BAM: A combination of deep and shallow models for German Dialect Identification.

Andrei M. Butnaru
Department of Computer Science, University of Bucharest
14 Academiei, Bucharest, Romania
butnaruandreimadalin@gmail.com

Abstract

In this paper, we present a machine learning approach for the German Dialect Identification (GDI) Closed Shared Task of the DSL 2019 Challenge. The proposed approach combines deep and shallow models, by applying a voting scheme on the outputs resulted from a Character-level Convolutional Neural Networks (Char-CNN), a Long Short-Term Memory (LSTM) network, and a model based on String Kernels. The first model used is the Char-CNN model that merges multiple convolutions computed with kernels of different sizes. The second model is the LSTM network which applies a global max pooling over the returned sequences over time. Both models pass the activation maps to two fully-connected layers. The final model is based on String Kernels, computed on character p-grams extracted from speech transcripts. The model combines two blended kernel functions, one is the presence bits kernel, and the other is the intersection kernel. The empirical results obtained in the shared task prove that the approach can achieve good results. The system proposed in this paper obtained the fourth place with a macro-F_1 score of 62.55%.

1 Introduction

Being at its third edition, the 2019 VarDial Evaluation Campaign (Zampieri et al., 2019) includes two shared tasks on dialect identification which proves that researchers are still interested in this challenging NLP task. For example, in the 2018 GDI Shared Task (Zampieri et al., 2018), a system (Jauhiainen et al., 2018) that uses a series of language models based on character n-grams achieves state-of-the-art with a macro-F_1 score near 69%, in a 4-way classification setting. For the 2019 GDI Shared Task, the organizers have included audio features together with speech transcripts, and also provided a word-level normalization for each transcript. For solving this task, we propose a combination of deep and shallow models, by applying a voting scheme on the outputs resulted from a Character-level Convolutional Neural Networks (Char-CNN), a Long Short-Term Memory (LSTM) network, and a model based on String Kernels. In the 2019 GDI Shared Task, the participants had to discriminate between four German dialects, in a 4-way classification setting. A number of 6 participants have submitted their results, and the model proposed in this paper obtained 4^{th} place with an accuracy of 62.95%, and macro-F_1 score of 62.55%.

The best scoring system that we submitted for the GDI Shared Task is an ensemble that combines both deep and shallow models. The system uses features from two deep models, Char-CNNs and LSTMs, and also from a shallow model that combines several kernels using multiple kernel learning. The Char-CNN model merges convolutions computed with kernels of different sizes to learn a first group of features. The LSTM network learns the second group of features by applying a global max pooling over the returned sequences over time. For the String Kernel model, we combined two kernel functions. The first kernel used is the p-grams presence bits kernel[1] which takes into account only the presence of p-grams instead of their frequencies. The second kernel is the histogram intersection kernel[2], which was first used in a text mining task by (Ionescu et al., 2014). This kernel functions proved useful in previous dialect identification shared tasks (Ionescu and Popescu, 2016; Ionescu and Butnaru, 2017; Butnaru and Ionescu, 2018).

There are two steps in the learning process. In

[1]The p-grams presence bits kernel was computed using the code available at http://string-kernels.herokuapp.com.

[2]The intersection string kernel was computed using the code available at http://string-kernels.herokuapp.com.

Proceedings of VarDial, pages 128–137
Minneapolis, MN, June 7, 2019

the first step, the deep models are trained individually using the Adam optimization algorithm (Kingma and Ba, 2015). In the second step, the string kernel model is learned by applying Kernel Ridge Regression (KRR) (Shawe-Taylor and Cristianini, 2004). Finally, a voting schema is applied to obtain the final class for a test sample. Before deciding the final system, we tuned each model for the task. First of all, we tuned the string kernels model by trying out p-grams of various lengths, including blended variants of string kernels as well. Besides blended variants, we evaluated individual kernels, and also various kernel combinations. Second of all, we tuned the Char-CNN model, by trying out various convolution lengths, number of filters and depths. Finally, we tuned the LSTM model by seeking the best number of output units.

The paper is organized as follows. Work related to German dialect identification, models based on Character-Level Convolutional Neural Networks, Long Short-Term Memory Networks, and methods based on string kernels is presented in Section 2. Section 3 presents Char-CNNs, LSTMs and the string kernel models used in this approach. In this section, we also present the ensemble model. Details about the German dialect identification experiments are provided in Section 4. Finally, we draw the conclusion in Section 5.

