TwistBytes - Identification of Cuneiform Languages and German Dialects at VarDial 2019

Fernando Benites
benf@zhaw.ch
Zurich University of Applied Sciences, Switzerland

Pius von Däniken
vode@zhaw.ch
Zurich University of Applied Sciences, Switzerland

Mark Cieliebak
mc@spinningbytes.com
SpinningBytes AG, Switzerland

Abstract

We describe our approaches for the German Dialect Identification (GDI) and the Cuneiform Language Identification (CLI) tasks at the VarDial Evaluation Campaign 2019. The goal was to identify dialects of Swiss German in GDI and Sumerian and Akkadian in CLI.

In GDI, the system should distinguish four dialects from the German-speaking part of Switzerland. Our system for GDI achieved third place out of 6 teams, with a macro averaged F-1 of 74.6%. In CLI, the system should distinguish seven languages written in cuneiform script. Our system achieved third place out of 8 teams, with a macro averaged F-1 of 74.7%.

1 Introduction

The 6th Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2019) included an evaluation campaigns with five shared tasks with the goal to find approaches which can differentiate dialects in various languages. We describe our solutions for two sub-tasks: German Dialect Identification (GDI) and Cuneiform Language Identification (CLI).

GDI The GDI task (Zampieri et al., 2019) had the goal to classify sentences into four dialects of Swiss German. Each sentence was transcribed and annotated with the dialect area of the speaker (Bern, Basel, Lucerne, and Zurich). No additional information other than the task data should be used (closed submission). The task was a continuation of similar shared tasks in previous years (Zampieri et al., 2018).

In the German speaking part of Switzerland, there exist many dialects which are quite different, and speakers of one dialect might even have difficulty understanding dialects of regions not far away. There is no standardized writing for Swiss German.

The identification of dialect based on text is a challenging task, especially if there is no standardized written form of the dialects. First of all, transcribing audio signals to text is highly ambiguous and can be very subjective. Even with detailled transcription guidelines, the resulting text can differ significantly among annotators. This subjectivity of the annotations manifests in similar tasks such as in labelling multi-label samples (Benites, 2017). Another problem is that for short sentences there is little text which could point to a dialect. A good example can be found in the GDI dataset, which contains samples“jaja jaja” and “jaja ja ja”. Both roughly translate to “yes yes“ or “indeed“, and both mean the same for Swiss German speakers. The first is labelled with Lucerne dialect, while the second is labeled with Zurich dialect. Therefore these samples are easily misclassified. To cope with this issue, i-Vectors were provided by the organizers this year (see Section 3.2.4).

CLI The CLI task (Jauhiainen et al., 2019) is new to VarDial and consists of classifying texts written in cuneiform script into Sumerian or one of six dialects of Akkadian: Old Babylonian, Middle Babylonian Peripheral, Standard Babylonian, Neo-Babylonian, Late Babylonian, and Neo-Assyrian. Cuneiform is one of the oldest known writing systems and has been used for 3,000 years by different cultures around Mesopotamia.

Our Solutions We describe in this paper the approaches taken by our team TwistBytes. The GDI system is an updated version of our solution for the previous year’s competition (Benites et al., 2018). We improved several parameters and extended it to use semi-supervised learning and i-Vectors. For the sake of completeness, we recapitulate the base system description in Section 3.2,

and then describe the applied modifications. This system achieved 3rd place among 6 participants at GDI, with a macro averaged F-1 of 74.6%.

For CLI, we use a linear SVM as classifier with character n-gram features and include perplexities from character n-gram language models as additional features. This system achieved 3rd place among 8 participants at CLI, with a macro averaged F-1 of 74.7%.

2 Related Work

The central focus of the evaluation campaign at VarDial is to properly identify dialect of various languages. For GDI, there have been two previous editions of the shared task, which laid the basis for dialect identification in Swiss German (Zampieri et al., 2017, 2018). Solving this problem can have a positive impact on many tasks, e.g. for POS-tagging of dialectal data (Hollenstein and Aepli, 2014), for compilation of German dialect corpora (Hollenstein and Aepli, 2015), or for automatic speech recognition of Swiss German.

