IDIAP RESEARCH REPORT

IDIAP SUBMISSION TO SWISS-GERMAN LANGUAGE DETECTION SHARED TASK

Shantipriya Parida[a] Esaú VILLATORO-TELLO[b]
Sajit Kumar Petr Motlicek Qingran Zhan

Idiap-RR-11-2020

JUNE 2020

[a]Idiap Research Institute
[b]Idiap

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

Idiap Submission to Swiss-German Language Detection Shared Task

Shantipriya Parida[1], **Esaú Villatoro-Tello**[2,1], **Sajit Kumar**[3],
Petr Motlicek[1] **and Qingran Zhan**[1]

[1]Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland.
firstname.lastname@idiap.ch
[2]Universidad Autónoma Metropolitana, Unidad Cuajimalpa, Mexico City, Mexico.
evillatoro@correo.cua.uam.mx
[3]Centre of Excellence in AI, Indian Institute of Technology, Kharagpur, West Bengal, India.
kumar.sajit.sk@gmail.com

Abstract

Language detection is a key part of the NLP pipeline for text processing. The task of automatically detecting languages belonging to disjoint groups is relatively easy. It is considerably challenging to detect languages that have similar origins or dialects. This paper describes Idiap's submission to the 2020 Germeval evaluation campaign[1] on Swiss-German language detection. In this work, we have given high dimensional features generated from the text data as input to a supervised autoencoder for detecting languages with dialect variances. Bayesian optimizer was used to fine-tune the hyper-parameters of the supervised autoencoder. To the best of our knowledge, we are first to apply supervised autoencoder for the language detection task.

1 Introduction

The increased usage of smartphones, social media, and the internet has led to rapid growth in the generation of short linguistic texts. Thus, identification of language is a key component in building various NLP resources (Kocmi and Bojar, 2017). Language detection is the task of determining the language for the given text. Although it has progressed substantially, still few challenges exist: (1) distinguishing among similar languages, (2) detection of languages when multiple language contents exist within a single document, and (3) language identification in very short texts (Balazevic et al., 2016; Lui et al., 2014; Williams and Dagli, 2017).

It is a difficult task to discriminate between very close languages or dialects (for example, German dialect identification, Indo-Aryan language identification (Jauhiainen et al., 2019a)). Although dialect identification is commonly based on the distributions of letters or letter n-grams, it may not be possible to distinguish related dialects with very similar phoneme and grapheme inventories for some languages (Scherrer and Rambow, 2010).

Many authors proposed traditional machine learning approaches for language detection like Naive Bayes, SVM, word and character n-grams, graph-based n-grams, prediction partial matching (PPM), linear interpolation with post-independent weight optimization and majority voting for combining multiple classifiers, etc. (Jauhiainen et al., 2019b).

More recently, deep learning techniques have shown substantial performance in many NLP tasks including language detection (Oro et al., 2018). In the context of deep learning techniques, many papers have demonstrated the capability of semi-supervised autoencoders solving different tasks, indicating that the use of autoencoders allows learning a representation when trained with unlabeled data. (Ranzato and Szummer, 2008; Rasmus et al., 2015). However, as per our literature survey, none of the recent research has applied autoencoder for the language detection task. In this paper, we propose a supervised configuration of the autoencoders, which utilizes labels for learning the representation. To the best of our knowledge, this is the first time this technology is evaluated in the context of the language detection task.

1.1 Supervised Autoencoder

An autoencoder (AE) is a neural network that learns a representation (encoding) of input data and then learns to reconstruct the original input from the learned representation. The autoencoder is mainly

[1]https://sites.google.com/view/gswid2020

used for dimensionality reduction or feature extraction (Zhu and Zhang, 2019). Normally, it is used in an unsupervised learning fashion, meaning that we leverage the neural network for the task of representation learning. By learning to reconstruct the input, the AE extracts underlying abstract attributes that facilitate accurate prediction of the input.

Thus, a supervised autoencoder (SAE) is an autoencoder with the addition of a supervised loss on the representation layer. For the case of a single hidden layer, a supervised loss is added to the output layer and for a deeper autoencoder, the innermost (smallest) layer would have a supervised loss added to the bottleneck layer that is usually transferred to the supervised layer after training the autoencoder.