2 Related Work

2.1 German Dialect Identificaion

German dialect identification is not a widely researched task, but we can observe an increased interest in it within recent years. In 2010, a system for written dialect identification was proposed (Scherrer and Rambow, 2010) , and it was based on an automatically generated Swiss German lexicon that maps word forms with their geographical extensions. During test time, they split a sentence into words and look up their geographical extension in the lexicon. Another method (Hollenstein and Aepli, 2015) was proposed for solving the Swiss German dialect identification task based on trigrams. For each dialect, a language model was trained, and each test sentence was scored against every model. The predicted dialect is chosen based on the lowest perplexity. In 2016, a corpus (Samardžić et al., 2016) that can be used for GDI was presented, which was later used to evaluate the participants in the GDI Shared Task of the DSL 2017 Challenge. One of the participants in the previously mentioned shared task, defined a system (Ionescu and Butnaru, 2017) that is based on multiple string kernels. The team used a Kernel Ridge Regression classifier trained on a kernel combination of a blended presence bits kernel based on $3-6$-grams, a blended intersection kernel based on $3-6$-grams, and a kernel based on LRD with $3-5$-grams. The winning team of the GDI Shared Task of the VarDial 2018 Workshop defined a system (Jauhiainen et al., 2018) that is based on language models defined on character 4-grams, and having on top the HeLI method.

2.2 String Kernels

In the past years, we can find that techniques that approach text at the character level proved remarkable performance levels in various text analysis tasks (Lodhi et al., 2002; Sanderson and Guenter, 2006; Kate and Mooney, 2006; Escalante et al., 2011; Popescu and Grozea, 2012; Popescu and Ionescu, 2013; Ionescu et al., 2014, 2016; Giménez-Pérez et al., 2017; Popescu et al., 2017; Cozma et al., 2018; Ionescu and Butnaru, 2018a). String kernels are a natural way of using character level information to generate features that are helpful in solving various areas of tasks. They are a particular case of the more general convolution kernels (Haussler, 1999). Since the beginning of the 21st century, researchers applied string kernels on document categorization (Lodhi et al., 2002), obtaining excellent results. String kernels were also successfully used in authorship identification (Sanderson and Guenter, 2006; Popescu and Grozea, 2012). For example, the first ranked team of the PAN 2012 Traditional Authorship Attribution task, used a system (Popescu and Grozea, 2012) based on string kernels. In recent years, various blended string kernels reached state-of-the-art accuracy rates for native language identification (Ionescu et al., 2016; Ionescu and Popescu, 2017), Arabic dialect identification (Ionescu and Popescu, 2016; Ionescu and Butnaru, 2017; Butnaru and Ionescu, 2018), polarity classification (Giménez-Pérez et al., 2017; Popescu et al., 2017), automatic essay scoring (Cozma et al., 2018), and cross-domain text classification (Ionescu and Butnaru, 2018a,b).

2.3 Character-Level CNN Networks

Convolutional networks (LeCun et al., 1998) have proven to be very efficient in solving various

computer vision tasks (Krizhevsky et al., 2012; Szegedy et al., 2015; Ren et al., 2015). Therefore, many researchers decided to apply CNNs in their primary area of interest. For example, in the NLP domain, Convolutional Neural Networks (LeCun et al., 1998; Krizhevsky et al., 2012) were successfully applied on several NLP tasks such as part-of-speech tagging (Santos and Zadrozny, 2014), text categorization (Kim, 2014; Zhang et al., 2015; Johnson and Zhang, 2015), dialect identification (Belinkov and Glass, 2016; Ali, 2018), machine translation (Gehring et al., 2017) and language modeling (Kim et al., 2016; Dauphin et al., 2017). Word embeddings (Mikolov et al., 2013; Pennington et al., 2014) had a significant impact on NLP due to their ability to learn semantic and syntactic latent features. Because of this, researchers developed many CNN-based methods that rely on word embeddings. Trying to eliminate the pre-trained word embeddings from the pipeline, some researchers have tried to build end-to-end models using characters as input, in order to solve text classification (Zhang et al., 2015; Belinkov and Glass, 2016), language modeling (Kim et al., 2016) or dialect identification (Butnaru and Ionescu, 2019) tasks. Using characters as features can help the model learn unusual character sequences such as misspellings or take advantage of unseen words during test time. Working at the character-level can prove useful in solving the dialect identification task, since some state-of-the-art dialect identification methods (Ionescu and Butnaru, 2017; Butnaru and Ionescu, 2018) use character n-grams as features.