Many studies tackled the problem of language and dialect identification for other languages, creating a noticeable amount of related work, described in short in the evaluation campaign reports and (Jauhiainen et al., 2018b). A typical approach uses SVMs with different feature extraction methods. The use of character language models for language identification has previously been studied by (Vatanen et al., 2010).

Our approach is most similar to MAZA, which was proposed at VarDial 2017 (Malmasi and Zampieri, 2017b). MAZA uses Term Frequency (TF) on character-n-grams and unigrams for word features to train several SVMs. Then it uses a Random Forest meta-classifier with 10-fold cross-validation on the predictions of the SVMs. We extended this approach and used Term Frequency-Inverse Document Frequency (TF-IDF) on word and on character level. We used an SVM as meta-classifier, and we did not concatenate the output of the base classifiers but summed them. Similar to (Malmasi and Zampieri, 2017a) we used a single SVM classifier for the i-Vectors. More details are given in Section 3.2.

3 Data and Methodology

3.1 Task Definition

The task of GDI is to classify a transcribed sentence from the ArchiMob data set (Samardžić et al., 2016) into one of four classes of Swiss German dialect. Each class represents a dialect area of Swiss German: Bern (BE), Basel (BS), Lucerne (LU) and Zurich (ZH). Since the dialects are very different from Standard German, the sentences are transcribed using the guideline book by Dieth (Dieth and Schmid-Cadalbert, 1986). It is a phonetics oriented transcription method but it is orthographic and is partially adapted to standard German spelling habits and alphabet. As such it loses some of the precision and explicitness of phonetic transcription methods such as the International Phonetic Alphabet. We expected therefore that character-based and error tolerant methods will perform best, since different spellings of the same word might occur.

Table 1 shows the number of sentences in the training set, the validation set, and test set per dialect area. The training set was slightly changed in comparison to the one from last year. The validation set was the test set of 2018 GDI shared task. The sentence distribution is almost evenly balanced over all four dialect areas.

One peculiarity that occured in the data of last year was the ambiguity of labels for identical sentences, i.e. that multiple sentences with identical transcriptions had different labels. This was not the case when looking at the training and validation set in isolation[1]. Only when both were merged, the sentence "i däre ziit" was labelled as originating from Zurich in the training set and from Basel in the validation set. We expected that i-Vectors would help solving these kinds of ambiguities.

GDI dataset information					
Set	BE	BS	LU	ZH	Total
Train	3750	3268	3390	3870	14278
Validation	1053	1528	1016	932	4529
Test					4742

Table 1: Number of instances per dialect area for GDI dataset

For the Cuneiform Language Identification task, CLI (Jauhiainen et al., 2019), we have to distinguish Sumerian (SUX) and six variants of the Akkadian language: Old Babylonian (OLB), Middle Babylonian Peripheral (MPB), Standard Babylonian (STB), Neo-Babylonian (NEB), Late Baby-

[1]However, some sentences were on a character level very close, as pointed out in the Introduction.

CLI dataset information			
	Train	Validation	Test
LTB	15947	668	
MPB	5508	668	
NEA	32966	668	
NEB	9707	668	
OLB	3803	668	
STB	17817	668	
SUX	53673	668	
Total	139421	4676	6895

Table 2: Number of instances for CLI dataset

lonian (LTB), and Neo-Assyrian (NEA). Table 2 shows the distribution of labels per language in the dataset. The texts are all written in cuneiform script and provided as their Unicode representation. Since the different language variants are spread over multiple centuries, we assumed that the symbols in use differ across them. Table 3 shows the Jaccard Similarity (Jaccard, 1902) between the sets of unique symbols used by every language computed on the training set. Sumerian is the most dissimilar in its use of symbols compared to the other languages. This was to be expected, as it is the only language not in the Akkadian language family. We expect that dialects with lower similarity in symbol use can be easier distinguished, which was our motivation to use language modeling as part of the classification system.

3.2 System Definition TB-Meta for GDI

In this section, we describe our approach for GDI in detail. One part (meta crossvalidation) is based on the system from (Malmasi and Zampieri, 2017b) but extended in several ways. A previous version was already described in (Benites et al., 2018). The key improvements this year are the optimization of the preprocessing and and the feature extraction, a new preprocessing step between base classifier and meta-classifier, and using an SVM as meta-classifier. This year we extended our approach for GDI to also use a semi-supervised method.

