Identifying dialects with textual and acoustic cues

Abualsoud Hanani, Aziz Qaroush
Electrical & Computer Engineering Dept
Birzeit University
West Bank, Palestine
{ahanani,qaroush}@birzeit.edu

Stephen Taylor
Computer Science Dept
Fitchburg State University
Fitchburg, MA, USA
staylor@fitchburgstate.edu

Abstract

We describe several systems for identifying short samples of Arabic or Swiss-German dialects, which were prepared for the shared task of the 2017 DSL Workshop (Zampieri et al., 2017). The Arabic data comprises both text and acoustic files, and our best run combined both. The Swiss-German data is text-only. Coincidently, our best runs achieved a accuracy of nearly 63% on both the Swiss-German and Arabic dialects tasks.

1 Introduction

The 2017 Distinguishing Similar Languages Workshop sponsored four shared tasks, and our team participated in two of them: Arabic dialect identification, and Swiss-German dialect identification. The Arabic dialect data includes Automatic Speech Recognition transcripts of broadcasts, as well as the most helpful audio features, which were provided as 400-dimensional I-vector files. The raw audio data was also available for download. The Swiss-German data consists of transcripts only, transcribed to indicate pronunciation by human linguists.

The training set for Arabic comprises 14000 lines, totaling 1.7MB, each line labeled for one of five dialect groups. In addition, 1524 lines totaling 318KB of development data were also provided. The test set is 1492 lines.

We did not use the IS2016 data or the varDial3 shared task data, which have similar characteristics, and might have improved the efficacy of training.

For the three Arabic runs, we prepared six different text-based classifiers, and five wave-file-based classifiers, in addition to the two baseline word and I-vector systems, and combined them in two groups of four and one group of five classifiers.

Our best run on the Arabic test data has a weighted F1 score of 0.628; this run combined some of our classifiers with the provided `svm_multiclass` baseline classifiers.

The Swiss-German data consists of 14478 lines of data, totalling 700KB, labeled with one of four dialects. We divided this into a 13032 line training set, and two 723-line files for development. The test set is 3638 lines.

Only two of the classifiers prepared for Arabic were deployed on the Swiss-German test data. Our best run on this data has an accuracy of 0.63 and a weighted F1 score of 0.61.

2 Related Work

In Ferguson (1959), which introduced the term *diglossia* into English, two of his four principal examples are Arabic and Swiss-German. In these languages, every educated native speaker has two distinct languages, the mother tongue, and the language of education.

In both instances, the languages have a prestigous written form with a unified literary tradition, in which native speakers of all dialects are educated. In some registers, the spoken language of various regions is mutually unintelligible. At more formal registers, the distinctions between dialects include vocabulary shifts and phonemic variations, but vocabulary is more similar to the written language and communication is less difficult. For example, speakers using the more formal registers of Arabic dialects often claim to be speaking classical Arabic, albeit with an 'accent' – an accent which drops classical case markings, changes the vowels, and reassigns many phonemes.

Among other applications for dialect recogni-

Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects, pages 93–101,
Valencia, Spain, April 3, 2017.

tion, it might serve as a selector for acoustic and language models for ASR, as shown in Najafian et al. (2014), which achieved a 44% improvement in word error rate after 43 seconds of accent identification for British dialects.

Biadsy et al. (2009) distinguish four Arabic dialects and MSA[1] based on (audio) phone sequences; the phones were obtained by phone recognizers for English, German, Japanese, Hindi, Mandarin, Spanish, and three different MSA phone-recognizer implementations. The dialects were distinguished by phoneme sequences, and the results of classifications based on each phone-recognizer were combined using a logistic regression classifier. They train on 150 hours per dialect of telephone recordings. They report 61% accuracy on 5-second segments, and 84% accuracy on 120 second segments.

Zaidan and Callison-Burch (2011) describe building a text corpus, based on reader commentary on newspaper websites, with significant dialect content; the goal is to provide a corpus to improve machine translation for Arabic dialects. They used Amazon Mechanical Turk to provide annotation for a portion of the corpus. Zaidan and Callison-Burch (2014) describe the same work in greater detail, including dialect classifiers they built using the Mechanical Turk data for classes and origin metadata as additional features. They say these classifiers are 'approaching human quality.'