In supervised learning, the goal is to learn a function for a vector of inputs $\mathbf{x} \in \mathbb{R}^d$ to predict a vector of targets $\mathbf{y} \in \mathbb{R}^m$. Consider SAE with a single hidden layer of size k, and the weights for the first layer are $\mathbf{F} \in \mathbb{R}^{k \times d}$. The function is trained on a finite batch of independent and identically distributed (i.i.d.) data, $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_t, \mathbf{y}_t)$, with the goal of a more accurate prediction on new samples generated from the same distribution. The weight for the output layer consists of weights $\mathbf{W}_p \in \mathbb{R}^{m \times k}$ to predict $\mathbf{y}$ and $\mathbf{W}_r \in \mathbb{R}^{d \times k}$ to reconstruct $\mathbf{x}$. Let L_p be the supervised loss and L_r be the loss for the reconstruction error. In the case of regression, both losses might be represented by a squared error, resulting in the objective:

$$\frac{1}{t}\sum_{i=1}^{t}\Big[L_p(\mathbf{W}_p\mathbf{F}\mathbf{x}_i, \mathbf{y}_i) + L_r(\mathbf{W}_r\mathbf{F}\mathbf{x}_i, \mathbf{x}_i)\Big] = \frac{1}{2t}\sum_{i=1}^{t}\Big[||\mathbf{W}_p\mathbf{F}\mathbf{x}_i - \mathbf{y}_i||_2^2 + ||\mathbf{W}_r\mathbf{F}\mathbf{x}_i - \mathbf{x}_i||_2^2\Big] \quad (1)$$

The addition of supervised loss to the autoencoder loss function acts as regularizer and results (as shown in equation 1) in the learning of the better representation for the desired task (Le et al., 2018).

1.2 Bayesian Optimizer

In the case of SAE, there are many hyperparameters related to (a) Model construction and (b) Optimization. Hence, SAE training without any hyperparameter tuning usually results in poor performance due to the dependencies that may result in simultaneous over/under-fitting.

Global optimization is considered to be a challenging problem of finding the globally best solution of (possibly nonlinear) models, in the (possible or known) presence of multiple local optima. Bayesian optimization (BO) is shown to outperform other state-of-the-art global optimization algorithms on several challenging optimization benchmark functions (Snoek et al., 2012; Bergstra and Bengio, 2012). BO provides a principled technique based on Bayes theorem to direct a search for a global optimization problem that is efficient and effective. It works by building a probabilistic model of the objective function, called the surrogate function, that is then searched efficiently with an acquisition function before candidate samples are chosen for evaluation on the real objective function. It tries to solve the minimization problem:

$$X^* = \arg\min_{x \in \chi} f(x), \quad (2)$$

where we consider χ to be a compact subset of $\mathbb{R}^k$ (Snoek et al., 2015).

Thus, we employed BO for hyperparameter optimization where the objective is to find the hyperparameters of a given machine learning algorithm, for this, we preserved the best performance as measured on a validation set.

2 Proposed Method

The architecture of the proposed model is shown in Figure 1. We used character n-grams as features from the input text. In comparison to word n-grams, which only capture the identity of a word and its possible neighbors, character n-grams are additionally capable of providing an excellent trade-off between sparseness and word's identity, while at the same time they combine different types of information: punctuation, morphological makeup of a word, lexicon and even context (Wei et al., 2009; Kulmizev et al., 2017; Sánchez-Vega et al., 2019). The extracted n-gram features are input to the deep SAE as shown in the Figure 1. The deep SAE contains multiple hidden layers. We used the BO for selecting the optimal parameters.

3 Experimental Setup and Datasets

The training dataset was provided by the organizers of the shared task. The training[2] dataset consists of 2,000 tweets in the Swiss-German language. The

[2] Although 2K Twitter ids were provided, we were not able to retrieve them all, resulting in 1976 training instances.

