HelvetiaDial - Predicting Swiss Dialects

Raphael Schilling
raphaejs@stud.ntnu.no

Abstract

For quite some time automatic language identification has been investigated. This is especially challenging when dealing with similar languages or even dialects. In this context we introduce HelvetiaDial, a BERT and Byte-Pair TF-IDF based solution for classifying Swiss German dialects. HelvetiaDial was trained and tested based on the data of the *German Dialect Identification* tasks of the VarDial 2019 workshop. We also compare how our approach performs compared to the approaches of the teams that were part of the workshop.

1 Introduction

In this paper, we present HelvetiaDial, an approach to distinguish between four German Swiss dialects. Automatic language identification is a long-studied topic, as Jauhiainen et al. (2019b) summarize. Dialect identification is fundamentally the same as language identification. However, very similar languages, and hence also dialects, are harder to tell apart than very different languages. For this reason, there exists since 2016 the annual VarDial-workshop. Our approach aims to solve the *German Dialect Identification* task of the 2019 edition of *VarDial* (Zampieri et al. (2019)). The question this publication intends to answer is whether BERT embeddings (Devlin et al. (2019)) are suitable to distinguish Swiss German dialects. No Swiss German-specific BERT model was available in 2019, but in recent years at least 2 publicly available BERT models emerged, which target languages spoken in Switzerland. Another feature embedding that we used is Term-Frequency Inverse-Document-Frequency based on Byte-Pair tokenization. In section 2 we give an introduction to the *German Dialect Identification* task, talk about the four target dialects and explain the technologies that we used. Then we summarize the solutions of the teams that have participated in the task. Section 4 describes how we came up with our solution. The last sections are about our results, a discussion, a conclusion, and future work.

2 Background

This section describes the problem followed by a brief introduction to the four dialects and ends with various techniques and methodologies used in this paper.

2.1 Problem description

The *German Dialect Identification* task (GDI) is from the 2019 edition of VarDial (Zampieri et al. (2019)). The challenge is to classify the dialects spoken in the Swiss cities Lucerne, Basel, Bern, and Zurich. In total, the train set, dev set (we call it val set), and test set consists of sentences from 43 oral interviews in the 4 different dialects. Each sentence was transcribed using the writing system *Dieth-Schreibung* (Dieth and Schmid-Cadalbert (1986)), which was specifically designed to transcribe Swiss German dialects. Also, a representation of the audio was delivered, see subsubsection 2.3.3. Each speaker appears only in one of the three sets.

2.2 Swiss German

Zurich-dialect belongs to Eastern High Alemannic, Basel-dialect and Bern-dialect belong to West High Alemannic, while Lucerne-dialect is between those. All of those dialects belong to Swiss German. Interestingly, the name of the challenge is not *Swiss* Dialect Identification but *German* Dialect Identification.

This suggests that Swiss German is not a separate language, but a dialect itself. The Constitution of the Switzerland declares 4 languages to be the national languages, one of which no other country has as its national language. In this way, Switzerland (a landlocked country) resists the Weinreich witticism: "A language is a dialect with an army and *navy*" (Brown and Ogilvie, 2010, p.793). However, Switzerland does not resist the witticism by having Swiss German as its national language but a small language called Romansh. Swiss German is not one of the 4 national languages defined in the federal's constitution: German, French, Italian, and Romansh. This might be the reason why this task is called “German Dialect Identification”.

2.2.1 Distinguishing phonetic features

We highlight now, based on Siebenhaar and Voegeli (1997), phonetic features that help to distinguish between the 4 dialects.

The ’L’ is pronounced darker in Swiss German dialects than in other German dialects and is sometimes pronounced as a ’U’ in Bern and Lucerne.

In Standard and Swiss German exist three ways of pronouncing the ’R’: ’Zungenspitzen-r’ (alveolar trill), the Halszäpfchen-r (uvular trill), and the Reibe-r (fricative r). A specialty of Zurich and Bern is that only alveolar trill is used.