We observed that much of the recognition can be performed on character level, where character bigrams can provide a key insight, while demonstrating a high efficiency. The four processing steps of the system are: a) to preprocess the sentences, b) extract features from them, c) classify with a base classifier and d) pass the predictions to a meta-classifier which, in turn, provides the final prediction. This year, we were additionally provided with i-Vectors. These represent important new features, especially given the fact that there are only few speakers per label set. That means that if we can identify the speaker and the spoken dialect at least once with high confidence, we can easily label the other sentences spoken by them. Here especially, we expect that the semi-supervised approach improves the overall performance.

3.2.1 Preprocessing

The basic preprocessing step was to split the sentences in words by using white-spaces and convert them to lower case. No stopword removal or lemmatization was performed since these steps might erase any traces of key features for differentiating between the dialects (see (Maharjan et al., 2014)). Afterwards multiple feature extraction methods were applied, as explained in the next section.

3.2.2 Feature Extraction

In this edition of the VarDial GDI task, we were provided with i-Vectors in addition to textual features. This leads to the following observation: the textual feature vectors are very sparse, as the average word occurrence is 7.4 ± 4.16 per sentence in the training set, whereas the i-Vectors are dense. This has the effect that the SVMs need to cope with multi-modal features with different density and that was the main reason why we used a separate SVM for the i-Vectors.

We use Term Frequency (TF) with n-grams for characters and words for n ranging from 1 to 7. An additional preprocessing for the classifiers employed (see Section 3.2.3) is to normalize the TF values, at least per sentence, which in some cases can improve prediction quality. Also, we calculated the TF-IDF (Manning et al., 2008), which usually gives the best single feature set for prediction quality.

For the feature extraction, we mainly used the scikit-learn[2] package with one modification: We also used a custom character bigram analyzer (referred to later as CB) in order to produce character bigrams without spaces, since the standard implementation considers all characters in the text including the spaces, especially at the beginning and

[2]http://scikit-learn.org

	LTB	MPB	NEA	NEB	OLB	STB	SUX
LTB	1.00	0.71	0.80	0.77	0.72	0.71	0.53
MPB	0.71	1.00	0.65	0.68	0.71	0.57	0.43
NEA	0.80	0.65	1.00	0.85	0.70	0.82	0.61
NEB	0.77	0.68	0.85	1.00	0.71	0.81	0.60
OLB	0.72	0.71	0.70	0.71	1.00	0.62	0.46
STB	0.71	0.57	0.82	0.81	0.62	1.00	0.70
SUX	0.53	0.43	0.61	0.60	0.46	0.70	1.00

Table 3: Jaccard similarities between the sets of symbols used per language in the training set of CLI

end of a word[3]. We employed TF-IDF not only on word level but also on character level.

Each of the feature extraction methods from the texts served as a separate feature set which was processed by a base classifier. The entire list is: TF word n-grams (TF-W), TF character n-grams (TF-C), TF-IDF words (TF-IDF-W), custom bigrams analyzer (CB-C), TF-C normalized to range from 0 to 1 (TF-C-N) and TF-IDF character n-grams (TF-IDF-C).

Additionally, i-Vectors were used to extract characteristics about the speaker from the audio signal. These were provided by the task organizers. They are basically a simplified variant of the joint factor analysis (Chen et al., 2014) which assured the anonymity of the speakers. As in (Suh et al., 2011), we also normalized the i-Vectors to unit length for better performance based on experiments on the validation set. We used them as input features to a separate base classifier (along with the extracted text features), that is each textual feature had their separate classifier and likewise the i-Vectors, only the output of the classifiers were merged.

3.2.3 Classifiers

Last year edition of GDI (Zampieri et al., 2018) showed that concatenating textual features for SVM produced worse prediction quality than the use of ensemble classifiers. We use the ensemble S-Classifier, which sums the predictions of the base classifiers and gives as input to a linear SVM (see Figure 1), and achieved considerably good results in GDI.