ElFardy and Diab (2013) classify EGY[2] and MSA sentences from the Zaidan and Callison-Burch (2011) corpus, that is, from text. Not only is this a binary task, but orthographic hints, including repeated long vowels, emojis and multiple punctuation, give strong clues of the register, and hence whether MSA is being employed. They do a number of experiments comparing various preprocessing schemes and different training sizes, ranging from 2-28 million tokens. They achieve 80% – 86% accuracy for all of their attempts.

Malmasi et al. (2015) do Arabic dialect identification from text corpora, including the Multi-Dialect Parallel Corpus of Arabic (Bouamor et al., 2014) and the Arabic Online Commentary database (Zaidan and Callison-Burch, 2011).

Hanani et al. (2015) perform recognition of several Palestinian regional accents, evaluating four different acoustic models, achieving 81.5% accuracy for their best system, an I-vector framework with 64 Gaussian components.

Ali et al. (2016) developed the corpus on which the DSL Arabic shared task is based. Their own dialect detection efforts depended largely on acoustical cues.

Arabic dialect recognition appeared in the 2016 edition of the workshop's shared task (Malmasi et al., 2016). The shared task data was text-only. Our classifiers (Hanani et al., 2016) for that task gave middling performance relative to other entrants, but the best classifiers (Malmasi and Zampieri, 2016; Ionescu and Popescu, 2016) for the shared task performed far below the best results reported by some of the preceding researchers. Part of the reason must be that the amount of training data for the workshop is much smaller than that used by some of the other researchers; the workshop data also did not include the audio recordings on which the transcripts are based.

3 Methodology and Data

The Arabic training and test data are excerpted from the corpus described in Ali et al. (2016). The provided `.ivec` files contain selected audio features; a list of `.wav` files was also provided with the training data, but not included in the distribution, presumably for reasons of space. We also downloaded the `.wav` files, and build several classifiers using them, which were combined into our run3 on the test data.

The Swiss-German data is excerpted from Samardzic et al. (2016).

The two data sources differ in their presentation. The Arabic data seems to attempt to present words in dictionary spelling, independent of how they were pronounced. If a word is not present in the dictionary, the transcript shows <UNK>, not a phonetic transcription. For example, the particle هذا`h*A` *that*, which is frequently pronounced هدا `hdA` *that* in Levantine, is always presented in its MSA written form, which is of course how Levantine speakers would write it – since they are educated to write standard Arabic, not to indicate their regional dialect.

[1] Modern Standard Arabic – the language of television news programs.

[2] Egyptian dialect

In contrast, the Swiss-German transcripts are intended for scholarly study of the contrasts between dialects. They use the transcription guidelines of Dieth (1986) for this purpose. The spellings of words attempt to present those dialect contrasts, so that the same standard German word may be spelled in numerous different ways, depending on the pronunciation. There is an attempt in the transcription toward standardization, but it is within the dialect, not aimed toward unifying dialects. The result is that there is a large apparent vocabulary difference between Swiss-German dialects, whereas the corresponding vocabulary differences between Arabic dialects correspond to usage shifts, rather than pronunciation shifts.

In the subsections which follow, we present the methodology of each of our classifiers. We combined several classifiers for each run, and we present the fusion classifiers as well.

3.1 Word-focused baseline

This baseline classifier was provided with the training data. It treats each training or test segment as a bag of words and n-grams. The script which runs it preprocesses each segment into a line of integers and occurrence counts, with each integer representing a single word or bigram. (The setup program can be configured to use n-grams as features up to n=6. However, if n is greater than 3, the accuracy of the classifier declines; the difference between n=2 and n=3 doesn't look significant, so we followed the default, using n=2.) The resulting files are processed by Thorsten Joachim's `svm_multiclass_learn` (Tsochantaridis et al., 2004; Joachims, 2008) program which produces a model file. This can be used with the `svm_multiclass_classify` program to provide an output for each test segment with a best guess for the segment class and the scores for all classes.