For pronouncing ’K’, in most Swiss German dialects one uses the fricative [x] or [X]. However, Basel is an exception for that, which uses [k^h], which is closer to the Standard German [k].

Due to these pronunciation differences, there is reason to believe that taking audio features into account helps with classification.

2.3 Features

To predict the dialect, one needs to transform the text into a representation that our classifiers can work with. All three following techniques represent the sentence as a vector.

2.3.1 Term-Frequency Inverse-Document-Frequency

One can create a bag-of-words representation for each document. In a bag-of-words representation, each dimension of a vector corresponds to exactly one term in the whole corpus. We used the Term-Frequency Inverse-Document-Frequency (TF-IDF) implementation of scikit-learn (Pedregosa et al. (2011)), which was implemented as explained in Manning et al. (2008):

$$tf\_idf(t) = tf(t) * log(\frac{N}{df(t)+1})$$

The term-frequency *tf(t)* is the count of the term *t* in the document and the inverse-document-frequency *idf(t)* is the number of documents that contain *t*. The overall number of documents is *N*.

Byte-Pair Tokenisation TF-IDF requires that the sentences are separated into terms, respectively tokens. We tokenized the sentences using the Byte-Pair tokenization, introduced by Sennrich et al. (2016). It creates tokens out of frequently appearing symbol sequences. Very frequent words end up as a single token, and less frequent words are split up into smaller parts. We expect that dialect-specific word endings end up as separate tokens, that help the classifiers to distinguish the dialects.

2.3.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language model introduced by Devlin et al. (2019). It creates for each input token a hidden state. These hidden states can be used for a variety of tasks, including classification. There exist BERT models for multiple languages. “Swiss Language Model” (by Michael Jungo) [1] is a BERT model fine-tuned on Swiss German sentences from the Leipzig Corpus (Goldhahn et al. (2012)) and Swiss German sentences crawled by Linder et al. (2020). There exists also a newer model called SwissBERT (Vamvas et al. (2023)), which was fine-tuned on German, French, Italian, and Romansh. However, the German data is from Swiss news articles which do not consist of Swiss German dialect but only of standard German.

[1]https://github.com/jungomi/swiss-language-model

2.3.3 iVectors

iVectors are a vector representation of audio-recorded speech. They were introduced by Dehak et al. (2009) for speaker verification. Ali et al. (2016) showed that iVectors help classify Arabic dialects. Based on this finding, the organizers of VarDial 2019 decided on providing 400-dimensional iVectors alongside the written sentences.

2.4 Classifiers

We will now describe the used classifiers whose input are the previously mentioned features.

2.4.1 Gaussian Naive Bayes

Gaussian Naive Bayes (GaussianNB) works with the assumption, that for each label, there exists a unique multidimensional Gaussian distribution. From this distribution, the features are drawn. GaussianNB assumes, that the individual features are independent, which means that the covariance matrix is diagonal. Although this seems to be a very strong assumption, previous research shows good results using Naive Bayes. Hand and Yu (2001) make a case on why this independence assumption leads to good results. One advantage of GaussianNB is, that the training is linear in the number of samples.

2.4.2 Support Vector Machines

First implemented for performing digit recognition (Sparck Jones (1988)), Support Vector Machines (SVM) show good performance in classification tasks. Instead of creating a probabilistic model like GaussianNB, where decision boundaries are implicit, Support Vector Machines explicitly define decision boundaries. The goal is to have a large margin between the decision boundary and the samples of the categories. To extend SVM to non-separable data, Cortes and Vapnik (1995) allowed for samples to lie in the margin and even on the wrong side of the decision boundary. For this, the parameter C was introduced. A large C leads to a smaller boundary, but fewer outliers in the sample set. A small C increases the margin but accepts more outliers. Hence C can be used as regularisation and needs to be optimized. We tested the following two kernels. The used kernel decides which form the decision boundary can take.

Linear SVM In its standard form, SVM uses linear decision boundaries. The sklearn class *LinearSVC* implements that. The advantage is fast speed. A problem is, that data is often not linearly separable.