Meta Crossvalidation Classifier We used a two-tier meta-classifier bottom-up with crossvalidation (referred to as TB-Meta) to eliminate the need for parameter/weighting search. The workflow of the system is depicted in Figure 1. First an input sentence is preprocessed, then the features are extracted and passed to the base classifiers, one classifier per feature set. The predictions of the classifiers are summed (S-Classifier output), and these intermediate predictions are passed to a last classifier (meta-classifier) that decides about the final label. The second level of the procedure also ensures that the class interdiscrimination is improved. Further, each base classifier prediction is then weighted by the meta-classifier. That means a weighting scheme and time-consuming parameter search is not needed anymore. For a detailed description please see (Benites et al., 2018).

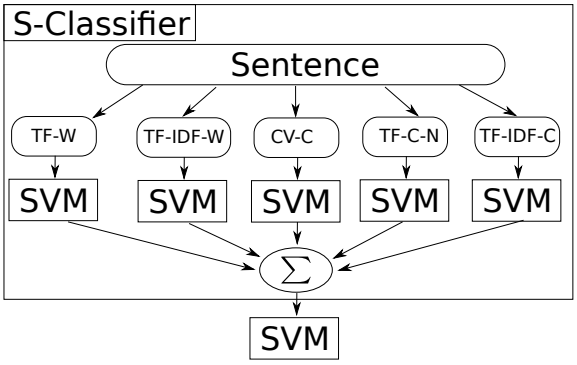

Figure 1: TB-Meta classifier workflow, with S-classifier and textual features

Semi-Supervised Learning After the training, we augmented TB-Meta with a semi-supervised learning similar as in (Jauhiainen et al., 2018a). This approach consists of classifying the unlabelled test set with a model based on the training data, then selecting the predictions with the highest confidence and using them as new additional (weak) labelled training samples. The method can be very useful if there are few training samples and a test set with out-of-domain data is expected. Our approached consisted of setting a threshold for the confidence output of the SVM (fitted with regres-

[3]This can also be probably implemented with sklearn setting the analyzer to "char_wb" but we did not evaluate the differences in implementations.

sion) to 0.9 and in each iteration decreasing it by $\frac{i}{20}$ where i is the number of the iteration.

3.2.4 I-Vectors

There is strong indication that the i-Vectors in the context of dialect identification provide strong features for Arabic dialect classification, as described in (Bahari et al., 2014; Malmasi and Zampieri, 2017a). We used them in the context of GDI with caution, especially, because there are only few speakers in each set. On the one hand, if a speaker was identified and correctly classified, the problem would be solved. However, if there was something off with the i-Vectors of one speaker, it could throw the classifier off the right track. We integrate the i-Vectors into last year's system by training a separate base SVM classifier with the i-Vectors and the output is used as additional input for the S-classifier.

3.3 System Definition TB-LM for CLI

The TB-LM system consists of TF-IDF features, language modelling features, and a linear SVM classifier. We do not apply any preprocessing steps to the texts but work directly with the Unicode codepoints.

TF-IDF We use TF-IDF features on the symbol level for n-gram lengths from 1 to 3. Most samples in the training set are short: On average they are 7.05 symbols long, with a median length of 5 symbols. Therefore, we use only binary term frequency counts, as those tend to be more robust for classification of short texts.

Language Modelling Additionally, we train a 3-gram language model with Kneser-Ney smoothing (Kneser and Ney, 1995) for every language. We use the language model scores as additional features for every sample.

We again use scikit-learn for the TF-IDF features and the SVM, as well as nltk (*Natural Language Toolkit*[4]) for language modeling.

During experimentation, we train the system on the training set and evaluate on the validation set. For the final submission, we train on training and validation set jointly.

System	macro F-1
GDI validation set	
Random(10)	0.2468±0.0079
SVM Char-Ngram(1,7)	0.6494
TB-Meta	0.6984
TB-Meta$-iV$	0.6769
TB-Meta$-S$	0.7516
TB-Meta$-SiV$	0.9028
TB-Meta$-SiV_p$	**0.9031**
GDI test set	
TB-Meta$-iV$	0.6823
TB-Meta$-SiV$	**0.7455**
TB-Meta$-SiV_p$	0.7349

Table 4: Results for the GDI task on the validation and test set for TB approaches.