2.4 Long-Short Term Memory Networks

Recurrent Neural Networks (RNNs) (Elman, 1990) have the ability to process fixed length sequences and learn short-term dependencies between items from the sequence (Lin et al., 1996). A limitation of the RNN model is that it cannot learn long-distance correlations between items within a sequence (Hochreiter and Schmidhuber, 1997; Hochreiter et al., 2001). Long Short-Term Memory (LSTMs) (Hochreiter and Schmidhuber, 1997) have been proposed as a solution for the RNNs issue, introducing a memory cell inside the network. LSTMs have become more popular after being successfully applied in statistical machine translation (Sutskever et al., 2014). Besides this, researchers employed LSTMs in various areas, from speech recognition (Graves et al., 2013a,b; Amodei et al., 2016), to language modelling (Kim et al., 2016), and text classification (Zhang et al., 2015).

2.5 Ensemble Learning

Ensemble learning combines a number of previously trained classifiers to classify new data samples by applying a voting schema on their predictions. Ensemble methods have been successfully employed in various machine learning tasks, including feature selection (Saeys et al., 2008), sentiment analysis (Xia et al., 2011; Wang et al., 2014), complex word identification (Malmasi et al., 2016), and dialect identification (Malmasi and Dras, 2015; Malmasi and Zampieri, 2017).

3 Method

The model presented in this paper combines the results obtained from three different learning algorithms: Kernel Ridge Regression over String Kernels, Character-level Convolutional Neural Networks, and a Long Short-Term Memory Network. The intuition to use three different learning algorithms comes from the idea that each trained model can discover from the same input different discriminant features, thus combining them will increase the accuracy of each and any model. String Kernels uses a function to compute a similarity matrix that carries the correlation between all pairs of samples. Based on the similarity between the samples (that is held in the similarity matrix) the KRR can learn a function that discriminates between them. Character-level Convolutional Neural Networks can learn to discriminate between samples by discovering patterns at character-level. The intuition in using this method comes from the idea that the same words in different dialects can have small character differences. Besides small character differences, there can be connections between words within a text. Having this in mind we propose to employ an LSTM netowrk to learn patterns between words within texts from the same dialect.

3.1 String Kernels

Kernel functions (Shawe-Taylor and Cristianini, 2004) have the ability to capture the concept of similarity between objects within a specific domain. The kernel function gives kernel methods

the power to naturally handle input data that is not in the form of numerical vectors, for example strings. There are many kernel functions that can be applied on strings, with significant impact in domains such as computational biology and computational linguistics. String kernels embed the texts in a very large feature space, given by all the substrings of length p, and leave the job of selecting important (discriminative) features for the specific classification task to the learning algorithm, which assigns higher weights to the important features (character p-grams). Perhaps one of the most natural ways to measure the similarity of two strings is to count how many substrings of length p the two strings have in common. This gives rise to the p-spectrum kernel. Formally, for two strings over an alphabet Σ, $s, t \in \Sigma^*$, the p-spectrum kernel is defined as:

$$k_p(s,t) = \sum_{v \in \Sigma^p} \text{num}_v(s) \cdot \text{num}_v(t),$$

where $\text{num}_v(s)$ is the number of occurrences of string v as a substring in s. The feature map defined by this kernel associates to each string a vector of dimension $|\Sigma|^p$ containing the histogram of frequencies of all its substrings of length p (p-grams). A variant of this kernel can be obtained if the embedding feature map is modified to associate to each string a vector of dimension $|\Sigma|^p$ containing the presence bits (instead of frequencies) of all its substrings of length p. Thus, the character p-grams presence bits kernel is obtained:

$$k_p^{0/1}(s,t) = \sum_{v \in \Sigma^p} \text{in}_v(s) \cdot \text{in}_v(t),$$

where $\text{in}_v(s)$ is 1 if string v occurs as a substring in s, and 0 otherwise.

In computer vision, the (histogram) intersection kernel has successfully been used for object class recognition from images (Maji et al., 2008; Vedaldi and Zisserman, 2010). Ionescu et al. (2014) have used the intersection kernel as a kernel for strings, in the context of native language identification. The intersection string kernel is defined as follows:

$$k_p^{\cap}(s,t) = \sum_{v \in \Sigma^p} \min\{\text{num}_v(s), \text{num}_v(t)\}.$$

For the p-spectrum kernel, the frequency of a p-gram has a very significant contribution to the kernel, since it considers the product of such frequencies. On the other hand, the frequency of a p-gram is completely disregarded in the p-grams presence bits kernel. The intersection kernel lies somewhere in the middle between the p-grams presence bits kernel and the p-spectrum kernel, in the sense that the frequency of a p-gram has a moderate contribution to the intersection kernel. In other words, the intersection kernel assigns a high score to a p-gram only if it has a high frequency in both strings, since it considers the minimum of the two frequencies. The p-spectrum kernel assigns a high score even when the p-gram has a high frequency in only one of the two strings. Thus, the intersection kernel captures something more about the correlation between the p-gram frequencies in the two strings. Based on these comments, in the experiments we use only the p-grams presence bits kernel and the intersection string kernel.