This word-focused baseline classifier was combined with the I-vector baseline classifier, the word-entropy classifier, and the character string entropy classifier for ADI run1.

Applying the word-focussed baseline classifier to the ADI development data gives an accuracy of 48%.

3.2 I-vector baseline

This baseline classifier was also provided with the training data. It also uses `svm_multiclass`. The input files consist of one line per training or test segment, with the class as the first integer on the line, and the integers from 1 to 400, in order, each with a real-valued feature value. The output file, like that of the word-focused baseline classifier, contains one line for each test segment, with the first integer on the line the class with the highest score, followed by scores for this segment for each class.

This classifier, applied standalone to the ADI development data, gets an accuracy of 57%.

3.3 Word entropy

This classifier reads through the training file, and builds a table of word, bigram, and trigram frequencies for each dialect.[3]

Using the frequencies as estimates of the true probability of the n-gram occurring in a test sample if the sample is in the corresponding dialect, it estimates the probability that the sample is in each of the dialects which appeared in training. In other words, it creates n-gram language models for each dialect, and for each test sentence chooses the dialect with the best cross-entropy. The classifier can be configured to ignore words which occur less than m times. It can write files either in vardial3 submission format, or in the input format used by the Focal Multiclass toolkit, for combining its results with other classifiers.

This classifier is used alone for our GDI run1, and in combination for our ADI run1.

On the ADI development data, this classifier gives an accuracy of 52%, and on 723 lines of reserved GDI data, it gives an accuracy of 84%.

On the test data, it is used standalone only on GDI run1, where it shows an accuracy of 56%.

3.4 Character-string entropy

This classifier ignores word boundaries.[4] It accumulates statistics for all of the strings up to twenty-five bytes long in the training file, except for those strings which end in a UTF-8 sequence which is broken by the 25-byte boundary. For each dialect, it greedily attempts to pave the test strings with strings from training, trying the longest strings first. Once the test segment is completely covered by strings seen in training, a log-probability is

[3]The classifier is implemented by the file `https://github.com/StephenETaylor/vardial4/blob/master/wordfreq.py`.

[4]The character-string entropy classifier is implemented by `https://github.com/StephenETaylor/vardial4/blob/master/chars.c`.

computed by adding log-frequencies for the covering strings. The dialect with the largest log-probability for the segment is selected as the dialect. When the classifier is configured for saving scores, the log-probabilities for each dialect are the scores for the test segment.

On the ADI development data, this classifier has an accuracy of 44%. On the 723 lines of reserved GDI training data, this classifier has an accuracy of 79%.

For the test runs, it is used standalone on GDI run2, where it achieves an accuracy of 63%.

3.5 Fusing estimates

To combine the estimates of the four classifiers used for ADI run1, we used the Focal Multi-class Toolkit (Brümmer, 2007), which is written in MATLAB. We ported it to Octave (Eaton and others, 2012), a trivial effort.[5]

The toolkit script calibrates the scores of the four classifiers (the winning class is always the largest, but the scores aren't necessarily in the same range, let alone a probability distribution) then applies logistic regression to fit the various scores to the known correct answers for the development data. The same fitting is then used to combine the scores of the classifiers on the test data.

It accepts files in precisely the format produced by the baseline classifiers, so we modified the word-entropy classifier and the character n-gram entropy classifier to produce files in the same format. We wrote python scripts to convert the output to the expected format for the workshop test runs.

3.6 ADI run2 combination

We used a combination of 4 classifiers on the system level. All four of these classifiers used the same features: character unigrams, bigrams, and trigrams derived from the training data, presented to the software as a feature vector.

Systems are:

- Naive Bayes with multinomial distribution
- SVM with RBF kernel
- Linear logistic regression
- Random forests with 300 trees

[5]See the goal `test.f4` in the file `https://github.com/StephenETaylor/vardial4/blob/master/v17/dialectID/Makefile`

All these classifiers were trained on the training dataset part and tested on the development dataset part. The feature vector used was built based on character trigram model combined with word unigram model and word bigram model. The final output was generated by applying voting (max was chosen) on output of the four classifiers for each class label. To build the language models (character trigram, word unigram, and word bigram) to prepare the feature vector and to do the classification process we used the Weka toolkit (Hall et al., 2009), which is written in java.