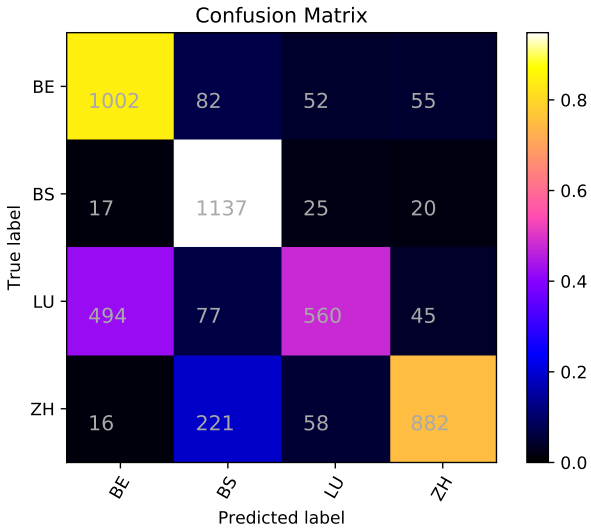

Figure 2: Confusion matrix of TB-Meta-SiV GDI

4 Results

4.1 GDI

Parameters For this year's shared task, we used a maximum of 100'000 features per subclassifier/feature set for the TB-Meta approach, based on our experimental results from last year. For the semi-supervised method we used 10 iterations.

Results Discussion We present in Table 4 the different approaches used in the GDI task and how they performed on the validation set. We also put the submitted results for comparison, which allows to see how the submitted approaches performed on the test set. TB-Meta refers to the TwistBytes Meta classifier, the $-iV$ suffix refers

[4]https://www.nltk.org/

Team	macro F-1	Place
tearsofjoy	0.7593	1
SUKI	0.7541	2
TwistBytes	0.7455	3
BAM	0.6255	4
dkosmajac	0.5616	5
ghpaetzold	0.5575	6

Table 5: Competition result of GDI with TwistBytes approach TB-Meta-SiV

System	F-1 (macro)	Accuracy
TB-Meta	0.6669	0.6751
TB-LM	**0.7433**	**0.7469**

Table 6: Performance of the runs submitted for the CLI task on the test set.

to the use of i-Vectors, the suffix $-S$ to the semi-supervised version, and $-SiV$ to i-Vectors with semi-supervised. The $_p$ index points to a different parameter set with a maximum of 90'000 features and C=1.5 for the SVM.

System TB-Meta-SiV shows a surprisingly good score of 0.90 on the validation set. As stated before, one reason could be the low numbers of speakers in the validation set. Also with a baseline composed by TF-IDF and SVM (SVM Char-Ngram(1,7), see Table 4), we achieved better results as from our approach in VarDial 2018 GDI, which was 64.6% macro F-1. This points to the fact that the data was curated, and therefore easier to classify. The assumption that the test set was similarly built like the validation set guided our approach. However, the results showed clearly that there was some difference, since the scores on the test data were significantly lower for the TB-Meta$-SiV$ systems. We intend to investigate this observation in a future study.

Figure 2 shows the confusion matrix of our system on the test data. It shows that sentences from Lucerne were often predicted as from Bern, and some of Zurich were predicted as Basel. Apart from that, error rates were mostly below 10%.

In Table 5 the results of the shared task are shown. The best three results achieved macro F-1 scores between 74.55% and 75.93%. This pushes forward by a considerable margin the results of last year. One reason might be the i-Vectors features, which were available for the first time this year. Our system achieved 74.55% macro F-1.