Data normalization helps to improve machine learning performance for various applications. Since the value range of raw data can have large variations, classifier objective functions will not work properly without normalization. After normalization, each feature has an approximately equal contribution to the similarity between two samples. To obtain a normalized kernel matrix of pairwise similarities between samples, each component is divided by the square root of the product of the two corresponding diagonal components:

$$\hat{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii} \cdot K_{jj}}}.$$

To ensure a fair comparison among strings of different lengths, normalized versions of the p-grams presence bits kernel and the intersection kernel were used in the experiments. Taking into account p-grams of different lengths and summing up the corresponding kernels, new kernels, termed *blended spectrum kernels*, can be obtained. Various blended spectrum kernels were used in the experiments in order to find the best combination.

3.2 Character-Level CNN

Convolutional Neural Networks can discover patterns within the data. These patterns are then later used to classify new data samples. Character-level CNNs learn such patterns in texts by searching through sequences of characters. The inspiration for this model is drawn from Kim (2014), but instead of using word embeddings as inputs, we use character encodings. This means that every character from an alphabet of size t is mapped to a

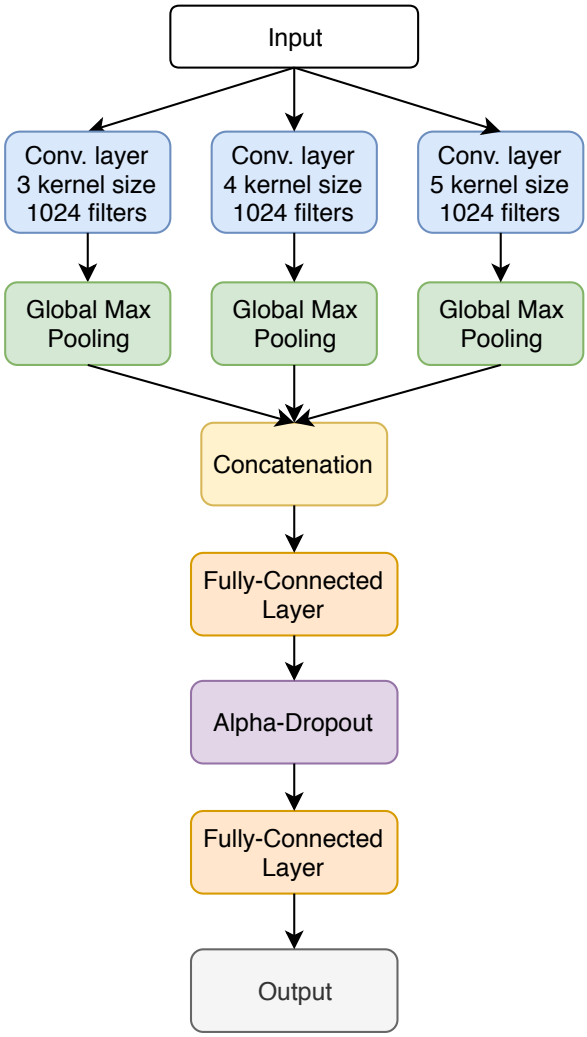

Figure 1: The architecture of the Character-level CNN model.

one-hot vector. For example, having the alphabet $\Sigma = \{a, b, c\}$, the encoding for character a is $[1, 0, 0]$, for b is $[0, 1, 0]$, and for c is $[0, 0, 1]$. Each character from the input text is encoded, and only a fixed size l of the input is kept. In the experiments presented in this paper, l was set to 270 characters. The documents that are under the length were zero-padded. The alphabet was extracted from the dataset and it contains a total of 32 characters from which 26 are the lower case letters of the English alphabet, plus 6 Swiss-German diacritics (such as ã, ä, õ, ö, ü, ẽ). Characters that do not appear in the alphabet are encoded as a blank character.

As illustrated in Figure 1, the architecture is 5 layers deep. The first layer is composed of three different convolutional layers, each followed by a global max-pooling layer. The third layer concatenates the features extracted from the global max-pooling layers, then passes the concatenation to two fully-connected layers. The convolutional layers are based on one-dimensional filters, one with filter size 3, another one with filter size 4, and the last one with filter size 5. After the concatenation step, the activation maps pass through a fully-connected layer having ReLU activation and after that, through an alpha dropout layer with the drop probability of 0.1. The last fully-connected layer has a softmax activation, which provides the final output. All convolutional layers have 1024 filters, and the first fully-connected layer has 256 neurons. The network is trained with the Adam optimizer using categorical cross-entropy as loss function, and a learning rate of 0.001.