Radial-Basis-Function SVM In Sparck Jones (1988) Radial Basis Function (rbf) has been proposed among other kernels. It can separate non-linear separable data. Rbf is the default kernel when using *SVC* from sklearn. From now on, we will use *SVC* to denote the sklearn SVM implementation with rbf kernel.

2.5 Macro averaged F1 Score

For evaluating the GDI task, the organizers decided on calculating the macro averaged F1 score of the test set predictions. We first show the F1 score for the binary case and then explain the macro-averaged F1 score.

Binary F1 Score Default metrics to evaluate classifiers are Recall R and Precision P. To combine those two values into one, (Van Rijsbergen, 1979, p.134) came up with effectiveness measure:

$$E = 1 - \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = 1 - \frac{\frac{1}{\alpha}PR}{R + \frac{1-\alpha}{\alpha}P} = 1 - \frac{(1+b^2)PR}{R + b^2P}$$

The last equation holds if one sets $\alpha := \frac{1}{1+\beta^2}$, like (Frakes and Baeza-Yates (1992)) does. Based on this, the F-Score was introduced by Chinchor (1992):

$$F = 1 - E = \frac{(1+b^2)PR}{R + b^2P}$$

If we set $b = 1$, we call this F1 score.

Macro Average If we have more than 2 classes we can use the macro averaged F1 score. This is calculated by first calculating the F1 score for each class separately. The results are then summed up and divided by the number of classes. There is also a micro-averaged F1 score, which does a weighted average. However, we only use the macro averaged F1 score since it is the measure used by VarDial 2019 for the GDI task.

3 Related Work

Since the *German Dialect Identification* campaign already took place in 2019, we give an overview of the solutions of the participating teams. We will then mention a publication that used audio features successfully for classifying dialects in the past.

3.1 GDI 2019

In the VarDial 2019 report is a table that shows the best results for each challenge. The best F1 score in GDI was achieved by the team *tearsofjoy* (Wu et al. (2019)) with a single linear SVM. The input to the SVM was a concatenation of BM25 weighted n-grams of words, n-grams of characters, and the 400-dimensional iVectors. They also experimented with an ensemble of SVMs, which did not produce as good results as the single SVM.

The team *SUKI* (Jauhiainen et al. (2019a)) achieved the second-best result. They developed an algorithm called HeLi 2.0, which is based on words and n-grams. The interesting part of HeLi 2.0 is, that words and n-grams that do not appear in the train-set, get incorporated during prediction into the used models if the surrounding text is predicted with high certainty.

The third best result was achieved by the team *TwistBytes* (Benites et al. (2019)). They used n-grams, words, and iVectors as embeddings and trained for each feature an SVM. The output of those SVM was aggregated by a meta-SVM. Similar to HeLi, TwistBytes also adjusted the classifiers during prediction. This helps especially with the iVectors. Since there is no speaker overlap between the train test and dev set, it is advantageous if the iVectors SVM can adjust to the test set speakers.

Team BAM (Butnaru (2019)) reached 4th place. They combined three neuronal models using a voting schema. The three models used: String Kernels, a character-level CNN, and a Long Short-Term Memory network.

For team dkosmajac exist no publication. According to the VarDial 2019 report (Zampieri et al. (2019)), dkosmajac classifies the iVectors with a quadratic discriminant analysis classifier and the text with random forests. The results are combined using a random forest classifier.

Also for team ghpaetzold exists no publication. The VarDial 2019 report (Zampieri et al. (2019)) states that they used recurrent neural networks to create sentence and word embeddings.

