 points). Even though the CNN model could not achieve the best results, it still makes a viable approach to discriminating between similar languages.

1 Introduction

Language identification is the task of detecting the language of a given text segment. Although methods that are able to achieve an accuracy of over 99% for clearly distinct languages like English and Spanish do exist (Dunning, 1994), it is still a major problem to distinguish between closely related languages, like Bosnian and Croatian, and language varieties, like Brazilian and European Portuguese (Goutte et al., 2016). The problem of discriminating between similar languages was addressed in the DSL shared task at VarDial 2017. In DSL 2017, participants were asked to develop systems that could distinguish between 14 language varieties, distributed over 6 language groups. Two participation tracks were available: closed and open training. In closed track, systems should be trained exclusively in the DSL Corpus Collection (Tan et al., 2014), provided by the organizers (see Section 3), while in open training the use of external resources was allowed. For a detailed description of the VarDial workshop and of DSL 2017, refer to the shared task report (Zampieri et al., 2017).

This paper describes our system and the results of our submissions for closed track at DSL 2017. Our goal was to experiment with deep neural networks in language variety distinction, in particular word-level Convolutional Neural Networks (CNN). This kind of network has been successfully applied to several natural language processing tasks, such as text classification (Kim, 2014) and question answering (Severyn and Moschitti, 2015; Wang et al., 2016).

Like other participants did in previous editions of the DSL shared task (Zampieri et al., 2015), we chose to use two-stage classification. First, each sentence gets a group label, that guides the selection of a model especially trained for that group. Then, it goes through a classifier that predicts the final language variety. We experimented with different machine learning techniques for variety prediction while the language group classifier was kept the same. This allowed us to compare, not only the overall accuracy of each classifier, but also its accuracy within each language group.

To distinguish between language groups, the efficiency of character *n*-grams was leveraged (Vatanen et al., 2010), while three configurations had their performances comparared for language variety prediction. One run was submitted for each of the following configurations: (a) `run1`: a word-

Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects, pages 124–130,
Valencia, Spain, April 3, 2017. ©2017 Association for Computational Linguistics

level CNN that learns word vectors from scratch; (b) `run2`: a multi-layer perceptron (MLP) fed by tf-idf vectors of word *n*-grams, and (c) `run3`: a hybrid configuration composed by word-level MLP models and character-level Naive Bayes models. Our best run (`run3`) was positioned 8th among 11 participants, with 90.76% of accuracy in the test set and with a difference of 1.98 percentage points from the first system in the rank.

Although our word-level CNN did not outperform the other two configurations, it scored very close to our best run. We also found that combinations of unigrams and bigrams produce higher scores than unigrams alone. This was observed in both convolutional networks and multi-layer perceptron networks.

2 Related Work

Many approaches to discriminating between similar languages have been attempted in previous DSL shared tasks, and best results were achieved by simpler machine learning methods like SVMs and Logistic Regression (Malmasi et al., 2016). However, since deep neural networks have been successfully applied to many NLP tasks such as question answering (Severyn and Moschitti, 2015; Santos et al., 2015; Rao et al., 2016), we wanted to experiment with similar network architectures, particularly CNNs, in the task of discriminating between similar languages.

In the last shared task (DSL 2016), four teams used some form of convolutional neural network. The team *mitsls* (Belinkov and Glass, 2016) developed a character-level CNN, meaning that each sentence character was embedded in vector space. Their system ranked 6th out of seven rank positions, with 0.830 of overall accuracy, while the 1st system scored 0.894 using SVMs and character *n*-grams.

Cianflone and Kosseim (2016) used a character-level convolutional network with a bidirectional long short term memory (BiLSTM) layer. This approach achieved accuracy of 0.785.

A similar approach was used by the team *ResIdent* (Bjerva, 2016). They developed a residual network (a CNN combined with recurrent units) and represented sentences at byte-level, arguing that UTF-8 encodes non-ascii symbols with more than one byte, which potentially allows for more disambiguating power. This system achieved accuracy of 0.849. The fourth team used a word-level CNN (Malmasi et al., 2016), but details are not available since a paper was not submitted.