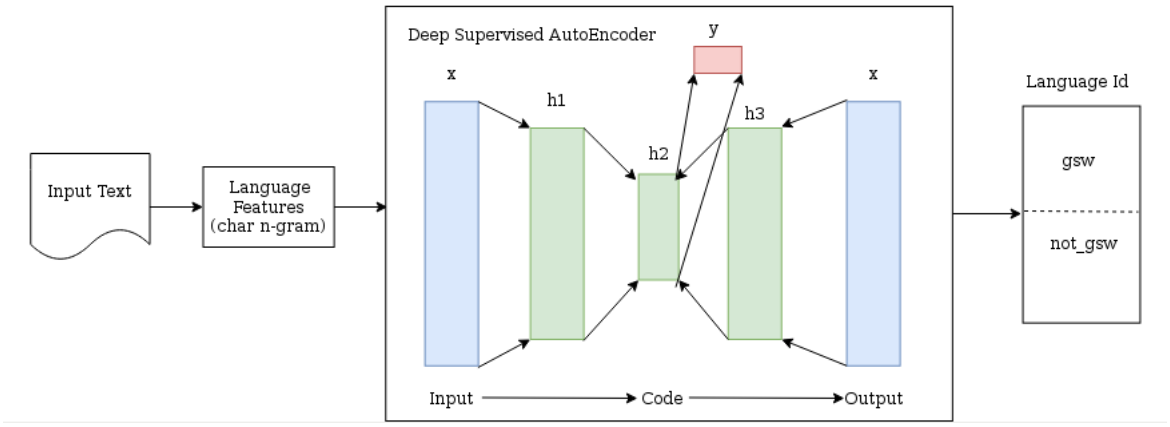


Figure 1: Proposed model architecture. The extracted features of the text are fed to the supervised autoencoder. The targets "y" are included. The classification output are the language ids for the classified languages.

participants were allowed to use any additional resources as training datasets. As part of the additional resources recommended by the organizers, the following Swiss-German datasets were suggested: NOAH [3] (Hollenstein and Aepli, 2015), and SwissCrawl [4](Linder et al., 2019); which we used in our experiments.

The test data released by the organizers consists of 5,374 Tweets (mix of different languages) to be classified as Swiss-German versus not Swiss-German.

The training dataset provided by the organizer did not have any non-Swiss-German text. In addition to the recommended Swiss-German datasets, we have used other non-Swiss-German datasets (DSL [5] (Tan et al., 2014a), and Ling10 [6]) for training our models.

- *DSL Dataset:* The data obtained from the "Discriminating between Similar Language (DSL) Shared Task 2015" contains 13 different languages as shown in Table 1. The DSL corpus collection have different versions based on different language group which provides datasets for researchers to test their systems (Tan et al., 2014a). We selected DSLCC version 2.0 [7] in our experiments (Tan et al., 2014b).
- *Ling10 Dataset :* The Ling10 dataset contains 190,000 sentences categorized into 10 languages (English, French, Portuguese, Chinese Mandarin, Russian, Hebrew, Polish, Japanese, Italian, Dutch) mainly used for language detection and benchmarking NLP algorithms. We considered "Ling10-trainlarge" (one of the three variants of Ling10 dataset) in our experiment.

Group Name	Language	Id
South Eastern Slavic	Bulgarian Macedonian	bg mk
South Western Slavic	Bosnian Croatian Serbian	bs hr sr
West-Slavic	Czech Slovak	cz sk
Ibero-Romance(Spanish)	Peninsular Spain Argentinian Spanish	es-ES es-AR
Ibero-Romance(Portuguese)	Brazilian Portuguese European Portuguese	pt-BR pt-PT
Astronesian	Indonesian Malay	id my

Table 1: DSL Language Group. Similar languages with their language code.

As the task is a binary classification of Swiss-German versus not Swiss-German, we have split all our collection of datasets including the training set provided by the organizers into two categories as follows:

- Swiss-German (NOAH, SwissCrawl, Swiss-German Training Tweets).
- not Swiss-German (DSL, Ling10).

Accordingly, we labeled the target class of all the Swiss-German text as "gsw" (Swiss-German) and labeled the target class of all other language

[3] https://noe-eva.github.io/NOAH-Corpus/
[4] https://icosys.ch/swisscrawl
[5] http://ttg.uni-saarland.de/resources/DSLCC/
[6] https://github.com/johnolafenwa/Ling10
[7] https://github.com/Simdiva/DSL-Task/tree/master/data/DSLCC-v2.0

text as "not_gsw").

We prepared three settings (S1, S2, and S3) combining the above datasets in different proportions of Swiss-German versus not Swiss-German languages for training the model. The statistics of the datasets for the settings are shown in Table 2.

We mixed the datasets of Swiss-German and other languages and split them into different ratios for training and development as per the settings. In each setting, the training and development set is different based on the selection of the number of sentences from each dataset. We used the test set provided by the shared task organizers. As the test set includes twitter text during preprocessing, we removed emojis and other unnecessary symbols.