3.2 Dialect Prediction

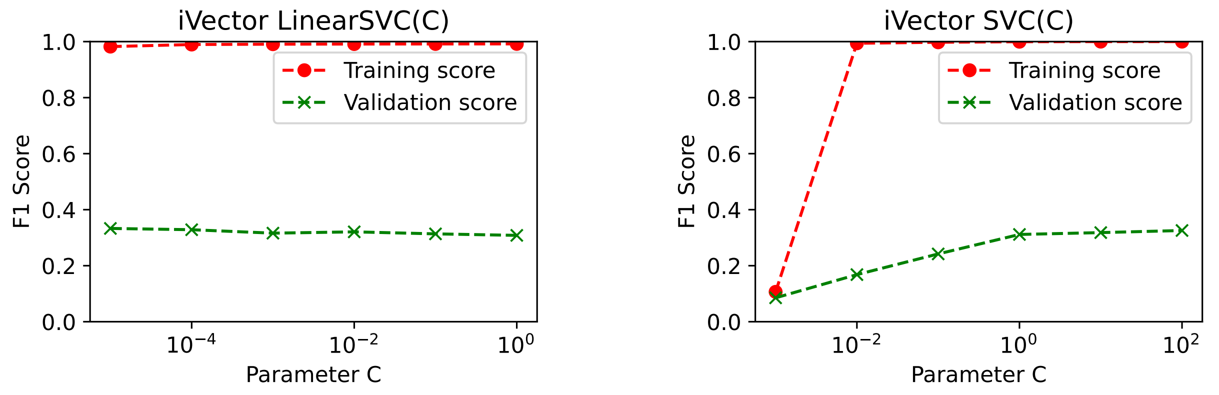
Ali et al. (2016) tested multiple approaches to distinguish between Arabic dialects. Their approaches use both phonetic and lexical features. What makes this publication especially interesting is that they also used iVectors. Incorporating the iVectors increased the performance of their system.

4 Architecture

For each of the feature vectors (BERT-embedding, Byte-Pair-TF-IDF, and iVectors), our model separately classifies the sentences (first-stage). A meta-classifier combines these predictions to generate the overall prediction (second-stage). In this section, we first discuss how we chose the first-stage classifiers. Afterward, we discuss how the second-stage classifier was chosen. The decisions were driven by the F1 score of the development set.

4.1 First stage classifiers

The first-stage classifiers should separately provide solid predictions. Another requirement is that the first-stage classifiers can create feature vectors themselves. These intermediate feature vectors are fed



F1-score	GaussianNB	LinearSVC(1e-05)	SVC(100)
train	0.98	0.98	1.0
dev	**0.43**	0.33	0.33

Figure 1: Train and dev F1 score for three classifiers using iVectors. GaussianNB shows the best val score. For GaussianNB exists no graph since we did not optimize any parameter.

into the second stage. For all three feature vectors we examine the following sklearn classifiers over the mentioned parameters:

- LinearSVC: SVM with linear kernel. Parameter to optimize: C of SVM
- SVC: SVM with rbf kernel. Parameter to optimize: C
- GaussianNB: Gaussian-Naive-Bayes. No parameter to optimize.

4.1.1 iVectors

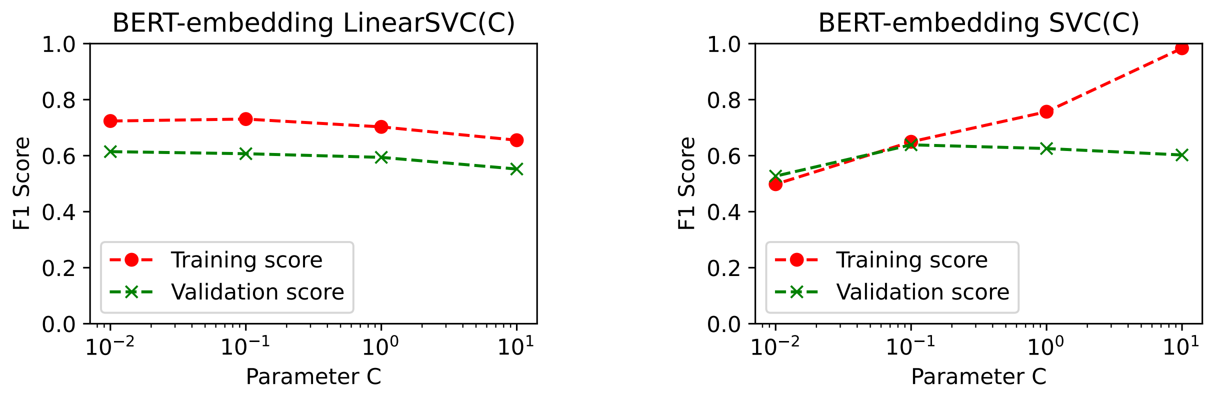
In subsection 2.2 we gave examples of pronunciation differences between the 4 dialects. Hence it is reasonable to test if an audio representation helps to classify. Since the competition organizers already provide the iVector representation, we don't have to create an audio feature-embedding. Each iVector consists of 400 dimensions. In Figure 1 we show the results of the test F1 score and the dev F1-score. For all three classifiers, the train F1-score is very high. However, this seems to be overfitting since none of the classifiers reach a high val F1-score. GaussianNB delivers the highest val F1-score which is only 0.43. Due to this low score, we will later see, that the iVectors don't contribute to the overall classification. We experimented with applying *TruncatedSVD* on the 400-dimensional iVectors to reduce the gap between the train and dev score. However, this dimension reduction only worsened the train score without improving the dev score.