On the ADI development data, this system gave an accuracy of 52.03%. It is used in ADI run2, where it achieves an accuracy of 32%.

3.7 Acoustic processing: ADI run3

3.7.1 Front-end Processing

Each utterance is divided into short frames by a 20-ms window progressing at a 10-ms frame rate; then 19 Mel-scale Cepstral Coefficients (MFCC) are extracted from each speech frame. Next, Shifted-Delta Cepstra (SDC) with 7-3-1-7 configuration, are computed and appended to the MFCC feature vectors resulting in feature vectors with dimension equal to 68. RASTA filtration is applied to the power spectra. A simple energy-based voice activity detection (VAD) was performed to discard the non-speech frames.

Finally, Cepstral mean and variance normalization (CMVN) was applied on the resulting 68-dimensional feature vectors.

3.7.2 GMM-UBM AID

A Universal Background Model (UBM) GMM (Gaussian Mixture Model) is trained on the acoustic features (68 feature vectors) extracted from all training dataset of all Arabic dialects. The K-means clustering algorithm is used for finding initial parameters of UBM GMM (means, diagonal covariance matrices and weights).

A dialect-dependent GMM is obtained by MAP adaptation (means only) of the UBM using the dialect specific enrollment features. This results in one UBM model and one dialect-dependent model for each of the target dialects. We have tried different numbers of Gaussians: 64, 256 and 2048. Applying these three systems to the ADI development data gives an accuracy of 35.6%, 36% and 40.16%, respectively.

3.7.3 GMM Tokenization

This system is similar to the Phonotactic systems in which a sequence of phones is extracted from the speech waveform using a phone recognizer. In GMM tokenization, the phone recognizer is replaced by a Multi-Dialect Model (MDM), which is a GMM trained on training data of all dialects (same UBM used in the GMM-UBM system described above). For each utterance, a sequence of GMM components (tokens) is extracted by representing each acoustic vector with the GMM component which gives the highest log likelihood.

The n-gram components of the sequence of tokens generated from an utterance U can be represented as a D-dimensional vector p where, D is the number of all n-grams (in our case GMM components), C_j is the jth n-gram and the probability p_j of C_j is estimated using counts of n-grams,

$$p_j = \frac{Count(C_j)}{\sum_i Count(C_i)} \tag{1}$$

where the sum in (1) is performed over all n-grams and $Count(C_j)$ is the number of times the n-gram C_j occurs in the produced sequence of tokens.

Before we apply the SVM, the probabilities of the n-grams are estimated for each utterance. Then, these probabilities are weighted to emphasize the most discriminative components (i.e. those which occur frequently in one dialect and infrequently in others). The n-gram components which are common in most dialects, such as silence or common phones, contain little discriminative information and are de-emphasized. Numerous weighting techniques are available for this purpose, such as the Inverse Document Frequency (IDF) from Information Retrieval (IR) and the Log-Likelihood Ratio (LLR) weighting. The LLR weighting w_j for component C_j is given by:

$$w_j = g_j \left(\frac{1}{P(C_j|all)} \right) \tag{2}$$

where g_j is a function used to smooth and compress the dynamic range (for example, $g_j(x) = \sqrt{x}$, or $g_j(x) = log(x) + 1$). $p(C_j/all)$ is the probability of n-gram component C_j across all dialects. The components which have zero occupancy in all dialects are removed since they do not carry any useful information. A benefit of discarding these non-visited components is that it reduces the feature dimension dramatically, particularly for the high order n-gram system as the dimension of the n-gram increases exponentially $O(M^n)$ with GMM model order (M).