Team	macro F-1	Place
NRC-CNRC	0.7695	1
tearsofjoy	0.7632	2
TwistBytes	0.7433	3
PMZ	0.7387	4
ghmerti	0.7210	5
ghpaetzold	0.5562	6
ekh	0.5501	7
situx	0.1276	8

Table 7: Competition result for CLI with TwistBytes approach TB-LM

	Precision	Recall	F-1
LTB	0.93	0.95	0.94
MPB	0.87	0.84	0.85
NEA	0.60	0.84	0.70
NEB	0.73	0.49	0.58
OLB	0.89	0.43	0.58
STB	0.67	0.71	0.69
SUX	0.66	0.93	0.77
micro avg	0.74	0.74	0.74
macro avg	0.76	0.74	0.73

Table 8: Evaluation of TB-LM on the validation set of CLI

4.2 CLI

Table 6 shows the performance of our submitted systems on the test set of the CLI shared task. TB-Meta is the same architecture as used for GDI and described in Section 3.2.3. The TB-LM system is described in Section 3.3. Table 7 shows how our system performs compared to the other participants. We achieve third place out of 8 participants with a macro F-1 score of 0.74. Table 8 shows the performance of TB-LM on the validation set in more detail. The performance on the validation set is 0.73, which is slightly lower than on the test set.

Figure 3 shows the confusion matrix of our system on the test set, and Figure 4 shows the confusion matrix on the validation set. They are mostly similar. Overall, LTB was the easiest language to identify, with an F-1 score of 0.94. NEB and OLB were the hardest to identify, OLB having overall the lowest number of samples in the training set. Noteworthy is that the number of samples alone is not an indicator of how well a class can be distinguished. For example, MPB has the second lowest number of samples (5508) but the second high-

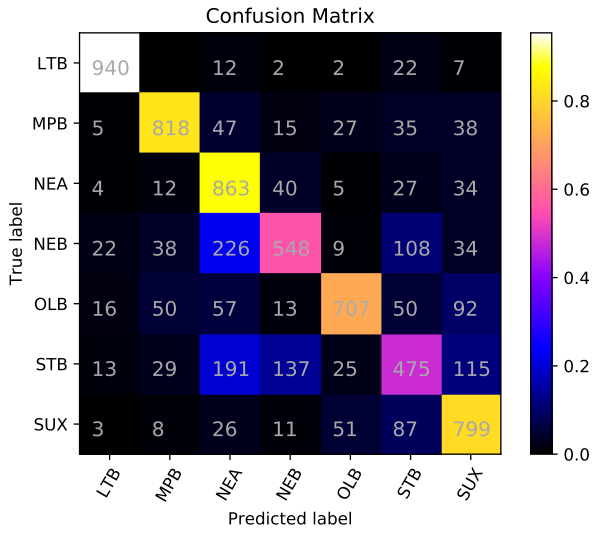

Figure 3: Confusion matrix of TB-LM for CLI on the test set

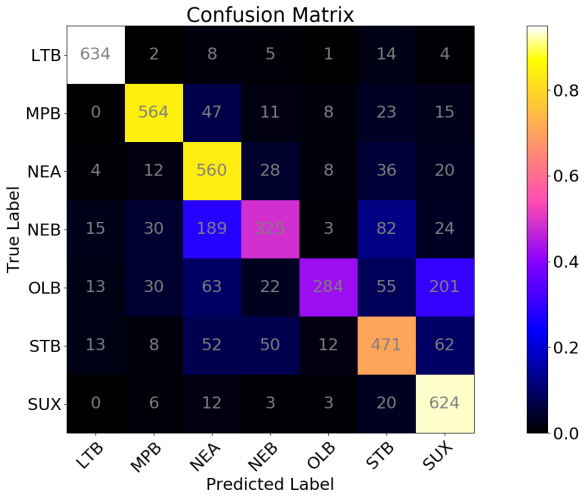

Figure 4: Confusion matrix of TB-LM for CLI on the validation set

est F-1 score (0.85), whereas SUX has by far the largest number of samples (53673), but only the third highest F-1 (0.77).

5 Conclusion

We described our dialect identification systems that were submitted to the VarDial shared tasks GDI and CLI. In GDI, we achieved 3rd place out of 6, using a linear SVM as base, semi-supervised meta crossvalidation training, multiple word and character features, and i-Vectors. In CLI, we achieved 3rd place among 8 teams, using a linear SVM with character n-gram and language model perplexity features.

Acknowledgement

We thank the task organizers for their support and the reviewers for their detailed and helpful feedback. This research has been funded by Commission for Technology and Innovation (CTI) project no. 28190.1 PFES-ES and by SpinningBytes AG, Switzerland.