In DSL 2015, Franco-Salvador et al. (2015) used logistic regression and SVM models fed by pre-trained distributed vectors. Two strategies were explored for sentence representation: sentences represented as an average of its word vectors trained by word2vec (Mikolov et al., 2013), and sentences represented directly as vectors trained by Paragraph Vector (Le and Mikolov, 2014). This system ranked 7th out of 9 participants.

Collobert et al. (2011) propose avoiding task-specific engineering by learning features during model training. In that work, several NLP tasks were used as benchmarks to measure the relevance of the internal representations discovered by the learning procedure. One of these benchmarks used a convolutional layer to produce local features around each word in a sentence.

We intended to experiment with learning word vectors in the target task, in an approach similar to that of Collobert et al. (2011). We are particulary interested in local features captured by convolutional networks. We believe these networks can learn words and language constructions commonly used in particular language varieties.

3 Data

Since we participated in the closed track, all models were trained and tested in the DSL Corpus Collection (Tan et al., 2014), provided by the organizers. This corpus was composed by merging different corpora subsets, for the purpose of the DSL shared task, and comprises news data of various language varieties.

New versions of the DSL Corpus Collection (DSLCC) are build upon lessons learned by the organizers. Thus, an overview of the version used in DSL 2017 is provided in Table 1. It encompasses 14 language varieties distributed over 6 language groups. Since its first release, the DSLCC contains 18,000 training sentences, 2,000 development sentences and 1,000 test sentences for each language variety; each sentence contains at least 20 tokens (Tan et al., 2014).

4 Methodology

Three system configurations were experimented, and one run was submitted for each. We use two-stage classification, and apply different machine

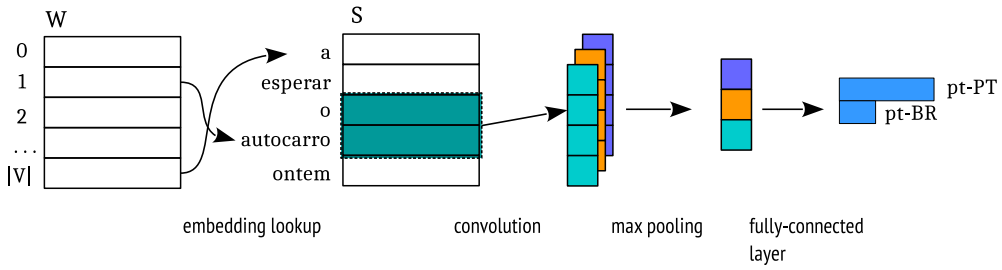

Figure 1: Architecture of the convolutional neural network.

Group	Language/Variety	Code
A	Bosnian	bs
	Croatian	hr
	Serbian	sr
B	Indonesian	id
	Malay	my
C	Persian	fa-IR
	Dari	fa-AF
D	Canadian French	fr-CA
	Hexagonal French	fr-FR
E	Brazilian Portuguese	pt-BR
	European Portuguese	pt-PT
F	Argentine Spanish	es-AR
	Peninsular Spanish	es-ES
	Peruvian Spanish	es-PE

Table 1: Language groups and language varieties contained in DSL Corpus Collection provided for DSL 2017.

learning techniques to train one classifier per language group in each configuration.

Our pipeline starts with language group prediction. After getting a group label, each sentence is forwarded to the corresponding variety classifier. In all configurations, the group classifier was kept fixed.

Character *n*-grams are used to train a Naive Bayes classifier[1] that distinguishes between language groups. Before training, language codes are replaced with the respective group code (*bs*, *hr*, or *sr* becomes *A*, for example), sentences are tokenized, and each token gets an end mark (\$). Tokens are defined as character segments delimited by whitespaces. Better results were achieved in the development set when letter case was kept original, so it was not changed. Named entities were not changed either. We found 5 to be the best size for *n*-grams, with accuracy of 0.9981 in the development set. Values greater than 5 also give good results, but training is much slower.

In the first system configuration, language varieties are classified using convolutional neural networks. This is our main approach.