3.3 Long Short-Term Memory Network

Long Short-Term Memory networks (LSTMs) have the capacity to learn long-term dependencies from a sequence. When applied on text, LSTMs can discover connections between words within a sentence or a text. Those connections can help the learning algorithm to find patterns that can later be used to solve a specific task. For example, discovering such connections can be useful to say if a text belongs to one class or another. Based on this idea, this paper proposes an LSTM model that can learn to discriminate between different dialects.

The input for this model is the same as the input used for the Char-CNN model. Characters were chosen as input because of the idea that between dialects, there are subtle character differences that makes a word belong to one dialect or another. The other reason is that there might be connections between substrings within the whole text of a specific dialect. The LSTM model defined for the GDI task is illustrated in Figure 2. The architecture consists of an LSTM layer, with the number of cells equal to 256, followed by a global max-pooling layer over the hidden state of each input character. The activation maps then pass through a fully-connected layer with 128 neurons, having ReLU activation. Finally, the predictions are computed by another fully-connected layer. The network is trained with the Adam optimizer using categorical cross-entropy as loss function, and a learning rate of 0.01.

3.4 Ensemble Model

Ensemble methods combine multiple learning algorithms to obtain a new model that has better performance than any individual model used in the ensemble. In this paper, a simple ensemble

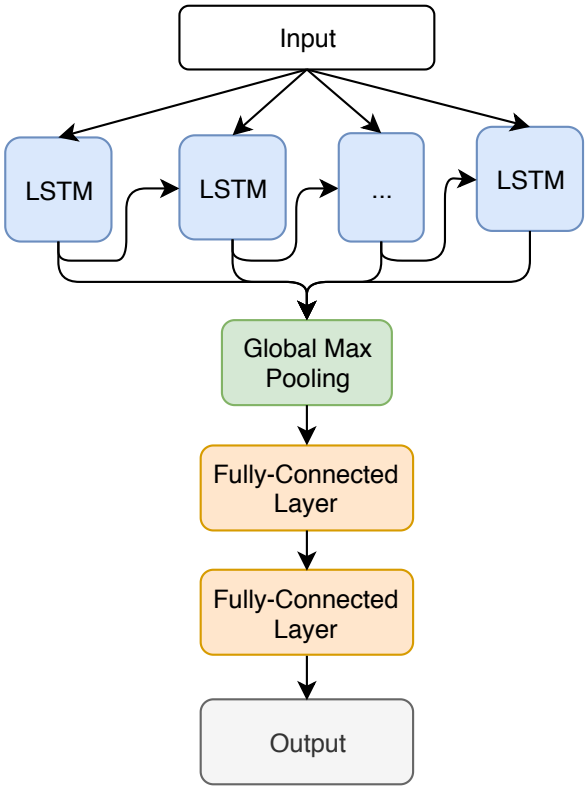

Figure 2: The architecture of the LSTM model.

method is employed. A voting schema is applied over the predictions from the Char-CNN, LSTM and String Kernel model. The vote from each model has equal weight. The class voted by the majority of the models is the final class for a sample data. If there is a tie, the class is chosen between the prediction of the Char-CNN model or the LSTM model, whichever prediction has the highest confidence among the outputs of the two models.

4 Experiments

4.1 Data Set

The 2019 GDI Shared Task data set (Zampieri et al., 2019) contains manually annotated transcripts of Swiss German speech, acoustic features for each transcript, and word-level normalization for each text. The task is to discriminate between Swiss German dialects from four different areas: Basel (BS), Bern (BE), Lucerne (LU), Zurich (ZH). As the samples are almost evenly distributed, an accuracy of 27.10% can be obtained with a majority class baseline on the test set. In our experiments, we used only the text transcripts to generate features.

4.2 Parameter and System Choices

The approach presented in this paper treats transcripts as strings. Because the approach works at the character level, there is no need to split the texts into words or to do any NLP-specific processing before computing the string kernels or feed the data to the deep networks. One thing to mention here is that for the deep networks the characters were mapped to one-hot encodings.