In addition, a feature selection technique is needed to minimize the number of n-gram components by keeping only those which are most discriminative. This is particularly necessary in high order n-gram systems because the dimension is increased exponentially. Consequently, reducing the number of n-gram components decreases the computational cost and the required amount of memory. A powerful iterative feature selection algorithm based on the SVM is proposed by Guyon et al. (2002). This is applied to phone-based language recognition with discriminative keyword selection in Richardson and Campbell (2008), where more details can be found. A similar algorithm is applied on the bigram data of the GMM tokens.

For GMM tokenization, we have used UBM with 256 and 2048 order. Due to resources limitation, bigram and unigram of UBM with 256 components, but only unigram of UBM with 2048 components have been implemented. When applied to the ADI development data, the unigram, bigram of 256 UBM and unigram of 2048 are 42%, 45.15% and 46.85%, respectively.

3.7.4 I-vector based system

I-vectors is a technique introduced in Dehak et al. (2011) for speaker identification. This technique has also been proven to work well in language and dialect identification (Martínez et al., 2011; Hanani et al., 2015). An I-vector classifier is based on a configuration determined by the size of the UBM, the number of factor dimensions for the total variability subspace, as well as the various compensation methods to attenuate within-dialect variability.

Feature vectors of each utterance in the training data are used for adapting means of UBM (which is trained on all available training data) in order to estimate an utterance dependent GMM using eigenvoice adaptation technique.

The eigenvoice adaptation technique assumes that all the pertinent variability is captured by a low rank rectangular, total variability matrix T. Then the GMM supervector (vector created by concatenating all mean vectors from the utterance dependent GMM) for a given utterance can be modeled as follows:

$$M = m + Tx + \epsilon \tag{3}$$

where m is the UBM supervector, the I-vector x is a random vector having a normal distribution $N(0, I)$, and the residual noise term $\epsilon \sim N(0, \Sigma)$ models the variability not captured by the matrix T. In training total variability matrix for dialect recognition, we assume that every utterance for a given dialect is considered a different class. Additional details on the I-vector extraction procedure are described in Dehak et al. (2011).

Linear Discriminant Analysis (LDA) is used for reducing I-vectors dimension. The LDA procedure consists of finding the basis that maximizes the between classes variability while minimizing the intra-dialect variability.

Recently, Gaussian-PLDA has been used to make the I-vector distribution more normal, which improves performance of I-vector system based on standard LDA Bousquet et al. (2012). A Gaussian-PLDA model has been trained on dimensionally-reduced I-vectors of training data, and then used for scoring in our I-vector system. In addition to the text transcription and wav files of each utterance, 400-dimensional I-vectors are provided with the dataset released for VarDial 2017. These I-vectors are extracted using a UBM with 2048 components and Bottleneck features instead of the traditional MFCC and SDC (Shifted Delta Cepstral) acoustic features. More details about the provided I-vectors can be found in Ali et al. (2016). When applied to the ADI development data and with setting LDA dimension to four, the accuracy is 58%.

3.7.5 Acoustic Overall system

The best four acoustic sub-systems: GMM-UBM with 2048 components; bigram of GMM tokenization with 256 components; unigram with 2048 components; and I-vector system, are fused together to get the overall acoustic system, using Focal multi-class linear logistic regression (Brümmer, 2007). The fusion parameters were trained on the ADI Development data. The resulting system was used to classify the ADI testing data (run 3 in the results of ADI task). In order to have an idea how well the overall acoustic system compared with the sub-systems, we divided the development data of each dialect into two parts (nearly equally). The fusion parameters were estimated using one part and applied to the second part and vice versa. In this way, we got the system performance on the development data without overlapping between training and evaluation data. The accuracy of the fused (overall acoustic) system on the development data was 61%.

Table 1: Classifier Accuracy on ADI Development Data, Test Sest

Section Described	**Dev. Set**	**Test Set**
3.1	0.48	
3.2	0.57	
3.4	0.44	
3.3	0.52	
3.5		0.63
3.6	0.52	0.32
3.7.2	0.40	
3.7.3 (256 bigrams)	0.45	
3.7.3 (2048 unigrams)	0.47	
3.7.4	0.58	
3.7.5	0.61	0.59

4 Results

There were six teams participating in the Arabic Dialect Identification task for 2017; in contrast, there were eighteen for 2016.