4.1.2 BERT-embedding

The next first-stage classifier that we examine uses the "Swiss Language Model", the BERT model from Michael Jungo. For each input text, we take the last hidden state of the BERT symbol [CLS]. This gives us a 768-dimensional vector. Figure 2 displays the results for the three examined classifiers. Again, GaussianNB reaches a better val F1-score than the two SVM algorithms. Hence we will use GaussianNB as the first-stage classifier. We also tested "SwissBERT" Vamvas et al. (2023). However, the performance was significantly worse, since it was not trained on Swiss German but on Standard German data.

4.1.3 Byte-Pair TF-IDF

Our last first-stage classifier encodes the sentences using Byte-Pair encoding with *vocab_size = 1000*. The result is vectorized using TF-IDF. In Figure 3 one can see that in this case, GaussianNB delivers the



F1-score	GaussianNB	LinearSVC(0.01)	SVC(0.1)
train	0.59	0.72	0.65
dev	**0.67**	0.61	0.64

Figure 2: Train and dev F1-score for three classifiers over using BERT-embedding. GaussianNB shows the best val score. For GaussianNB exists no graph since we did not optimize any parameter.

worst results and both SVM algorithms result in a similar val F1-score. Since *LinearSVC* is simpler than the rbf-based SVC, we chose *LinearSVC* for this embedding. To validate that the parameter *vocab_size* is chosen appropriately, we calculated the val F1-score for different values using *LinearSVC(C=1.0)*. The val F1-score increases until *vocab_size = 1000* and stagnates afterward.