In order to tune the parameters and find the best system choices, the development set was used. Each model used in the ensemble was tuned. For tuning the parameters of the String Kernel method, we carried out a set of preliminary experiments to determine the optimal range of p-grams for each string kernel. We fix the learning method to KRR and evaluated all p-grams in the range $2-7$. The best accuracy (63.99%) is obtained with 4-grams. Having set the optimal number of p-grams, we experimented with different blended kernels to find out if combining p-grams of different lengths will improve the accuracy. For both kernels, presence and intersection, the best accuracy was obtained by combining p-grams with the length in range $3-5$.

After determining the optimal range of p-grams for each kernel function, we conducted further experiments by combining the presence bits kernel with the intersection kernel. When multiple kernels are combined, the features are actually embedded in a higher-dimensional space. As a consequence, the search space of linear patterns grows, which helps the classifier to select a better discriminant function. The most natural way of combining two or more kernels is to sum them up. The process of summing up kernels or kernel matrices is equivalent to feature vector concatenation. The results obtained by the individual kernels and also with the combined version are reported in Table 1. We can notice that even the individual kernels yield similar accuracy, when combined, the accuracy increases by a small amount.

After tuning the String Kernel method, we tuned the Character-level CNN. With the information discovered while tuning the String Kernels, we fixed the kernel size for the convolutional layers within the same range. Having the kernel size fixed, one convolution layer with kernel size 3, one with kernel size 4 and one with kernel size 5, we carried out further experiments in order to find the optimal number of filters and fully-connected lay-

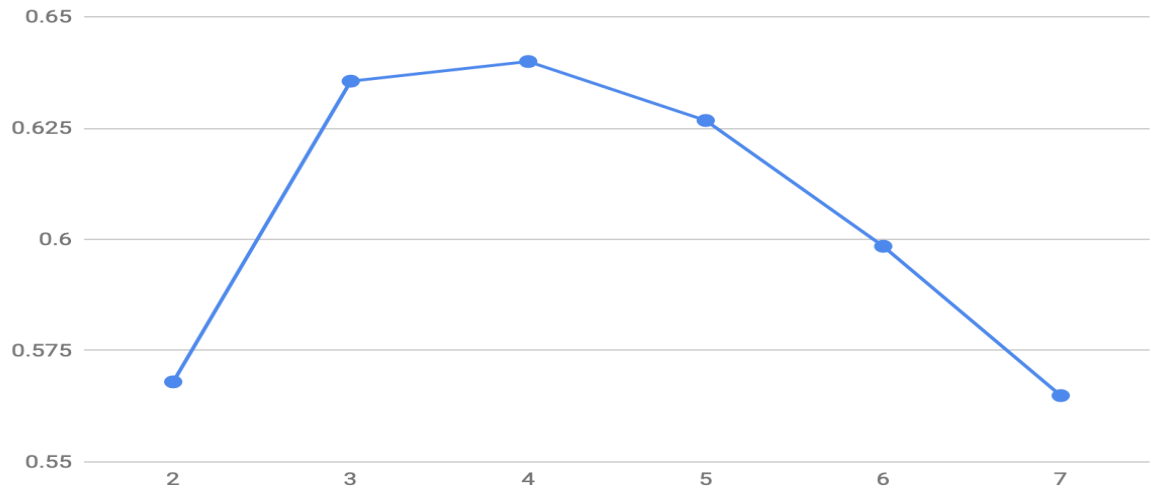

Figure 3: Accuracy rates of the KRR based on the presence bits kernel with p-grams in the range $2-7$.

Model	Accuracy
$\hat{k}^{0/1}_{3-5}$	66.05%
$\hat{k}^{\cap}_{3-5}$	66.00%
$\hat{k}^{0/1}_{3-5} + \hat{k}^{\cap}_{3-5}$	66.09%
$Char-CNN$	66.09%
$LSTM$	65.39%
$\hat{k}^{0/1}_{3-5} + \hat{k}^{\cap}_{3-5} + Char-CNN + LSTM$	67.95%

Table 1: Accuracy rates of various models used in experiments. The results are obtained over the development set.

ers. From those experiments, we discovered that having 1024 filters on each convolutional layer, and one fully-connected layer having 256 neurons employs the best result (66.09%). In the experiments, the model was trained using Adam optimizer having the learning rate set at 0.001 and with mini-batches of 64 samples. The network was trained for 25 epochs.

The last defined model was the LSTM model. For this model, we fixed the length of the hidden cell to be a number that is a power of two and close to the maximum length of a sample. Because the maximum length of a sample was set to 270 characters, we fixed the hidden cell number to be 256. The fully-connected layer was also fixed to 128 neurons from the beginning. For this model, we tuned the learning rate and the mini-batch size used for training. Through experiments, we fixed the learning rate to 0.01 and the mini-batch size to 32. This model was also trained using Adam optimizer. The best accuracy obtained with this model was 65.39% and it was obtained after training the model for 25 iterations.