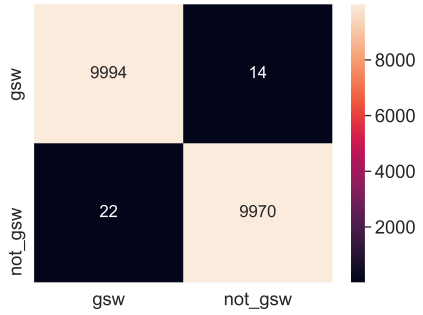
The range of values for the hyperparameters search space is shown in Table 3. During training, BO chooses the best hyperparameters from this range. The overall configuration of the SAE model is shown in Table 4.

4 Results and Discussion

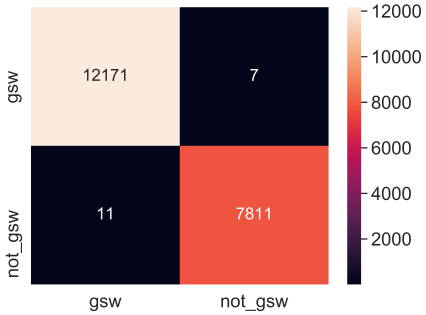
We evaluated the development set performance and the test set evaluation performed by the shared task organizers. The development set performance is given in section Section 4.1 and the test set performance in Section 4.2.

Our evaluation includes calculating classification accuracy based on the predicted label compared with the actual label. The organizers calculated *precision*, average *precision*, *recall*, and *F1* score for each of the submissions. As known, *precision* is the ratio of correctly predicted positive observations to the total predicted positive observations; *recall* (or sensitivity) is the ratio of correctly predicted positive observations to all observations in actual positive class, and the *F1* score is the weighted average of *precision* and *recall*.

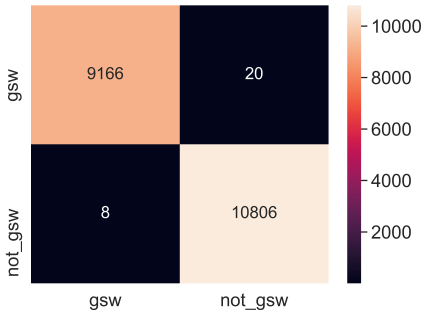
Organizers also generated the Receiver Operating Characteristic curve (ROC), Area Under the ROC Curve (AUC), and Precision-Recall (PR) curves. The AUC - ROC curve is a performance measurement at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It indicates how much a trained model is capable of distinguishing between classes, thus, the higher the AUC, the better the model performance. Finally, PR curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds; hence, a good model is represented by a curve that bows towards (1,1).



Confusion matrix for setting S1 on dev set.



Confusion matrix for setting S2 on dev set.



Confusion matrix for setting S3 on dev set.

Figure 2: Confusion matrix on the development (dev) set for the setting S1, S2, and S3. The confusion matrix shows the correct and incorrect predictions with count values broken down by each class i.e. "gsw" (Swiss-German) or "not_gsw" (not Swiss-German).

4.1 Development Set

The SAE model performance for the three settings (S1, S2, and S3) on the development set is shown in Table 5. The confusion matrix for all the settings on the development set is shown in Figure 2. The confusion matrix shows the correct and incorrect predictions with count values broken down by each class i.e. "gsw" (Swiss-German) or "not_gsw" (not

Setting	Datasets and Language	Distribution	Distribution (Overall)	Training	Dev	Test
S1	NOAH (*Swiss-German*) SwissCrawl (*Swiss-German*) SwissTextTrain (*Swiss-German*) DSL (*not Swiss-German*) Ling10 (*not Swiss-German*)	7,327 (8%) 40,697 (40%) 1,976 (2 %) 25,000 (25 %) 25,000 (25 %)	50% Swiss-German 50% not Swiss-German	80,000	20,000	5,374
S2	NOAH (*Swiss-German*) SwissCrawl (*Swiss-German*) SwissTextTrain (*Swiss-German*) DSL (*not Swiss-German*) Ling10 (*not Swiss-German*)	7,327 (5%) 81,841 (55 %) 1,976 (1 %) 25,000 (17 %) 33,856 (22 %)	61% Swiss-German 39% not Swiss German	130,000	20,000	5,374
S3	NOAH (*Swiss-German*) SwissCrawl (*Swiss-German*) SwissTextTrain (*Swiss-German*) DSL (*not Swiss-German*) Ling10 (*not Swiss-German*)	7,327 (4 %) 81,841 (41 %) 1,976 (1 %) 50,000 (25 %) 58,856 (29 %)	46% Swiss-German 54% not Swiss-German	180,000	20,000	5,374

Table 2: Dataset Statistics. The training-development-test set distribution for each of setting (S1, S2 and S3). The distribution is based on the number of sentences selected from the datasets.