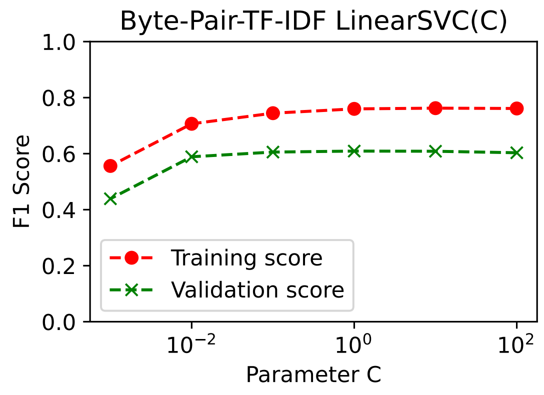
4.2 Final architecture

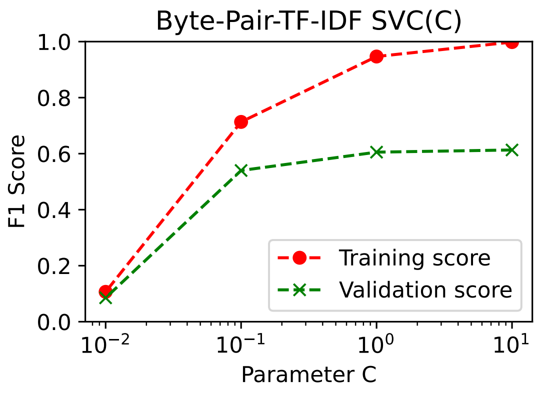
The three first-stage classifiers have to be combined now. *GaussianNB* can predict the probability distribution of the 4 dialects. LinearSVC can't do that, however, we use *LinearSVC.decision_function(X)* which also returns a vector with 4 values. If we concatenate the output of the three first-stage classifiers we retrieve a 12-dimensional intermediate vector. This vector can now be inputted into a final classifier. For that we use GaussianNB. To figure out if all first-stage classifiers add value to the prediction, we tested if leaving out one or two of the three first-stage classifiers would improve the result. The best val F1 score was reached when we excluded the iVectors. This is what we already expected since the result of only using iVectors was poor. The final architecture is sketched in Figure 4 in the appendix.

5 Test Results

During the *German Dialect Identification* (GDI VarDial2019) all competitors had access to all sentences of the train, validation, and test set. However, they only had the ground truth labels of the train and validation set. Every team needed to hand in their predictions for the test set to evaluate the F1 score. To be comparable to those results, we decided to not touch the test ground truth labels until we defined the final architecture and parameters. Otherwise, there would be a risk of overfitting the architecture onto the test data.

The F1 score for the validation set, when using only train data for training, is with our architecture *0.6798*. As stated in the VarDial report (Zampieri et al. (2019)) the organizers encouraged training the final model with both train and validation data. Doing this led to an F1 score of *0.6169* for the test set. This is significantly worse than the F1 score for the validation set using only the train set for training. It is also only slightly better than the baseline of the completion and would rank between the 4th and 5th place. Figure 5 in the appendix shows our HelvetiaDial inserted into the GDI rank table from the VarDial report. Figure 6 in the appendix shows the confusion matrix for the test set predictions from





F1-score	GaussianNB	LinearSVC(1)	SVC(10)
train	0.65	0.76	0.99
dev	0.53	**0.61**	**0.61**

Figure 3: Train and dev F1 score for three classifiers over using Byte-Pair-TF-IDF. LinearSVC and SVC show the best val score.

HelvetiaDial. The most frequent confusion is between Bern and Lucerne.

6 Discussion

The F1 score of the validation data was promising. But the test data led to a significantly worse result. The difference in performance is probably because there is only a small number of speakers per dataset, which also do not overlap between the sets. Hence some of the specialties per dataset and dialect are probably speaker specific instead of dialect-specific. We believe that this effect is largest for iVectors since these were originally designed to identify different speakers and not dialects. Nevertheless, team *tearsofjoy*, *SUKI* and *TwistBytes* have reached significantly better predictions than our architecture. Accordingly, we assume that it is beneficial to adapt the model also based on the test data, like *SUKI* and *TwistBytes* do it. Our goal was to solve the GDI task using different features than the existing solutions. The BERT embeddings are the best of the embeddings that we tested. Also, the Byte-Pair-TF-IDF delivers solid results.

7 Conclusion

In this publication, we tested if BERT embeddings and Byte-Pair-TF-IDF can be used as features to classify Swiss German dialects. We did this by comparing our results to the results of the competitors of the 2019 VarDial *German Dialect Identification*. Even if the validation F1 score was promising, our architecture did not perform as well as the best teams in the challenge. However, we showed that it is possible to use BERT embeddings and Byte-Pair-TF-IDF for Swiss German dialect classification.

8 Future Work

As a next step, it would be interesting to integrate the BERT and Byte-Pair TF-IDF embeddings into the solutions of *tearsofjoy*, *SUKI*, and *TwistBytes*. Another idea would be to cluster, for each dialect separately, the iVectors into as many clusters as there are speakers. Based on the assumption that each speaker ends up in one cluster, one could search for distinguishing dimensions between the clusters. For the actual training and prediction, one could remove those dimensions from the iVectors. This would force the classifiers to focus on dialect-specific dimensions, and not to learn which speaker belongs to which dialect.

References

Ahmed Ali et al. Automatic Dialect Detection in Arabic Broadcast Speech. In *Proc. Interspeech 2016*, pages 2934–2938, 2016. doi:10.21437/Interspeech.2016-1297.