Applying a voting schema over the three models, we observe an increase of almost 2% over the best individual model.

System	Accuracy	Macro-F_1
Run 1	62.66%	62.19%
Run 2	62.83%	62.38%
Run 3	62.96%	62.55%

Table 2: Results on the test set of the 2019 GDI Shared Task (closed training) of the method described in this paper.

4.3 Results

Table 2 presents our results for the German Dialect Identification Closed Shared Task for the 2019 VarDial Evaluation Campaign. The only difference between the three runs is the regularization parameter used in training the string kernel method. On the first run, the regularization parameter was set to 10^{-3}, on the second one it was set to 10^{-4}, and on the last run it was set to 10^{-5}. Among the submitted systems, the best performance is obtained when the KRR regularization parameter, for the String Kernel model, is set to 10^{-5}. The String Kernel model used in the three runs was trained on both training and development set. The submitted systems were ranked by their macro-F_1 scores and among the 6 participants, the best model that we submitted obtained the fourth place with a macro-F_1 score of 62.55%. The confusion matrix for the model is presented in Figure 4. The confusion matrix reveals that the system wrongly predicted over 400 samples of the Lucerne dialect as part of the Bern dialect. Furthermore, it has some difficulties in distinguish-

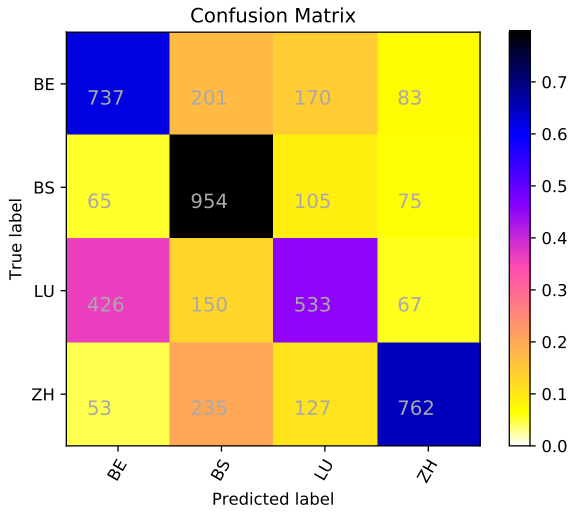

Figure 4: The confusion matrix of our best submission (run 3).

ing the Zurich dialect from the Basel dialect on one hand, and the Bern dialect from the Basel dialect on the other hand. Overall, the results look good, as the main diagonal scores dominate the other matrix components.

5 Conclusion

In this paper, we presented an approach for the GDI Shared Task of the DSL 2019 Challenge (Zampieri et al., 2019). The approach is based on an ensemble model that combines using a voting scheme results from three different models: Character-level Convolutional Neuronal Networks, Long Short-Term Memory network, and a String Kernel model. The approach obtained the fourth place.

References

Mohamed Ali. 2018. Character level convolutional neural network for arabic dialect identification. In *Proceedings of VarDial*, pages 122–127.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182.

Yonatan Belinkov and James Glass. 2016. A Character-level Convolutional Neural Network for Distinguishing Similar Languages and Dialects. In *Proceedings of VarDial*, pages 145–152.

Andrei M. Butnaru and Radu Tudor Ionescu. 2018. UnibucKernel Reloaded: First Place in Arabic Dialect Identification for the Second Year in a Row. In *Proceedings of VarDial*, pages 77–87.

Andrei M Butnaru and Radu Tudor Ionescu. 2019. Moroco: The moldavian and romanian dialectal corpus. *arXiv preprint arXiv:1901.06543*.

Mădălina Cozma, Andrei M. Butnaru, and Radu Tudor Ionescu. 2018. Automated essay scoring with string kernels and word embeddings. In *Proceedings of ACL*, pages 503–509.

Yann Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *Proceedings of ICML*, pages 933–941.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Hugo Jair Escalante, Thamar Solorio, and Manuel Montes-y-Gómez. 2011. Local Histograms of Character N-grams for Authorship Attribution. In *Proceedings of ACL: HLT*, volume 1, pages 288–298.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A Convolutional Encoder Model for Neural Machine Translation. In *Proceedings of ACL*, pages 123–135.

Rosa M. Giménez-Pérez, Marc Franco-Salvador, and Paolo Rosso. 2017. Single and Cross-domain Polarity Classification using String Kernels. In *Proceedings of EACL*, pages 558–563.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013a. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013b. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

David Haussler. 1999. Convolution Kernels on Discrete Structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nora Hollenstein and Noëmi Aepli. 2015. A resource for natural language processing of swiss german dialects. In *GSCL*, pages 108–109.