4.1 Convolutional Neural Network

The model, shown in Figure 1, is similar to one of the architectures experimented by Kim (2014). It takes raw sentences as input and generates class probabilities as output. The highest probability is selected as the predicted class.

Let $s = \{w_1, w_2, w_3, \ldots, w_L\}$ be a sentence of fixed length L. Each word w_j must be mapped to a row vector $x_j \in \mathbb{R}^d$ embedded in matrix $W_{|V|+1\times d}$, where $|V|$ is the number of distinct words in the language group. Rows in W follow the same order as words in the vocabulary, so that the *i*-th row in W represents the vector of the *i*-th word in the vocabulary V. Words are mapped to vectors by looking up their corresponding indexes in W (*embedding lookup*). Words that are not found in the vocabulary V are skipped.

Matrix $S_{L\times d}$ represents the sentence s and is obtained by concatenation of word vectors x_j. Notice that W has $|V| + 1$ rows. The first row corresponds to a special token `PAD`, used to fill up sentences shorter than L.

Convolution filters are slided over S to generate intermediate feature vectors known as *feature maps*. Filters are always of width d, but there may be different filter lengths and multiple filters of each length.

Formally, each feature c_i in a feature map c is computed as

$$c_i = f(w \cdot S_{i:i+h-1} + b) \qquad (1)$$

where $w \in \mathbb{R}^{h\times d}$ is a convolution filter, $b \in \mathbb{R}$ is a bias term, $f(\cdot)$ is a non-linear function such as the hyperbolic tangent, and h is the filter length.

[1] We use scikit-learn multinomial Naive Bayes.

The convolution of 3 filters of length 2 is represented in Figure 1. Each filter generates one feature map.

Max-over-time pooling is applied to each feature map c to take the maximum value $\hat{c} = \max(c)$. Those pooled values are concatenated to form a final feature vector that is fed to a fully-connected layer followed by softmax. For regularization, dropout is applied to the fully-connected layer. The final output is a probability distribution over the class labels.

4.1.1 Model Training

To train the model, sentences are tokenized and all digits (0-9) are replaced with zeros. Letter case is not changed. Tokens are delimited by whitespaces, but no end marker is appended to them. Maximum sentence length L is set to 80, since the longest sentence found in the training set had 77 tokens.

One model is trained for each language group. The vocabulary V is the set of unique tokens found in the training set for the current group. Vocabulary sizes are shown in in Table 2.

Group	Languages	# of tokens
A	bs, hr, sr	175,665
B	id, my	74,654
C	fa-AF, fa-IR	38,145
D	fr-CA, fr-FR	66,891
E	pt-BR, pt-PT	72,694
F	es-AR, es-ES, es-PE	92,062

Table 2: Vocabulary size for each language group.

Word vectors (matrix W) are initialized randomly and updated by backpropagation along with other network weights. Since we intend to minimize the dependence of our model on external resources, that may not be readily available for specific languages, the use of pre-trained word embeddings is entirely avoided.

The model hyperparameters are: vector dimension $d = 200$, filters of lengths (h) 1 and 2 with 100 feature maps each, hyperbolic tangent for non-linearity, drop-rate of 0.20 (or keeping probability of 0.80) for dropout, and shuffled mini-batches of size 50. Parameter values were found by grid search on the development set. All models are trained for 3 epochs, using Adam optimizer (Kingma and Ba, 2014) to minimize the cross-entropy, without early stopping. We use TensorFlow (Abadi et al., 2016) for implementation.

Group	Code	Precision	Recall	F1
A	bs	0.74	0.72	0.73
	hr	0.83	0.84	0.84
	sr	0.85	0.88	0.86
B	id	0.98	0.97	0.97
	my	0.97	0.98	0.98
C	fa-ir	0.95	0.94	0.95
	fa-af	0.94	0.95	0.95
D	fr-ca	0.89	0.91	0.90
	fr-fr	0.90	0.89	0.89
E	pt-br	0.93	0.91	0.92
	pt-pt	0.91	0.93	0.92
F	es-ar	0.85	0.80	0.82
	es-es	0.85	0.84	0.85
	es-pe	0.82	0.88	0.85

Table 3: Performance of run1 (CNN) in each language variety.