Given the reduced field, the rise of our team, AHAQST, from 14th to 4th place, can be ascribed in part to decreased competition! However, all the teams who entered both shared tasks posted scores for 2017 much better than their scores for 2016.

Table 2 shows the best results for each team for the two years.

Table 2: 2017 versus 2016 ADI results

Team	**F1 2017**	**F1 2016**
unibuckernel	0.763	0.5131[6]
MAZA	0.717	0.5132[7]
tubasfs	0.697	0.472[8]
ahaqst	0.628	0.426[9]
qcri_mit	0.616	-new-
deepCybErNet	0.574	-new-

In our own case, some of the improvement is due to combining the acoustic and the text data. Table 3 shows our three ADI runs. run1 and run3 both use acoustic data, whereas run2 does not; run3 uses only acoustic data, while run1 uses both kinds.

The Swiss-German task was new this year, and attracted attention from teams who also entered other tasks, as shown in Table 4.

[6] Ionescu and Popescu (2016)
[7] Malmasi and Zampieri (2016)
[8] Çöltekin and Rama (2016)
[9] Hanani et al. (2016)

Table 3: Performance of our merged classifiers

Run (Data)	Accuracy	F1 (mic)	F1 (wt'd)
2 (Text)	0.3231	0.3231	0.3137
3 (Acoust.)	0.5932	0.5932	0.5861
1 (both)	0.6287	0.6287	0.628

Table 4: Participation of Swiss-German teams in other tasks

Team	GDI	DSL	ADI
MAZA	1		2
CECL	2	1	
CLUZH	3		
qcri_mit	4		5
unibuckernel	5		1
tubasfs	6	4	3
ahaqst	7		4
Citius_Ixa_Imaxin	8	9	
XAC_Bayesline	9	3	
deepCybErNet	10	11	6

It's interesting to note the imperfect correlations between the tasks, but they are less interesting than the table makes them look, because on the GDI task the accuracy for the best run of all the teams except for the first and the last is within a range of four percentage points.

Table 5 shows our two runs were more widely separated than that, but only the best run for each team contributes to the rank above.

Table 5: AHAQST results on GDI task

Run	Accuracy	F1 (micro)	F1 (weighted)
run1	0.5621	0.5621	0.5484
run2	0.6289	0.6289	0.6143

The top run for the GDI task had an accuracy of 68%, and the bottom an accuraccy of 26%. Omitting the bottom outlier, the weighted F1 scores of the other nine teams are all within 1.35 standard deviations of the mean. The range of values is not nearly so interesting as we see for the ADI task.

We would expect the GDI task to be easier, since only four classes need be distinguished, versus five for the ADI task, but it looks like there are other factors at work. Since only the CLUZH team entered only the GDI task, it may be that other teams devoted less effort to the task, focussing their primary attention on one of the other tasks. Or it may be that there is something else at work.

Since our own classifiers performed much better on our reserved training data, it may be that the Swiss-German corpus is inhomogenous, and that the test data is drawn from a part of the corpus which is different in some way from the training data.

However, a simpler theory for differing performance is a topic bias. If the training sentences are drawn from coherent conversation, one would expect neighboring sentences to have theme words in common. Since both turns of a conversation will normally be entirely in one dialect, rare theme words are trained as dialect-unique, when in fact they may have no relevance to dialect. Of course, even when not dialect-specific, theme words may still be helpful for distinguishing dialect. In the training data, "Zürich" occurs only in instances of the ZH dialect. While someone from Berne may talk about Zürich, Berne is far more likely to come up in their conversation. Similarly a particular restaurant or street is probably indicative of their neighborhood.

5 Discussion

We were pleased to be able to so quickly put into practice some of the ideas we considered for the 2016 workshop. But we ran out of time to implement others. For example, deep learning has fared poorly in the shared tasks in the past, including in our 2016 submission, but considering its success in other machine learning tasks, it seems possible that there is an approach which will fare better, even if the (relatively small by neural-network standards) 1-2 megabyte training sets typical of the two ADI tasks and one GDI task we've seen continue to prevail.