Radu Tudor Ionescu and Andrei M. Butnaru. 2017. Learning to Identify Arabic and German Dialects using Multiple Kernels. In *Proceedings of VarDial*, pages 200–209.

Radu Tudor Ionescu and Andrei M. Butnaru. 2018a. Improving the results of string kernels in sentiment analysis and Arabic dialect identification by adapting them to your test set. In *Proceedings of EMNLP*, pages 1084–1090.

Radu Tudor Ionescu and Andrei Madalin Butnaru. 2018b. Transductive learning with string kernels for cross-domain text classification. In *International Conference on Neural Information Processing*, pages 484–496. Springer.

Radu Tudor Ionescu and Marius Popescu. 2016. UnibucKernel: An Approach for Arabic Dialect Identification based on Multiple String Kernels. In *Proceedings of VarDial*, pages 135–144.

Radu Tudor Ionescu and Marius Popescu. 2017. Can string kernels pass the test of time in native language identification? In *Proceedings of BEA-12*, pages 224–234.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? A language-independent approach to native language identification. In *Proceedings of EMNLP*, pages 1363–1373.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2016. String kernels for native language identification: Insights from behind the curtains. *Computational Linguistics*, 42(3):491–525.

Tommi Jauhiainen, Heidi Jauhiainen, and Krister Lindén. 2018. 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.

Rie Johnson and Tong Zhang. 2015. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *Proceedings of NAACL*, pages 103–112.

Rohit J. Kate and Raymond J. Mooney. 2006. Using String-kernels for Learning Semantic Parsers. In *Proceedings of ACL*, pages 913–920, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP*, pages 1746–1751.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. In *Proceedings of AAAI*, pages 2741–2749.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of NIPS*, pages 1106–1114.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Tsungnan Lin, Bill G Horne, Peter Tino, and C Lee Giles. 1996. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Subhransu Maji, Alexander C. Berg, and Jitendra Malik. 2008. Classification using intersection kernel support vector machines is efficient. *Proceedings of CVPR*, pages 1–8.

Shervin Malmasi and Mark Dras. 2015. Language identification using classifier ensembles. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 35–43.

Shervin Malmasi, Mark Dras, and Marcos Zampieri. 2016. Ltg at semeval-2016 task 11: Complex word identification with classifier ensembles. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 996–1000.

Shervin Malmasi and Marcos Zampieri. 2017. Arabic Dialect Identification Using iVectors and ASR Transcripts. In *Proceedings of the VarDial*, pages 178–183.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*, pages 1532–1543.

Marius Popescu and Cristian Grozea. 2012. Kernel methods and string kernels for authorship analysis. In *Proceedings of CLEF (Online Working Notes/Labs/Workshop)*.

Marius Popescu, Cristian Grozea, and Radu Tudor Ionescu. 2017. HASKER: An efficient algorithm for string kernels. Application to polarity classification in various languages. In *Proceedings of KES*, pages 1755–1763.

Marius Popescu and Radu Tudor Ionescu. 2013. The Story of the Characters, the DNA and the Native Language. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 270–278.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Proceedings of NIPS*, pages 91–99.

Yvan Saeys, Thomas Abeel, and Yves Van de Peer. 2008. Robust feature selection using ensemble feature selection techniques. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 313–325. Springer.

Tanja Samardžić, Yves Scherrer, and Elvira Glaser. 2016. ArchiMob–A corpus of spoken Swiss German. In *Proceedings of LREC*, pages 4061–4066.

Conrad Sanderson and Simon Guenter. 2006. Short Text Authorship Attribution via Sequence Kernels, Markov Chains and Author Unmasking: An Investigation. In *Proceedings of EMNLP*, pages 482–491, Sydney, Australia. Association for Computational Linguistics.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of ICML*, pages 1818–1826.

Yves Scherrer and Owen Rambow. 2010. Word-based dialect identification with georeferenced rules. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1151–1161. Association for Computational Linguistics.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper With Convolutions. In *Proceedings of CVPR*.

Andrea Vedaldi and Andrew Zisserman. 2010. Efficient additive kernels via explicit feature maps. *Proceedings of CVPR*, pages 3539–3546.

Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. 2014. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, 57:77–93.

Rui Xia, Chengqing Zong, and Shoushan Li. 2011. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 181(6):1138–1152.

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 VarDial*, pages 1–17.

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of NIPS*, pages 649–657.