4.2 Multi-Layer Perceptron

A vanilla Multi-Layer Perceptron[2] (MLP) was used to compare the CNN performance with that of another neural model.

In this approach, one classifier is trained for each language group, just as before. Sentences are represented as bag of word *n*-grams structured as high-dimensional tf-idf vectors. To make *n*-grams comparable to filters in the CNN models, they are extracted from sentences in sizes of 1 and 2 words (unigrams and bigrams). Letter case is not changed and no transformation is done on digits.

The model has a hidden layer of size 30 and each language variety corresponds to one unit in the output layer. The activation function is hyperbolic tangent. Models are trained for 10 epochs without early stopping by stochastic gradient descent with mini-batches of 200 examples. Optimization is carried out by having Adam optimizer to minimize the cross entropy.

4.3 Hybrid System Configuration

Considering the lower performance of both previous configurations in group A, relatively to other groups, we came up with a hybrid system configuration in which all language varieties are predicted by MLP classifiers, except for group A. For that

[2]We use the MLP classifier implemented in scikit-learn.

group, a standard character *n*-gram model is applied. It is exactly the model described in Section 4 as the first component of our pipeline.

This change caused little impact on performance, as discussed later in Section 6.

5 Results

Table 3 shows the performance of our convolutional neural network (`run1`) in each language variety, while Table 4 shows the corresponding confusion matrix. In Table 4, the horizontal axis indicates predicted labels, while true labels are indicated on the vertical axis. For example, it can be understood that 28 *hr* sentences were wrongly predicted as *sr*. For fine grained results, we opted to report on our main approach (CNN) instead of reporting on our best performing system.

The overall results of our three submitted runs, along with a random baseline, are summarized on Table 5. The result of the best performing system is also reported, and an extra column was appended to the table to report on development set accuracy. Our best run (`run3`) ranked 8th out of 11 participants according to the official evaluation. It achieved an accuracy of 0.9076, with a small difference of 0.0198 percentage points to the best system. Our deep neural network (`run1`) achieved an accuracy of 0.8878, indicating that the CNN scored close to our best run, but could not outperform it. Accuracy values computed on the development set behave similarly to that of the official evaluation.

The result of a traditional single-stage character *n*-gram model is also reported in Table 5 as a baseline for the development set. This is the Naive Bayes model described in Section 4, used to distinguish between language groups, but trained over all 14 language varieties.

6 Discussion

Although we focus on results of our main approach, all three runs behaved similarly. We can see in Table 4 that the confusion between language groups is minimal. This is due to the two-stage architecture that separates sentences in groups before discriminating between varieties.

The group classifier performs its task almost perfectly. In the development set, the group classifier achieved accuracy of 99.81%. We have conducted an error analysis by sampling misclassified sentences, and found that most of them really seems to belong to the predicted language group. In the following example, the classifier predicted group D (French) instead of the true label F (Spanish):

> *Jean-Paul Bondoux, chef propietario de La Bourgogne & Jérôme Mathe, chef de Le Café des Arts (Figueroa*

In most examples, the classifier is misguided by proper nouns in foreign languages, like names of soccer players commonly found in news texts.

Prior classification of language groups narrows down the set of output classes for variety classifiers, allowing for their optimization in a single language. We believe this raises the accuracy within language groups.

However, some language groups are more challenging than others, as is shown in Table 3. Groups A and F are responsible for the lowest scores. Group A, particulary, contains the most difficult language to discriminate (*bs*) for our three system configurations. Even the change from a neural to a statistical approach in our hybrid configuration had little impact in that group performance (Table 5). This was observed both in the development set and the official runs.

The vocabulary of group A may lead to more sparse language models that hinders performance of classifiers. Group A contains almost 2 times the number of tokens in group F, the second largest group which also comprises 3 language varieties (Table 2).

Overall, our hybrid configuration showed the best performance, which is very close to the MLP. In fact, we would still rank the same position if the MLP configuration (`run2`) were considered instead.