Hyper Parameter	Range
number of layer	1-5
learning rate	$10^{-5} - 10^{-2}$
weight decay	$10^{-6} - 10^{-3}$
activation functions	'relu', 'sigma'

Table 3: Search space hyper parameter range.

Parameter	Value
char `n_gram` range	1-3
number of target	2
embedding dimension	300
supervision	'clf' (classification)
converge threshold	0.00001
number of epochs	500

Table 4: SAE model configuration used for training.

Swiss-German).

		Accuracy (%)
Model	**Setting**	**Development Set**
SAE (char-3gram)	S1	100
SAE (char-3gram)	S2	100
SAE (char-3gram)	S3	100

Table 5: Swiss-German language detection performance (classification accuracy) of the proposed model on the development set based on the setting S1, S2, and S3.

4.2 Test Set

The overall result announced by the organizers on test set is shown in the Table 6 and in the Figure 3. Our submission labeled as *"IDIAP"*, obtained the results *0.777*, *0.998*, and *0.872* for precision (prec), recall (rec), and F1 score respectively for the setting S3 as shown in Table 6. The detailed performance of each of our setting is shown in Table 7.

	Precision	Recall	F1
IDIAP	0.775	0.998	0.872
jj-cl-uzh	0.945	0.993	0.968
Mohammadreza Banaei	0.984	0.979	**0.982**

Table 6: Shared task result announced by the organizers displaying participant team and their model performance (Precision, Recall, and F1).

Setting	Prec (gsw)	Rec (gsw)	F1 (gsw)	Avg. Prec	AUROC
S1	0.649	0.997	0.786	0.871	0.924
S2	0.673	0.997	0.804	0.911	0.946
S3	**0.775**	**0.998**	**0.872**	**0.965**	**0.975**

Table 7: Performance of setting S1, S2, and S3.

Based on our initial analysis, we presume that the low performance of the SAE on the test set is due to the very few samples of twitter data available in the training data.

5 Conclusion

In this paper, we have shown the pertinence of SAE with Bayesian optimizer for the language detection task. Obtained results are encouraging, and SAE was found effective for discriminate between very close languages or dialects. The proposed model can be extended by creating a host of features such as character n-gram, word n-gram, word counts, etc and then passing it through autoencoder to choose the best features. In future work, we plan to (i) verify our model (SAE with BO) with other language detection datasets, and (ii) include more short texts, particularly Twitter data, in the training set and

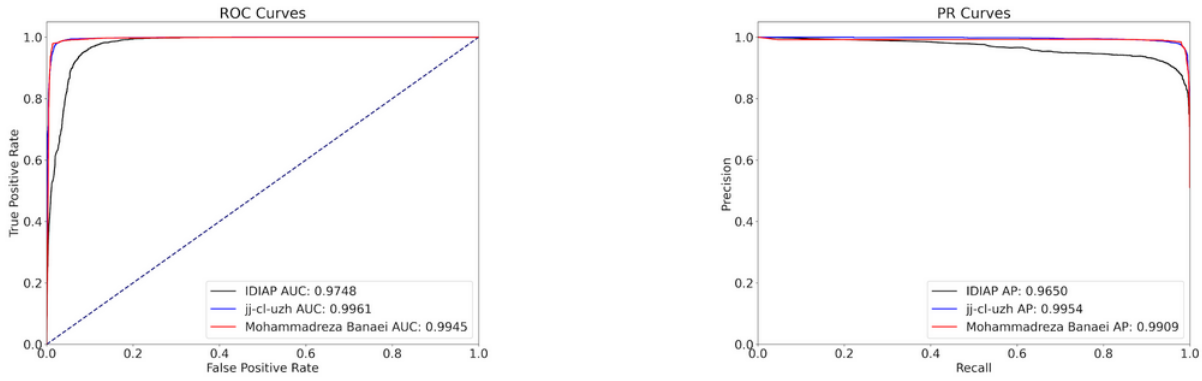


Figure 3: Official results announced by the organizers displaying team's performance (ROC, PR curves).

verify the performance of our model under a more balanced data type scenario.