Fernando Benites et al. TwistBytes - identification of cuneiform languages and German dialects at VarDial 2019. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 194–201, Ann Arbor, Michigan, 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-1421. URL https://aclanthology.org/W19-1421.

Keith Brown and Sarah Ogilvie. *Concise encyclopedia of languages of the world*. Elsevier, 2010.

Andrei M. Butnaru. BAM: A combination of deep and shallow models for German dialect identification. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 128–137, Ann Arbor, Michigan, 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-1413. URL https://aclanthology.org/W19-1413.

Nancy Chinchor. Muc-4 evaluation metrics. In *Proceedings of the 4th Conference on Message Understanding*, page 22–29. Association for Computational Linguistics, 1992. ISBN 1558602739. URL https://doi.org/10.3115/1072064.1072067.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, sep 1995. ISSN 0885-6125. doi:10.1023/A:1022627411411. URL https://doi.org/10.1023/A:1022627411411.

Najim Dehak et al. Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification. In *Proc. Interspeech 2009*, pages 1559–1562, 2009. doi:10.21437/Interspeech.2009-385.

Jacob Devlin et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, June 2019. doi:10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Eugen Dieth and Christian Schmid-Cadalbert. Schwyzertütschi dialäktschrift. *Sauerländer, Aarau*, 2, 1986.

William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., USA, 1992. ISBN 0134638379.

Dirk Goldhahn et al. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/327_Paper.pdf.

David J. Hand and Keming Yu. Idiot's bayes: Not so stupid after all? *International Statistical Review / Revue Internationale de Statistique*, 69(3):385–398, 2001. ISSN 03067734, 17515823. URL http://www.jstor.org/stable/1403452.

Tommi Jauhiainen et al. Discriminating between Mandarin Chinese and Swiss-German varieties using adaptive language models. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 178–187, Ann Arbor, Michigan, 2019a. Association for Computational Linguistics. doi:10.18653/v1/W19-1419. URL https://aclanthology.org/W19-1419.

Tommi Jauhiainen et al. Automatic language identification in texts: A survey. *Journal of Artificial Intelligence Research*, 65:675–782, 2019b.

Lucy Linder et al. Automatic creation of text corpora for low-resource languages from the Internet: The case of Swiss German. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2706–2711, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.329.

Christopher D. Manning et al. *Introduction to Information Retrieval*. Cambridge University Press, 2008. doi:10.1017/CBO9780511809071.

Fabian Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rico Sennrich et al. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Beat Siebenhaar and Walter Voegeli. Mundart und hochdeutsch im vergleich, 1997. URL https://home.uni-leipzig.de/siebenh/pdf/Siebenhaar_Voegeli_iPr.pdf. Accessed: 18-3-2023.

Karen Sparck Jones. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR, 1988. ISBN 0947568212.

Jannis Vamvas et al. SwissBERT: The multilingual language model for switzerland. 2023. URL https://arxiv.org/abs/2303.13310.

C Van Rijsbergen. Information retrieval (book 2nd ed), 1979.

Nianheng Wu et al. Language discrimination and transfer learning for similar languages: Experiments with feature combinations and adaptation. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 54–63. Association for Computational Linguistics, 2019. doi:10.18653/v1/W19-1406. URL https://aclanthology.org/W19-1406.

Marcos Zampieri et al. A report on the third VarDial evaluation campaign. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–16, Ann Arbor, Michigan, June 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-1401. URL https://aclanthology.org/W19-1401.