References

Mohamad Hasan Bahari, Najim Dehak, Hugo Van Hamme, Lukas Burget, Ahmed M. Ali, and Jim Glass. 2014. Non-negative Factor Analysis of Gaussian Mixture Model Weight Adaptation for Language and Dialect Recognition. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 22(7):1117–1129.

Fernando Benites. 2017. *Multi-label Classification with Multiple Class Ontologies*. Ph.D. thesis, University of Konstanz, Konstanz.

Fernando Benites, Ralf Grubenmann, Pius von Däniken, Dirk von Grünigen, Jan Deriu, and Mark Cieliebak. 2018. Twist Bytes-German Dialect Identification with Data Mining Optimization. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 218–227.

Kuan-Yu Chen, Hung-Shin Lee, Hsin-Min Wang, Berlin Chen, and Hsin-Hsi Chen. 2014. I-vector based language modeling for spoken document retrieval. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7083–7088. IEEE.

E. Dieth and C. Schmid-Cadalbert. 1986. *Schwyzertütschi Dialäktschrift: Dieth-Schreibung*. Lebendige Mundart. Sauerländer.

Nora Hollenstein and Noëmi Aepli. 2014. Compilation of a Swiss German dialect corpus and its application to PoS tagging. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 85–94.

Nora Hollenstein and Noëmi Aepli. 2015. A Resource for Natural Language Processing of Swiss German Dialects. In *GSCL*.

Paul Jaccard. 1902. Distribution comparée de la flore alpine dans quelques régions des Alpes occidentales et orientales. *Bulletin de la Murithienne*, (31):81–92.

Tommi Jauhiainen, Heidi Jauhiainen, Tero Alstola, and Krister Lindén. 2019. Language and Dialect Identification of Cuneiform Texts.

Tommi Jauhiainen, Heidi Jauhiainen, and Krister Lindén. 2018a. HeLI-based experiments in Swiss

German dialect identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 254–262.

Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2018b. Automatic Language Identification in Texts: A Survey. *arXiv preprint arXiv:1804.08186*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *ICASSP*, pages 181–184. IEEE Computer Society.

Suraj Maharjan, Prasha Shrestha, and Thamar Solorio. 2014. A Simple Approach to Author Profiling in MapReduce. In *CLEF*.

Shervin Malmasi and Marcos Zampieri. 2017a. Arabic Dialect Identification Using iVectors and ASR Transcripts. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 178–183, Valencia, Spain.

Shervin Malmasi and Marcos Zampieri. 2017b. German Dialect Identification in Interview Transcriptions. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 164–169, Valencia, Spain.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Tanja Samardžić, Yves Scherrer, and Elvira Glaser. 2016. ArchiMob–A corpus of spoken Swiss German. In *Proceedings of the Language Resources and Evaluation (LREC)*, pages 4061–4066, Portoroz, Slovenia).

Jun-Won Suh, Seyed Omid Sadjadi, Gang Liu, Taufiq Hasan, Keith W Godin, and John HL Hansen. 2011. Exploring Hilbert envelope based acoustic features in i-vector speaker verification using HT-PLDA. In *Proc. of NIST 2011 Speaker Recognition Evaluation Workshop*.

Tommi Vatanen, Jaakko J. Väyrynen, and Sami Virpioja. 2010. Language Identification of Short Text Segments with N-gram Models. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

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.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Ahmed Ali, Suwon Shon, James Glass, Yves Scherrer, Tanja Samardžić, Nikola Ljubešić, Jörg Tiedemann, Chris van der Lee, Stefan Grondelaers, Nelleke Oostdijk, Antal van den Bosch, Ritesh Kumar, Bornini Lahiri, and Mayank Jain. 2018. Language Identification and Morphosyntactic Tagging: The Second VarDial Evaluation Campaign. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, USA.

Marcos Zampieri, Shervin Malmasi, Yves Scherrer, Tanja Samardžić, Francis Tyers, Miikka Silfverberg, Natalia Klyueva, Tung-Le Pan, Chu-Ren Huang, Radu Tudor Ionescu, Andrei Butnaru, and Tommi Jauhiainen. 2019. A Report on the Third VarDial Evaluation Campaign. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*. Association for Computational Linguistics.