Acknowledgments

The work was supported by an innovation project (under an InnoSuisse grant) oriented to improve the automatic speech recognition and natural language understanding technologies for German. Title: "SM2: Extracting Semantic Meaning from Spoken Material" funding application no. 29814.1 IP-ICT and EU H2020 project "Real-time network, text, and speaker analytics for combating organized crime" (ROXANNE), grant agreement: 833635. The second author, Esaú Villatoro-Tello is supported partially by Idiap, UAM-C Mexico, and SNI-CONACyT Mexico during the elaboration of this work.

References

Ivana Balazevic, Mikio Braun, and Klaus-Robert Müller. 2016. Language detection for short text messages in social media. *arXiv preprint arXiv:1608.08515*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305.

Nora Hollenstein and Noëmi Aepli. 2015. A resource for natural language processing of swiss german dialects.

Tommi Jauhiainen, Krister Lindén, and Heidi Jauhiainen. 2019a. Language model adaptation for language and dialect identification of text. *Natural Language Engineering*, 25(5):561–583.

Tommi Sakari Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2019b. Automatic language identification in texts: A survey. *Journal of Artificial Intelligence Research*, 65:675–782.

Tom Kocmi and Ondřej Bojar. 2017. Lanidenn: Multilingual language identification on character window. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 927–936.

Artur Kulmizev, Bo Blankers, Johannes Bjerva, Malvina Nissim, Gertjan van Noord, Barbara Plank, and Martijn Wieling. 2017. The power of character n-grams in native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 382–389.

Lei Le, Andrew Patterson, and Martha White. 2018. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In *Advances in Neural Information Processing Systems*, pages 107–117.

Lucy Linder, Michael Jungo, Jean Hennebert, Claudiu Musat, and Andreas Fischer. 2019. Automatic creation of text corpora for low-resource languages from the internet: The case of swiss german.

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

Ermelinda Oro, Massimo Ruffolo, and Mostafa Sheikhalishahi. 2018. Language identification of similar languages using recurrent neural networks. In *ICAART*.

Marc'Aurelio Ranzato and Martin Szummer. 2008. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. 2015. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554.

Fernando Sánchez-Vega, Esaú Villatoro-Tello, Manuel Montes-y Gómez, Paolo Rosso, Efstathios Stamatatos, and Luis Villaseñor-Pineda. 2019. Paraphrase plagiarism identification with character-level features. *Pattern Analysis and Applications*, 22(2):669–681.

Yves Scherrer and Owen Rambow. 2010. Natural language processing for the swiss german dialect area. In *Semantic Approaches in Natural Language Processing-Proceedings of the Conference on Natural Language Processing 2010 (KONVENS)*, pages 93–102. Universaar.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. 2015. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180.

Liling Tan, Marcos Zampieri, Nikola Ljubešic, and Jörg Tiedemann. 2014a. Merging comparable data sources for the discrimination of similar languages: The dsl corpus collection. In *Proceedings of the 7th Workshop on Building and Using Comparable Corpora (BUCC)*, pages 11–15.

Liling Tan, Marcos Zampieri, Nikola Ljubešic, and Jörg Tiedemann. 2014b. Merging comparable data sources for the discrimination of similar languages: The dsl corpus collection. In *Proceedings of the 7th Workshop on Building and Using Comparable Corpora (BUCC)*, pages 11–15, Reykjavik, Iceland.

Zhihua Wei, Duoqian Miao, Jean-Hugues Chauchat, Rui Zhao, and Wen Li. 2009. N-grams based feature selection and text representation for chinese text classification. *International Journal of Computational Intelligence Systems*, 2(4):365–374.

Jennifer Williams and Charlie Dagli. 2017. Twitter language identification of similar languages and dialects without ground truth. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 73–83.

Qiuyu Zhu and Ruixin Zhang. 2019. A classification supervised auto-encoder based on predefined evenly-distributed class centroids. *arXiv preprint arXiv:1902.00220*.