A Additional Figures

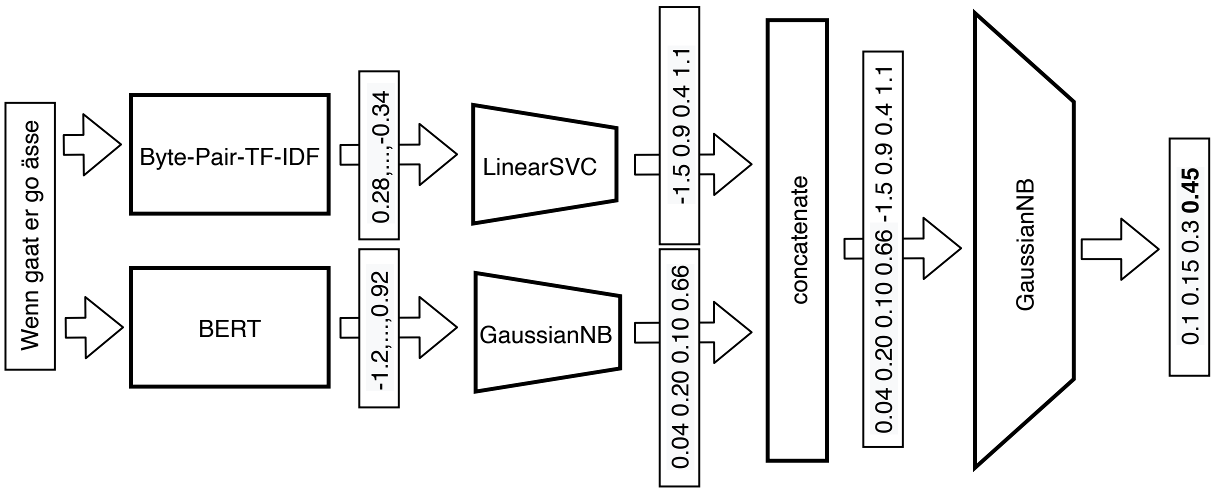


Figure 4: Shows how HelvetiaDial processes a sentence. The dialect with the highest probability is predicted.

GDI rank	Team	F1 (macro)
1	tearsofjoy	0.7593
2	SUKI	0.7541
3	Twist Bytes	0.7455
4	BAM	0.6255
	HelvetiaDial	**0.6169**
	Baseline	0.6078
5	dkosmajac	0.5616
6	ghpaetzold	0.5575

Figure 5: Test F1-score of HelvetiaDial inserted into rank table of GDI from Zampieri et al. (2019)

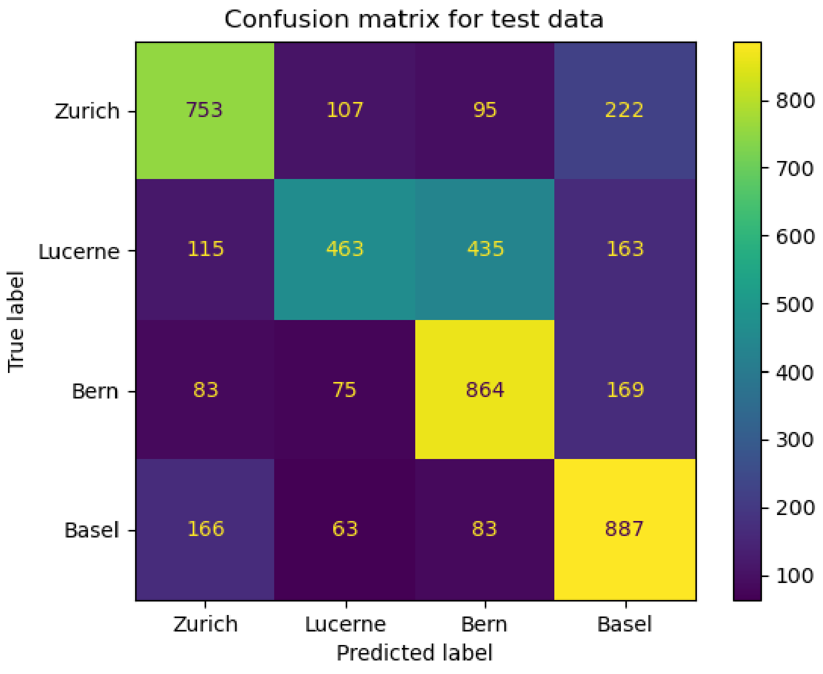


Figure 6: Confusion matrix between HelvetiaDial predictions and test data ground truth.