Some of our negative results seem surprising. Why does including larger word n-grams actually hurt recall? At worst it is noise, and there are plenty of other sources of noise.

We'd like to revisit tools which can provide explanations of their behavior. For example, for 2016, one of our classifiers reported that the word

يعني yEny *that is* which is very common in all varieties of Arabic was actually a useful predictor for dialect, because although it is not uncommon in MSA, it is *very* common in all the dialects. The explanation doesn't greatly improve the class discrimination, but it is a nice converational tidbit.

References

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 *Proceedings of Interspeech 2016*, pages 2934–2938.

Fadi Biadsy, Julia Hirschberg, and Nizar Habash. 2009. Spoken arabic dialect identification using phonotactic modeling. In *Proceedings of the EACL Workshop on Computational Approaches to Semitic Languages*, pages 53–61.

Houda Bouamor, Nizar Habash, and Kemal Oflazer. 2014. A multidialectal parallel corpus of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC14)*. European Language Resources Association (ELRA), May.

Pierre-Michel Bousquet, Anthony Larcher, Driss Matrouf, Jean-François Bonastre, and Oldrich Plchot. 2012. Variance-spectra based normalization for i-vector standard and probabilistic linear discriminant analysis. In *Odyssey: The Speaker and Language Recognition Workshop*, pages 157–164.

Niko Brümmer. 2007. Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scores tutorial and user manual. downloaded in January 2017 from https://sites.google.com/site/nikobrummer/focalmulticlass.

Çağri Çöltekin and Taraka Rama. 2016. Discriminating Similar Languages with Linear SVMs and Neural Networks. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 15–24, Osaka, Japan.

Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2011. Front end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, 19:788–798.

Eugen Dieth. 1986. *Schwyzertütschi Dialäktschrift: Dieth-Schreibung*. Sauerländer Verlage, Aarau, Switzerland.

John W. Eaton et al. 2012. Gnu octave program for scientific calculation. current version available from `http://www.octave.org`.

Heba ElFardy and Mona Diab. 2013. Sentence level dialect identification in Arabic. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 456–461.

Charles A. Ferguson. 1959. Diglossia. *WORD*, 15(2):325–340.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1).

Abualsoud Hanani, Hanna Basha, Yasmeen Sharaf, and Stephen Taylor. 2015. Palestinian Arabic regional accent recognition. In *The 8th International Conference on Speech Technology and Human-Computer Dialogue*.

Abualsoud Hanani, Aziz Qaroush, and Stephen Taylor. 2016. Classifying ASR Transcriptions According to Arabic Dialect. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 126–134, Osaka, Japan.

Radu Tudor Ionescu and Marius Popescu. 2016. UnibucKernel: An Approach for Arabic Dialect Identification Based on Multiple String Kernels. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 135–144, Osaka, Japan.

Thorsten Joachims. 2008. Multi-class support vector machine. downloaded in January 2017 from `http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html`.

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, Eshrag Refaee, and Mark Dras. 2015. 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, 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.

David Martínez, Oldrich Plchot, Lukás Burget, Ondrej Glembek, and Pavel Matejka. 2011. Language recognition in i-vectors space. In *Interspeech*, pages 861–864.

Maryam Najafian, Andrea DeMarco, Stephen Cox, and Martin Russell. 2014. Unsupervised model selection for recognition of regional accented speech. In *Proceedings of Interspeech 2014*.

Fred S. Richardson and William M. Campbell. 2008. Language recognition with discriminative keyword selection. In *ICASSP'08*, pages 4145–4148.

Tanja Samardzic, 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).

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 104–, New York, NY, USA. ACM.

Omar F. Zaidan and Chris Callison-Burch. 2011. The Arabic Online Commentary dataset: An annotated dataset of informal Arabic with high dialectal content. In *Proceedings of ACL*, pages 37–41.

Omar F. Zaidan and Chris Callison-Burch. 2014. Arabic dialect identification. *Computational Linguistics*, 40(1):171–202.

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.