Although the MLP scored higher than the CNN, difference was small. Also, the convolutional model is trained relatively fast in appropriate hardware, considering that pre-trained word vectors are not used and all model values are initialized randomly. With its minimum preprocessing requirements, these characteristics make our word-level CNN a viable model for discriminating between similar languages.

7 Conclusion

In this work we explored word-level convolutional neural networks to discriminate between similar languages and language varieties. Our intuition

	hr	bs	sr	es-ar	es-es	es-pe	fa-af	fa-ir	fr-ca	fr-fr	id	my	pt-br	pt-pt
hr	837	131	28	0	1	0	0	0	0	1	2	0	0	0
bs	156	718	125	0	0	0	0	0	0	1	0	0	0	0
sr	10	114	876	0	0	0	0	0	0	0	0	0	0	0
es-ar	0	0	0	798	77	123	0	0	0	0	0	0	2	0
es-es	0	0	0	90	842	63	0	1	1	0	0	0	0	3
es-pe	0	0	0	55	67	878	0	0	0	0	0	0	0	0
fa-af	0	0	0	0	0	0	953	47	0	0	0	0	0	0
fa-ir	0	0	0	0	0	0	59	940	0	1	0	0	0	0
fr-ca	0	0	0	0	0	0	0	0	909	91	0	0	0	0
fr-fr	0	1	2	0	1	0	0	0	110	885	0	0	0	1
id	0	0	0	0	1	0	0	0	0	1	971	27	0	0
my	0	0	0	0	0	0	0	0	0	1	19	980	0	0
pt-br	0	0	0	0	0	2	0	0	0	2	0	0	913	83
pt-pt	0	0	0	0	1	1	0	0	0	1	0	0	68	929

Table 4: Confusion matrix for the DSL task, run1 (CNN). The horizontal axis indicates predicted labels, while true labels are on the vertical axis.

Run	**Config.**	**Accuracy**	**F1 (micro)**	**F1 (macro)**	**F1 (weighted)**	**Dev Accuracy**
Random baseline		0.0710				
Best system		0.9274				
run1	CNN	0.8878	0.8878	0.8876	0.8876	0.8954
run2	MLP	0.9033	0.9033	0.9029	0.9029	0.9107
run3	Hybrid	0.9076	0.9076	0.9075	0.9075	0.9120
NB baseline						0.8976

Table 5: Results for the DSL task. Last column shows results computed on the development set.

is that language varieties can be distinguished by particular words and common language constructions. Even though we argue for avoiding task-specific feature engineering, we believe this kind of linguistic bias is fundamental to the success of methods that address the task of discriminating between similar languages. We believe both the CNN and the MLP models were able to capture particular words and common language constructions as features.

Acknowledgements

The work of Marcelo Criscuolo was fully funded by Federal Institute of Education, Science and Technology of São Paulo (IFSP).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Yonatan Belinkov and James Glass. 2016. A Character-level Convolutional Neural Network for Distinguishing Similar Languages and Dialects. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 145–152, Osaka, Japan.

Johannes Bjerva. 2016. Byte-based Language Identification with Deep Convolutional Networks. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 119–125, Osaka, Japan.

Andre Cianflone and Leila Kosseim. 2016. N-gram and Neural Language Models for Discriminating Similar Languages. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 243–250, Osaka, Japan.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Ted Dunning. 1994. *Statistical Identification of Language*. Computing Research Laboratory, New Mexico State University.

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, Hissar, Bulgaria.

Cyril Goutte, Serge Léger, Shervin Malmasi, and Marcos Zampieri. 2016. Discriminating Similar Languages: Evaluations and Explorations.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Shervin Malmasi and Marcos Zampieri. 2016. Arabic Dialect Identification in Speech Transcripts. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 106–113, Osaka, Japan.

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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Cícero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 694–699.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

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.

Tommi Vatanen, Jaakko J. Väyrynen, and Sami Virpioja. 2010. Language identification of short text segments with n-gram models. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *CoRR*, abs/1602.07019.

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)*, pages 1–9, Hissar, Bulgaria.

Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. 2017. Findings of the VarDial Evaluation Campaign 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain.