Optimizing the Size of Subword Vocabularies in Dialect Classification

Vani Kanjirangat
IDSIA-USI/SUPSI, Switzerland
vanik@idsia.ch

Tanja Samardžić
URPP Language and Space, UZH
tanja.samardzic@uzh.ch

Ljiljana Dolamic
armasuisse S+T, Switzerland
Ljiljana.Dolamic@armasuisse.ch

Fabio Rinaldi
IDSIA-USI/SUPSI, Switzerland
fabio.rinaldi@idsia.ch

Abstract

Pre-trained models usually come with a predefined tokenization and little flexibility as to what subword tokens can be used in downstream tasks. This problem concerns especially multilingual NLP and low-resource languages, which are typically processed using cross-lingual transfer. In this paper, we aim to find out if the right granularity of tokenization is helpful for a text classification task, namely dialect classification. Aiming at generalizations beyond the studied cases, we look for the optimal granularity in four dialect datasets, two with relatively consistent writing (one Arabic and one Indo-Aryan set) and two with considerably inconsistent writing (one Arabic and one Swiss German set). To gain more control over subword tokenization and ensure direct comparability in the experimental settings, we train a CNN classifier from scratch comparing two subword tokenization methods (Unigram model and BPE). For reference, we compare the results obtained in our analysis to the state of the art achieved by fine-tuning pre-trained models. We show that models trained from scratch with an optimal tokenization level perform better than fine-tuned classifiers in the case of highly inconsistent writing. In the case of relatively consistent writing, fine-tuned models remain better regardless of the tokenization level.[1]

1 Introduction

The change from word to subword tokenization opened a large space of tokenization possibilities: any substring of a word (subword) is potentially a good token, but some might be more useful than others. In contrast to this, pre-trained models usually come with a predefined tokenization and little flexibility in input preprocessing.

This problem is even more important in a multilingual setting, where, for many languages, only a little data is available, often written in a non-standard writing (e.g. transcriptions of spoken language, social media posts) with pronounced regional differences. Fine-tuning pretrained models (with cross-lingual transfer) has become the primary approach to processing such languages. Predefined tokenization, which is part of this research framework, is likely not to be suitable for the level of inconsistency that is typical for target low-resource languages.

In this paper, we study the benefits of optimal subword tokenization in one of the basic tasks in multilingual NLP — dialect classification. This task can be seen as a stand-alone task (e.g., for tracing the source of media posts) or a step in other end-user tasks such as machine translation or natural language understanding (NLU). We choose this task as an especially challenging case of text encoding bridging the work on language modelling and text classification. Although it is a classification task, it does not rely on an abstract semantic representation of the whole sentence (as in usual text classification) but on surface features of the text, such as distinctive suffixes or prefixes of words, phonetic clusters, and order of tokens, closer to language modelling. These features show up occasionally in the text, which otherwise might look the same in two different dialects (Zampieri et al., 2017; Tiedemann and Ljubešić, 2012). The right level of tokenization can be expected to help identify these features and thus encode the text better for other purposes too.

Aiming at generalizations beyond the studied cases, we work with four data sets (two Arabic, one Indo-Aryan, and one Swiss German) representing two levels of writing consistency (transcribed speech vs. originally written text) and three different types of languages. We consider three levels of tokenization (character, subword, word) testing two main subword tokenization methods: one example of a probabilistic model (Unigram model (Kudo,

[1]We will release our code for replication of our results.

Tenth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2023), pages 14–30

2018)) and one example of a bottom-up compression algorithm (BPE (Sennrich et al., 2016), also implemented by Kudo and Richardson (2018)). To gain flexibility with varying the level of tokenization, we train our own classifiers (one Bidirectional Long Short Term Memory (BiLSTM) and two Convolutional Neural Networks (CNNs)), which we also evaluate against comparable fine-tuned classifiers with BERT-based pre-trained models. Our findings are expected to generalize to other tasks similar to dialect classification and, to a certain degree, to NLU tasks.

2 Related Work

Dialect identification replaces language identification whenever a language has many regional variants as in the case of Arabic, Chinese or (Swiss) German. Such cases are mostly covered in a series of shared tasks (Zampieri et al., 2017, 2018, 2019; Chakravarthi et al., 2021; Gaman et al., 2020). The solutions submitted to the shared tasks range from traditional machine learning to state-of-the-art deep learning models. Traditional machine learning classifiers such as Support Vector Machines (SVM), Logistic Regression (LR), and Naive Bayes (NB) utilizing character or word level n-gram features were found to perform quite well across different languages and dialects (Çöltekin and Rama, 2016; Jauhiainen et al., 2018b; Zirikly et al., 2016; Bestgen, 2021; Jauhiainen et al., 2021). For instance, a version of a character-level n-gram language model with a domain adaptation technique is the state of the art for identifying Swiss German (F-score 0.75) and Indo-Aryan (F-score 0.96) dialects without acoustic features (Jauhiainen et al., 2018a,b, 2019). This approach, however, requires numerous model retraining iterations, which is not suitable for larger models.

Neural networks have been used for this task too including CNN, LSTM, and pre-trained Transformers models, in which are currently prevailing (Bernier-Colborne et al., 2019; Ceolin, 2021; Zaharia et al., 2020; Butnaru, 2019). The performance of these models varies depending on the datasets. Ensembles of neural and traditional models are also utilized (Popa and Ștefănescu, 2020; Hu et al., 2019; Yang and Xiang, 2019).

Character-level tokenization proves useful for capturing the relevant features, but previous studies do not address specifically the question of input granularity.

Outside of dialect identification, Domingo et al. (2018) suggest that tokenization could impact neural machine translation (NMT) quality. They compared tokenizers such as Moses, SentencePiece, OpenNMT, Stanford, and Mecab on Japanese, Russian, Chinese, German and Arabic translations to English. They found that Moses tokenizer gave the best result for Arabic, Russian and German; Mecab for Japanese; and Stanford for Chinese. Uysal and Gunal (2014) studied the effect of pre-processing in the English and Turkish languages and they observed that using appropriate domain and language dependant pre-processing can improve the performance. Gowda and May (2020) propose a general optimization method for finding subword tokens for machine translation. Mielke et al. (2019) find that the surprisal of a language model is minimised cross-linguistically at a particular level of subword segmentation with the resulting size of the input vocabulary being the word-level vocabulary multiplied by 0.4. Gutierrez-Vasques et al. (2021) find that much smaller vocabularies minimize text redundancy and lead to a converging text entropy across 47 languages.

Quite a few solutions have been proposed for unsupervised subword segmentation (Creutz and Lagus, 2005; Schuster and Nakajima, 2012; Poon et al., 2009; Narasimhan et al., 2015; Sennrich et al., 2016; Bergmanis and Goldwater, 2017; Grönroos et al., 2020; Kudo, 2018). The SentencePiece library (Kudo and Richardson, 2018) implements two very popular methods: BPE, a general data compression algorithm (Gage, 1994) first applied to text by Sennrich et al. (2016) and Unigram model (Kudo, 2018), similar to Morfessor (Creutz and Lagus, 2005; Grönroos et al., 2020) in that it considers multiple possible subword splits at the same time. Some related works on fine-tuning vocabulary sizes for NLP applications include (Cherry et al., 2018; Xu et al., 2021; Ding et al., 2019; Li et al., 2021).

The work on comparing subword tokenization algorithms reports rather inconsistent outcomes. For instance, Vania and Lopez (2017) find that BPE gives better results than Morfessor on the task of language modeling, but Ataman and Federico (2018) show that linguistically motivated vocabulary reduction (LMVR), which is an extension of Morfessor, gives better results in the context of machine translation. The benefit of using LMVR increases with increased morphological richness.

A similar conclusion is reached in a wide-scope multilingual comparison with language modeling as the downstream task is performed by Park et al. (2021). A study on English by Bostrom and Durrett (2020) compares BPE preprocessing with the Unigram method by Kudo and Richardson (2018), again on the task of language modeling, obtaining lower results with BPE tokenization, which also gives a slightly larger vocabulary than the competing method.

For our study, we select representative examples of neural models for dialect classification and subword tokenization methods, which are detailed in the next section.

3 Data and Methods

Four datasets used in the main study have been selected so that they represent different language types and different levels of consistency in writing. Table 1 shows the sizes of four datasets expressed as the number of utterances, the number of unique characters (character-level vocabulary) and the number of unique word types (word-level vocabulary). Three of the datasets were released as a part of the VarDial workshop shared tasks (Zampieri et al., 2017, 2018, 2019): the German Dialect Identification (**GDI**)[2], Indo-Aryan Language Identification (**ILI**) [3] and the Arabic Dialect Identification (**ADI**) datasets[4]. The fourth is the Arabic Online Commentary (**AOC**) (Zaidan and Callison-Burch, 2011) dataset.

The GDI dataset is compiled from the ArchiMob corpus of Spoken Swiss German and covers four areas, namely Basel, Bern, Lucern, and Zurich (Samardzic et al., 2016). We used the GDI-2018 dataset for our experiments and worked in a 4-way classification setting.

The ILI dataset includes five closely related Indo-Aryan language dialects: Hindi, Braj Bhasha, Awadhi, Bhojpuri, and Magahi. For each language, 15,000 sentences are extracted, mainly from the literature domain.

The ADI VarDial task (Malmasi et al., 2016; Ali et al., 2016) includes five Arabic dialects: Modern Standard Arabic (MSA), Egyptian (EGY), Gulf (GLF), Levantine (LAV), Moroccan (MOR), and North-African (NOR). MSA is the modern variety of language which is used in news and educational articles. It differs lexically, syntactically, and phonetically from the actual communication language of native speakers. The VarDial ADI dataset is both speech transcribed and transliterated to English from Arabic.

AOC constitutes a large-scale repository of Arabic dialects extracted from reader commentary of three Arabic online newspapers. It covers MSA and the dialectal varieties, viz., Egyptian (EGY), Gulf (GLF), Levantine (LEV), and Moroccan (MOR).

The languages and dialects represented in these four datasets belong to two language families (Indo-European and Semitic). Two of the data sets (GDI and ADI) are created by transcribing spoken language and show a high level of inconsistency in writing. The other two (ILI and AOC) are originally written texts with lower level of inconsistency.

In addition to these four datasets used in the main study, we perform additional experiments on the data from the Nuanced Arabic Dialect Identification (NADI) shared task, which deals with country-level and province-level Arabic dialect identifications (Abdul-Mageed et al., 2020, 2021). NADI 2022 shared task covers 18 country dialects with a training set of $\approx$ 20K tweets (Abdul-Mageed et al., 2022).

3.1 Levels of Tokenization

Dialect classification is usually performed at the level of utterance (loosely structured sentence): each utterance in a dataset is assigned a label. Classification features (typically n-grams) are typically either word-level or character-level. We introduce subword-level features and compare them to both character and word-level ones.

Word Level The most common tokenization is at the word level, mainly using white spaces and punctuation as delimiters. However, this approach is not convenient for languages lacking clear word boundaries (e.g., Chinese and Japanese). This type of tokenization produces large vocabularies, but shorter sequences, which are both important concerns for memory and time complexity.

Character Level Character level tokenization is the simplest way of segmenting the text using Unicode characters as tokens. This level is good for generalizing across languages (many languages share alphabets). It also helps solving some problems of word-level tokenization, such as out-of-

[2] https://drive.switch.ch/index.php/s/DZycFA9DPC8FgD9

[3] https://github.com/kmi-linguistics/VarDial2018

[4] https://arabicspeech.org/resources/

	GDI	ILI	ADI	AOC	NADI
Train	14647	68453	14591	86541	20398
Dev	4659	8286	1566	10820	4871
Test	4752	9032	1492	10812	4871
Word vocabulary (Train)	15041	115766	43150	171184	56163
Character vocabulary (Train)	30	209	52	158	445

Table 1: The size of datasets expressed as the number of utterances. The character (Character vocabulary) and word vocabulary (Word vocabulary) sizes (unique number of characters and words in the training set) is also given.

vocabulary (OOV) symbols. However, representing single characters is hard (too general) and sequences of character-level tokens are very long. Both of these factors have a negative impact on the performance on downstream tasks.

Subword Level The main idea behind the subword-level tokenization is to balance generalization and specificity so that frequently used words are considered a single token (as in word-level tokenization) and rare words are split into smaller units (as in character-level tokenization) called *subwords*. For instance, the word *lowest*, may be split into *low* and *est* depending upon the vocabulary sizes or merge operations. This helps in creating smaller vocabularies while preserving some of the lexical meaning.

3.2 Subword Tokenization Methods

Among many possibilities listed in Section 2, we select two methods, which represent two main approaches to finding subword units. We select **BPE** (Gage, 1994; Sennrich et al., 2016) as a bottom-up algorithm that goes from single characters to subwords by a sequence of merges. As an alternative approach, we select the **Unigram** model (Kudo, 2018), which considers all possible splits of a word gradually discarding some of them.

For BPE, text input is first tokenized at the word level. Each word is then split into a sequence of characters to which a special "end of the word" symbol is appended. The base vocabulary is created from the unique characters in the training corpora. The algorithm iterates through the data many times merging the most frequent pair of symbols into a single symbol every time. The new symbol is added to the vocabulary for the next iteration. The procedure is repeated until the desired vocabulary size, or a specific number of merge operations is obtained, which are the hyperparameters to be tuned.

Unlike the BPE algorithm, the Unigram model can be viewed as a probabilistic mixture model, where the likelihood of the whole data is computed under a given subword split hypothesis. The algorithm starts from a large vocabulary that contains many possible subword splits (a "reasonably" big seed vocabulary). It then reduces the vocabulary gradually by discarding a percentage of vocabulary entries. The decision on what entries to discard relies on a loss function: for each vocabulary entry, measure the difference in the overall likelihood of the data with and without that entry. Those entries that result in the smallest difference are discarded. A threshold $\eta\%$ is set to decide the percentage of vocabulary entries to be discarded. The process is repeated until the desired vocabulary size is reached, which is the hyperparameter to be tuned.

3.3 Optimizing the Size of the Subword Vocabulary

We optimize vocabulary sizes (vocab_sizes) for word-level and subword-level tokenization and take the character-level vocabularies from the data as the only option.

In case of the word-level tokenization, we conducted experiments with different vocab_sizes (2000-20000) and selected the vocab_size that yielded maximum performance on the dev set. Based on the experiments, we found the preferred word level vocab_size is 2000 for the dialect classification task on the specific languages tested. The unknown tokens are represented by *UNK*.

To find the range of vocab_sizes for subword level experiments, we consider different sizes from the character set to a limit identified by Mielke et al. (2019), who find that a BPE vocabulary corresponding to a proportion of the size of all word types $|V|$ minimizes the negative log-likelihood on the data (dev sets) across 21 languages from the Europarl dataset [5]. This proportion is the same for all languages: $0.4 * |V|$. Given this measure,

[5] https://www.statmt.org/europarl/

we consider all the subword vocab_sizes ranging from character level vocab_size to $0.4 * |V|$. These ranges for each dataset are reported in Table B1.

For finding the BPE and Unigram model vocabularies, we use the Google SentencePiece library[6], which is an unsupervised tokenizer-detokenizer that accepts raw input (no pre-tokenizations) with predefined vocabulary sizes as arguments. It adopts the BPE algorithm by Sennrich et al. (2016), but unlike specifying the required number of subword merge operations, here the desired final vocab_size has to be given (both approaches yield similar results). We start from the character vocab_size and increment the size by 100 if vocab_size $\leq$ 1000 and then by 1000 if vocab_size $\leq$ 10000. The process is repeated until the merge_size (number of merges) $\leq$ optimal_merge_size $(0.4 * |V|)$.

3.4 Models for Classification

For selecting the classification models, we consider two kinds of neural networks with shared parameters: convolutional (CNNs) and recurrent (specifically LSTM RNN). On the side of CNNs, we evaluate two concrete architectures: Kim_CNN (Kim et al., 2016) and Zhang_CNN (Zhang et al., 2015), which are known to perform well on the task of text classification and are widely used. On the side of RNNs, we evaluate the architecture Lin_SA_BiLSTM (Lin et al., 2017), which has been shown to give good results on the task of dialect classifications (Goswami et al., 2020). We manipulated the tokenizers of these models using different granularity levels without changing the overall architecture. The model architectures are briefly described in this section.

Lin_SA_BiLSTM This is a BiLSTM architecture with a self-attention component (Lin et al., 2017), where the sentence embeddings are computed by multiplying the hidden states from BiLSTM with the attention weights obtained across multiple attention hops. If $S = (w_1, w_2, ..., w_n)$ represents a sentence with n tokens, where w_i represents a d dimensional word embedding, then the sentence is represented by a 2D matrix of the shape $n \times d$. The BiLSTM component is used to compute the hidden state matrix H and further, the attention module takes the H vector and outputs the attention matrix A using the Equation 1:

$$A = softmax(W_{s2}tanh(W_{s1}H^T)) \quad (1)$$

Here, W_{s1} and W_{s2} represent the weight matrices. The final embedding is computed as $M = AH$. A penalization term is also used to ensure diversity among multiple attention hops. These embeddings are then to be used as input for a downstream task, such as dialect classification in our case.

Zhang_CNN Zhang et al. (2015) proposed a simple character level model for text classification utilizing a 1D convolution followed by max pooling layers. The model has six CNN layers and three fully connected layers.

Kim_CNN The architecture used by Kim et al. (2016) is originally a neural language model (NLM) used for several NLP tasks. We adapted it in particular for dialect classification. The original architecture uses a CNN with a highway network whose output is given to a recurrent neural network (RNN) neural language model. In the original Kim_CNN model, the input is segmented at the character level and hence a word token of length k is represented as $c_1, c_2, ..., c_k$. A filter F of width m is used to produce the feature maps. The main idea is that a filter captures the n-grams and the filter width corresponds to the n-gram size. Then a max-pooling layer is used to extract the important features. Since, our task is a classification problem, we utilized only the encoder part of the model with CNN, while the RNN layers were replaced by dense layers to perform softmax over the classes. The model has four convolutional layers and two fully connected layers.

4 Experimental Settings

We train and test on the task of dialect classification each of the architectures described in Section 3.4 on each version of the data produced with the tokenizers (one version of the data for each vocab_size). The vocabulary size that gave the best performance on the development set is chosen as the optimal vocabulary size. We compare these results to find out if optimizing the input vocabulary improves the classification performance. In addition to this, we compare the performance achieved with the best performing models trained from scratch with the performance achieved by fine-tuning respective pre-trained models.

In the remainder of this section, we describe the hyperparameters of the neural models trained from scratch, the vocabulary settings, and the fine-tuning settings, which we consider to be the state of the

[6] https://github.com/google/sentencepiece

art.[7]

4.1 Hyperparameters

For all the models trained from scratch, we used a *batch_size* of 128 and maximum input length (*max_len*) of 1014 (decided after repeated experiments). The number of epochs is decided by early stopping criteria, monitoring the validation loss with patience value set to 2. The optimal number of epochs ranged between 5-10. For initialization, we used the Keras embedding layer[8], which takes integer encoded vocabulary and learns the vectors during training.

Table A1 in the Appendix reports the parameters for each model as described in the original implementations. In Lin_SA_BiLSTM, the main parameters are the LSTM hidden dimensions, dense layers dimension, and the number of attention hops in the self-attention mechanism. For Kim_CNN and Zhang_CNN, the main parameters include the number of CNN layers and fully connected neural network (FCNN) layers with their corresponding dimensions. The Kim_CNN uses a global max pooling layer, which is common in NLP applications. Zhang_CNN uses a 1D max pooling with specific pool sizes except for layers 3, 4, and 5. The *Kernel_size* represents the n-gram width, and the n-grams will be based on the granularity of the tokenizers.

4.2 Pre-trained Models and Fine-tuning

For comparisons, we use transformer based pretrained models. We evaluate Vanilla BERT (English BERT) and multilingual BERT (mBERT) (Devlin et al., 2019) for all the datasets. The language-specific BERT models are as follows: German BERT[9] and Swiss-German BERT [10] were used for the GDI dataset; IndicTransformers [11] (Jain et al., 2020) for ILI; AraBERT[12] (Antoun et al., 2020) and Multi-dialect-Arabic-BERT [13] (Talafha et al., 2020) for AOC, ADI and NADI datasets. German BERT is pretrained on the latest German Wikipedia dump (6GB of raw text files), OpenLegalData dump (2.4 GB), and news articles (3.6 GB). Swiss-German BERT is fine-tuned on the Swiss German data of the Leipzig Corpora Collection[14] and SwissCrawl[15] on the top of German BERT. IndicTransformers is a BERT model trained with 3 GB of data from the OSCAR corpus[16] covering three Indo-Aryan languages, Hindi, Bengali, and Telugu. AraBERT is pretrained on Arabic news articles and two publicly available large Arabic corpora covering 24 Arab countries on the top of a BERT-based model. Multi-dialect-Arabic-BERT initializes the weights from Arabic BERT and is further pretrained on 10M Arabic tweets from Nuanced Arabic Dialect Identification (NADI)[17] shared task.

For all the BERT based experiments, we used the pretrained models from HuggingFace library[18]. We trained each model for four epochs with Adam optimizer using a learning rate of 2e-5 on the corresponding training set using 1 Tesla K80 GPU. Since all these baselines are BERT based, the default tokenizer is WordPiece.

5 Results and Comparisons

Since the Kim_CNN model gave the best results in all the from-scratch settings, we report only its performance in Table 2, in the test sets with the best vocab_sizes obtained. The detailed experimental results of all the models are reported in Appendix C, Tables C1 and C3.

From Table 2, it can be observed that subword level tokenizer performs better than their character and word level counterparts across all four datasets. Except for ILI and NADI, the Unigram model yields better results than BPE. Comparing the F1 scores, we noted an improvement of 3.9 points in GDI, 9.7 points in ILI, 5.2 points in AOC, 10.2 points in ADI and 3.4 points in NADI compared to the character level tokenizers. Similarly, comparing the subword level tokenizers with word level,

[7] We consider fine-tuned models the state of the art, despite the fact that simpler models can give better performance when combined with domain adaptation techniques. We note that domain adaptation can be combined with any model and should be evaluated separately.

[8] https://keras.io/api/layers/core_layers/embedding/

[9] https://www.deepset.ai/german-bert

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

[11] https://huggingface.co/neuralspace-reverie

[12] https://huggingface.co/aubmindlab/bert-base-arabert

[13] https://huggingface.co/bashar-talafha/multi-dialect-bert-base-arabic

[14] https://wortschatz.uni-leipzig.de/en/download/

[15] https://icosys.ch/swisscrawl

[16] https://oscar-corpus.com/

[17] https://sites.google.com/view/second-nadi-shared-task/home

[18] https://huggingface.co/models

Dataset	Number of Classes	Vocabulary Size				F-score (%)			
		Char	Uni	BPE	Word	Char	Uni	BPE	Word
GDI	4	30	2030	3030	15041	58	**61.9**	59.7	57
ILI	5	209	709	309	115776	78	84.8	**87.7**	85
AOC	4	158	8058	4058	171184	68	**73.2**	72.4	70
ADI	5	52	9052	952	43150	37	**47.2**	44.2	45
					Additional Experiment				
NADI	18	445	20045	7045	56163	13.3	16.2	**16.7**	16

Table 2: Performance of the Kim_CNN model at different tokenization levels. Char: Character-level, Uni: Unigram, BPE: Byte Pair Encoding, Word: Word-level. Kim_CNN gave the highest performance among the experimented non-pretrained neural models. The best result in each dataset is bolded.

Dataset	Best Model		F-score (%)	
	Pre-trained	**Kim_CNN**	**Pre-trained**	**Kim_CNN**
GDI	BERT-base-cased	Unigram	61.1	**61.9**
ILI	Indic Transformers	BPE	**88.1**	87.7
AOC	AraBERT	Unigram	**77.1**	73.2
ADI	AraBERT	Unigram	41.1	**47.2**
		Additional Experiment		
NADI	Multi-dialect-Arabic-BERT	BPE	**26.1**	16.7

Table 3: Comparison of the non-pretrained model with best tokenization level with the top performing baseline models in each dataset.

the F1 score was observed to increase by 4.9 points in GDI, 2.7 points in ILI, 3.1 points in AOC, 2.2 points in ADI and 0.7 points in NADI dataset. The optimal vocab_sizes are also reported, corresponding to vocab_size that gave the maximum F-scores. The variation with respect to different vocab_sizes in each dataset for the Kim_CNN with the Unigram model tokens is shown in Appendix D, Figure D1.

From these results, we conclude that optimized subword-level tokenization gives better dialect classification performance across all data sets (different languages, different levels of consistency) when working with a CNN architecture trained from scratch. Similar observations hold for all the non-transformer neural models in Table C1 in Appendix C.

5.1 Comparison with Fine-tuned Models

Table 3 shows the comparison between the results obtained in the trained (from scratch) setting and the best results obtained in the fine-tuned settings (with pre-trained models). The models that achieve the best results on each dataset are presented. The detailed results for all the models are given in Appendix C Table C3.

This comparison shows an interesting interaction between the writing consistency and performance on the classification task. For the two datasets with inconsistent writing (GDI and ADI, see Section 3 for details), the best scores are achieved with one of our models trained from scratch on optimized subword vocabulary (Kim_CNN with the Unigram model vocabulary). We note also that the best pre-trained setting in the case of GDI is BERT-base-cased and not the German BERT (see Table C3 in Appendix C for more details). In the case of ADI, Kim_CNN with the Unigram model tokenization improves the classification F1 score by 6.1 points compared to the best performing fine-tuned setting, which is the language-specific AraBERT model.

On datasets with more consistent writing (ILI and AOC), we see an opposite pattern: the best classification score is achieved in the fine-tuned settings using a language-specific pre-trained model (Indic Transformers and AraBERT respectively).

These results show that finding an appropriate level of tokenization granularity is especially important when datasets contain a considerable level of noise. Using pre-trained models does not bring the expected benefits unless one can count on a reasonably consistent writing. This conclusion is additionally reinforced by the scores obtained on

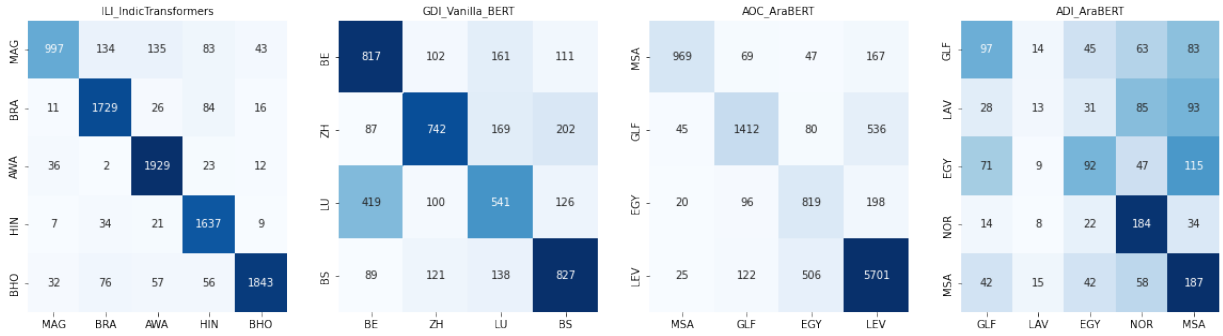

Figure 1: Confusion matrices for the best performing fine-tuned models on the ILI, GDI, AOC and ADI datasets

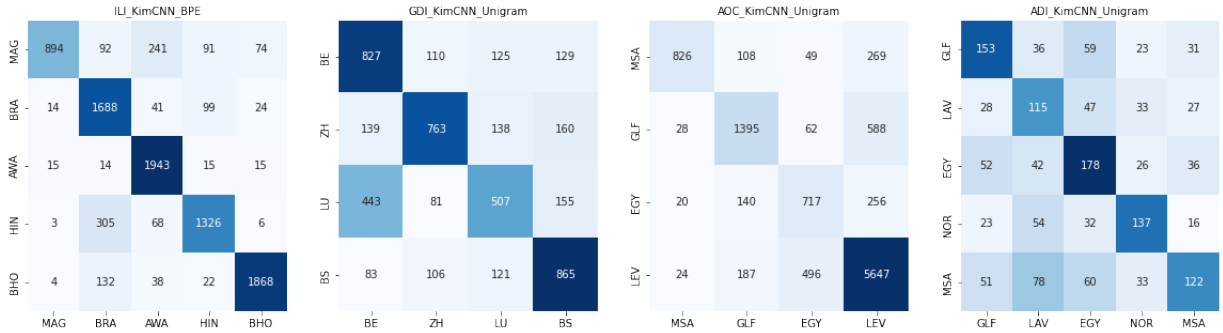

Figure 2: Confusion matrices Kim_CNN models (without pre-training) on the ILI, GDI, AOC and ADI datasets

the NADI dataset, where the fine-tuned classifier with a language specific pre-trained model achieves the best result. Even though the overall results are rather low in this case (likely due to the difficulty of distinguishing between 18 labels), they are better with fine-tuning. In this sense the NADI dataset, which also consists of originally written texts, patterns with ILI and AOC.

We also note the fact that BPE tokenizer gave better results than the Unigram model on 2/3 datasets with consistent writing. This observation is in line with previous research pointing out the sensitivity of BPE to noise in the data.

5.2 Per-class Comparison

To understand better the differences between Kim_CNN and the competing fine-tuned classifiers (results in Table 3), we plot two confusion matrices: Figure 1 shows the best performance with pre-trained models and Figure 2 shows the best performance with Kim_CNN. [19]

The matrices look very similar in all the cases except ADI. In this case, the fine-tuned classifier seems to have learned two classes well, while the success of Kim_CNN are more spread across different classes. The matrices for the AOC data set show that one class is much easier to identify for both approaches than the other classes. The GDI case shows one particularly confusing distinction (BE for Bern vs. LU for Luzern), which is almost equally hard for both approaches to distinguish. Finally, the class (MAG for Magahi) seems to be the most difficult for both approaches on the ILI dataset.

6 Conclusion

We have shown in this paper that optimizing subword vocabulary size is beneficial to text classification tasks, such as dialect classification, when the datasets contain relatively inconsistent writing (transcribed speech). With an optimized vocabulary as input, a CNN model trained from scratch outperforms fine-tuned models on such datasets. On the other hand, fine-tuning large language-specific pretrained models seems to be the best approach when datasets are relatively consistent (originally written, even if not edited). In this case, vocabulary size does not seem to matter much. Regarding the question of which kind of neural architecture is best to use without pretraining, our results point to the CNN architectures, which seem to capture the relevant surface features effectively.

Established on a relatively diverse sample (three

[19] We do not report the visualizations for NADI results here.

language types from two language families), our findings are especially relevant to multilingual NLP, where datasets tend to be inconsistent and the use of pre-trained models tempting.

7 Limitations

One of limitations of our work is the fact that we have not tried manipulating the tokenizers in BERT based models, which will be the focus of future work. In subword level tokenizers, we plan to explore other tokenizers such as WordPiece.

References

Muhammad Abdul-Mageed, Chiyu Zhang, Houda Bouamor, and Nizar Habash. 2020. NADI 2020: The first nuanced Arabic dialect identification shared task. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 97–110, Barcelona, Spain (Online). Association for Computational Linguistics.

Muhammad Abdul-Mageed, Chiyu Zhang, AbdelRahim Elmadany, Houda Bouamor, and Nizar Habash. 2021. NADI 2021: The second nuanced Arabic dialect identification shared task. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 244–259, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Muhammad Abdul-Mageed, Chiyu Zhang, AbdelRahim Elmadany, Houda Bouamor, and Nizar Habash. 2022. NADI 2022: The Third Nuanced Arabic Dialect Identification Shared Task. In *Proceedings of the Seven Arabic Natural Language Processing Workshop (WANLP 2022)*.

Ahmed Ali, Najim Dehak, Patrick Cardinal, Sameer Khurana, Sree Harsha Yella, James Glass, Peter Bell, and Steve Renals. 2016. Automatic dialect detection in arabic broadcast speech.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15.

Duygu Ataman and Marcello Federico. 2018. An evaluation of two vocabulary reduction methods for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 97–110, Boston, MA. Association for Machine Translation in the Americas.

Toms Bergmanis and Sharon Goldwater. 2017. From segmentation to analyses: a probabilistic model for unsupervised morphology induction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 337–346, Valencia, Spain. Association for Computational Linguistics.

Gabriel Bernier-Colborne, Cyril Goutte, and Serge Léger. 2019. Improving cuneiform language identification with bert. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 17–25.

Yves Bestgen. 2021. Optimizing a supervised classifier for a difficult language identification problem. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 96–101.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624.

Andrei Butnaru. 2019. 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.

Andrea Ceolin. 2021. Comparing the performance of cnns and shallow models for language identification. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 102–112.

Bharathi Raja Chakravarthi, Mihaela Gaman, Radu Tudor Ionescu, Heidi Jauhiainen, Tommi Jauhiainen, Krister Lindén, Nicola Ljubešić, Niko Partanen, Ruba Priyadharshini, Christoph Purschke, et al. 2021. Findings of the vardial evaluation campaign 2021. In *Proceedings of the 8th VarDial Workshop on NLP for Similar Languages, Varieties and Dialects*. The Association for Computational Linguistics.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305.

Çağrı Çöltekin and Taraka Rama. 2016. Discriminating similar languages with linear svms and neural networks. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 15–24.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *In Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05*, pages 106–113.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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.

Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. 2019. A call for prudent choice of subword merge operations in neural machine translation. In *Proceedings of Machine Translation Summit XVII: Research Track*, pages 204–213.

Miguel Domingo, Mercedes Garcıa-Martınez, Alexandre Helle, Francisco Casacuberta, and Manuel Herranz. 2018. How much does tokenization affect neural machine translation? *arXiv preprint arXiv:1812.08621*.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Mihaela Gaman, Dirk Hovy, Radu Tudor Ionescu, Heidi Jauhiainen, Tommi Jauhiainen, Krister Lindén, Nikola Ljubešić, Niko Partanen, Christoph Purschke, Yves Scherrer, et al. 2020. A report on the vardial evaluation campaign 2020. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–14.

Koustava Goswami, Rajdeep Sarkar, Bharathi Raja Chakravarthi, Theodorus Fransen, and John Philip McCrae. 2020. Unsupervised deep language and dialect identification for short texts. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1606–1617.

Thamme Gowda and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2020. Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3944–3953, Marseille, France. European Language Resources Association.

Ximena Gutierrez-Vasques, Christian Bentz, Olga Sozinova, and Tanja Samardzic. 2021. From characters to words: the turning point of bpe merges. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3454–3468.

Hai Hu, Wen Li, He Zhou, Zuoyu Tian, Yiwen Zhang, and Liang Zou. 2019. Ensemble methods to distinguish mainland and taiwan chinese. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 165–171.

Kushal Jain, Adwait Deshpande, Kumar Shridhar, Felix Laumann, and Ayushman Dash. 2020. Indic-transformers: An analysis of transformer language models for indian languages. *arXiv preprint arXiv:2011.02323*.

Tommi Jauhiainen, Heidi Jauhiainen, and Krister Lindén. 2018a. Iterative language model adaptation for indo-aryan language identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 66–75.

Tommi Jauhiainen, Krister Lindén, and Heidi Jauhiainen. 2019. 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. Association for Computational Linguistics.

Tommi Jauhiainen, Tharindu Ranasinghe, and Marcos Zampieri. 2021. Comparing approaches to dravidian language identification. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 120–127.

Tommi Sakari Jauhiainen, Heidi Annika Jauhiainen, Bo Krister Johan Linden, et al. 2018b. Heli-based experiments in swiss german dialect identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. The Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Jiahuan Li, Yutong Shen, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2021. When is char better than subword: A systematic study of segmentation algorithms for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 543–549.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Shervin Malmasi, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between similar languages and arabic dialect identification: A report on the third dsl shared task. In *Proceedings of the third workshop on NLP for similar languages, varieties and dialects (VarDial3)*, pages 1–14.

Sabrina J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What kind of language is hard to language-model? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4975–4989, Florence, Italy. Association for Computational Linguistics.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.

Hyunji Hayley Park, Katherine J. Zhang, Coleman Haley, Kenneth Steimel, Han Liu, and Lane Schwartz. 2021. Morphology matters: A multilingual language modeling analysis. *Transactions of the Association for Computational Linguistics*, 9:261–276.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado. Association for Computational Linguistics.

Cristian Popa and Vlad Ștefănescu. 2020. Applying multilingual and monolingual transformer-based models for dialect identification. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 193–201.

Tanja Samardzic, Yves Scherrer, and Elvira Glaser. 2016. Archimob-a corpus of spoken swiss german. In *Proceedings of the tenth international conference on language resources and evaluation (LREC 2016)*. European Language Resources Association (ELRA).

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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. Association for Computational Linguistics.

Bashar Talafha, Mohammad Ali, Muhy Eddin Za'ter, Haitham Seelawi, Ibraheem Tuffaha, Mostafa Samir, Wael Farhan, and Hussein Al-Natsheh. 2020. Multi-dialect arabic bert for country-level dialect identification. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 111–118.

Jörg Tiedemann and Nikola Ljubešić. 2012. Efficient discrimination between closely related languages. In *Proceedings of COLING 2012*, pages 2619–2634, Mumbai, India. The COLING 2012 Organizing Committee.

Alper Kursat Uysal and Serkan Gunal. 2014. The impact of preprocessing on text classification. *Information processing & management*, 50(1):104–112.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027, Vancouver, Canada. Association for Computational Linguistics.

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. 2021. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7361–7373.

Li Yang and Yang Xiang. 2019. Naive bayes and bilstm ensemble for discriminating between mainland and taiwan variation of mandarin chinese. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 120–127.

George-Eduard Zaharia, Andrei-Marius Avram, Dumitru-Clementin Cercel, and Traian Rebedea. 2020. Exploring the power of romanian bert for dialect identification. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 232–241.

Omar Zaidan and Chris Callison-Burch. 2011. The arabic online commentary dataset: an annotated dataset of informal arabic with high dialectal content. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 37–41.

Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. 2017. Findings of the vardial evaluation campaign 2017. In *Proceedings of the fourth workshop on NLP for similar languages, varieties and dialects*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Ahmed Ali, Suwon Shon, James Glass, Yves Scherrer, Tanja Samardžić, Nikola Ljubešić, Jörg Tiedemann, et al. 2018. Language identification and morphosyntactic tagging. the second vardial evaluation campaign.

Marcos Zampieri, Shervin Malmasi, Yves Scherrer, Tanja Samardžić, Francis Tyers, Miikka Silfverberg, Natalia Klyueva, Tung-Le Pan, Chu-Ren Huang, Radu Tudor Ionescu, et al. 2019. A report on the third vardial evaluation campaign. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Ayah Zirikly, Bart Desmet, and Mona Diab. 2016. The gw/lt3 vardial 2016 shared task system for dialects and similar languages detection. In *COLING*, pages 33–41. The COLING 2016 Organizing Committee.

A Model Hyperparameters

Model	Model Parameters	Parameter Values
Lin_SA_BiLSTM	LSTM hidden_dim	50
	Dense_layer_ dim	50
	Number of attention hops	10
Kim_CNN	Number of CNN layers	4
	Number of Filters	256
	Kernel_size	(10,7,5,3) respectively in each CNN layer
	Number of FCNN	2
	FCNN_dim	1024
Zhang_CNN	Number of CNN layers	6
	Number of Filters	256
	Kernel_size	7 in first two layers, 3 in other layers
	Pool_size	3 in first two layers and last layer (no pooling in other layers)
	Number of FCNN	3
	FCNN_dim	1024

Table A1: Parameter settings for the experimented neural models

B Subword Vocabulary Ranges

Dataset	vocab_size range (char_vocab_size - 0.4*|V|)
GDI	30-6016
ILI	209-46306
AOC	158-68473
ADI	52-17260
NADI	445-22465

Table B1: Subword vocabulary ranges considered in the experimental set-up for each dataset

C Detailed Experimental Results with Neural Models and Comparisons

From Table C3, it can be observed that between the three neural models experimented at different tokenization schemes, the subword level Kim_CNN model outperforms the Zhang_CNN and Lin_SA_BiLSTM models. Kim_CNN unigram model performs the best in GDI, AOC, and ADI with 61.9, 73.2, 47.21 % F1 scores, while the Kim_CNN BPE model presents the maximum performance in the ILI and NADI dataset with 87.79% and 16.7% F-scores. Compared with BERT based models, it can be noted that in GDI and ADI datasets, Kim_CNN performs slightly better than BERT models. In ILI, the subword level models surpass the vanilla BERT and mBERT. The NADI results were obtained from the official evaluation site [20].

Table C1 reports the performance based on accuracy, and F1 macro scores[21] and the vocab_sizes at which the peak performances are obtained (the best performances are bolded). It can be observed that in all the datasets except ILI, the best classification performance is obtained with the Kim_CNN Unigram model. In ILI, Kim_CNN BPE presented the best performance. During the analysis, we also observed that in all datasets except ADI, the vocab_sizes that presented the best performances were overlapping. The overlapping values are between 1000-6000 for GDI, 200-600 for ILI, and 700-5000 in AOC.

D Experiments on Vocabulary Sizes for Subword Tokenziers

Figure D1 depicts the variation of accuracy in Kim_CNN unigram model with respect to the different vocabulary sizes.

E Details of Experimental Runs

The BERT models trained on 1 Tesla K80 GPU took about 40-60 minutes training time and an inference time of 10-20 minutes. For the subword level experiments, the training of different subword level models took $\approx$ 50-60 minutes in HPC cluster and an inference time of 5-10 minutes.

[20] https://codalab.lisn.upsaclay.fr/competitions/6514#participate-submit_results

[21] Accuracy and Fmicro represent the same value for multi-class classification

Dataset	Model	Subword Tokenizers	Acc	F1	optimal vocab_size
GDI	Lin_SA_BiLSTM	BPE	28.5	28	830
		Unigram	59.18	59.2	4030
	Kim_CNN	BPE	59.53	59.73	3030
		Unigram	62.4	**61.9**	2030
	Zhang_CNN	BPE	56.25	55.19	4030
		Unigram	57.3	56.7	4030
ILI	Lin_SA_BiLSTM	BPE	81.3	79.4	20009
		Unigram	81.4	79.8	9009
	Kim_CNN	BPE	88.48	**87.79**	309
		Unigram	85.6	84.8	709
	Zhang_CNN	BPE	84.94	84.33	309
		Unigram	84.2	83.5	409
AOC	Lin_SA_BiLSTM	BPE	55.35	27.77	458
		Unigram	77.69	70.66	9058
	Kim_CNN	BPE	79.5	72.4	4058
		Unigram	79.4	**73.2**	8058
	Zhang_CNN	BPE	75.87	69.34	5058
		Unigram	79.4	73.2	8058
ADI	Lin_SA_BiLSTM	BPE	21.3	11.18	852
		Unigram	23.79	14.33	852
	Kim_CNN	BPE	45.37	44.2	952
		Unigram	47.25	**47.21**	9052
	Zhang_CNN	BPE	31.97	30.68	6052
		Unigram	32.8	31.2	6052
NADI	Lin_SA_BiLSTM	BPE	32.9	15.3	20045
		Unigram	16.1	5.6	845
	Kim_CNN	BPE	33.5	16.7	20045
		Unigram	31.4	16.2	7045
	Zhang_CNN	BPE	29.1	5.1	20045
		Unigram	29.2	4.9	9045

Table C1: Model performances (Accuracy and Fmacro%) with BPE and Unigram subword tokenizers and the optimal vocabulary sizes

Dataset	Model	Tokenization Levels	F1(%)
GDI	Lin_SA_BiLSTM	Character Level	49.4
		Subword_BPE	28
		Subword_Unigram	59.2
		Word Level	58
	Kim_CNN	Character Level	56.9
		Subword_BPE	59.7
		Subword_Unigram	**61.9**
		Word Level	57
	Zhang_CNN	Character Level	47
		Subword_BPE	55.2
		Subword_Unigram	56.7
		Word Level	25
ILI	Lin_SA_BiLSTM	Character Level	64.4
		Subword_BPE	79.4
		Subword_Unigram	79.8
		Word Level	84.6
	Kim_CNN	Character Level	76.9
		Subword_BPE	**87.8**
		Subword_Unigram	84.8
		Word Level	84.3
	Zhang_CNN	Character Level	80
		Subword_BPE	84.3
		Subword_Unigram	83.5
		Word Level	85
AOC	Lin_SA_BiLSTM	Character Level	63.3
		Subword_BPE	27.8
		Subword_Unigram	70.6
		Word Level	75.5
	Kim_CNN	Character Level	73.3
		Subword_BPE	**72.4**
		Subword_Unigram	**73.2**
		Word Level	65.6
	Zhang_CNN	Character Level	66.7
		Subword_BPE	69.3
		Subword_Unigram	73.2
		Word Level	66
ADI	Lin_SA_BiLSTM	Character Level	13.4
		Subword_BPE	11.18
		Subword_Unigram	14.33
		Word Level	15.6
	Kim_CNN	Character Level	36.6
		Subword_BPE	44.2
		Subword_Unigram	**47.2**
		Word Level	45
	Zhang_CNN	Character Level	23
		Subword_BPE	30.7
		Subword_Unigram	31.2
		Word Level	31
NADI	Lin_SA_BiLSTM	Character Level	14.5
		Subword_BPE	15.3
		Subword_Unigram	5.6
		Word Level	14
	Kim_CNN	Character Level	13.4
		Subword_BPE	16.7
		Subword_Unigram	16.2
		Word Level	16.1
	Zhang_CNN	Character Level	7.2
		Subword_BPE	5.1
		Subword_Unigram	4.9
		Word Level	2.6

Table C2: Comparisons(F1%) of the neural models analyzed using different tokenization levels

Dataset	Model	F1(%)
GDI	**Bert-base-cased**	**61**
	mBERT	59
	German BERT	60
	Swiss-German BERT	60
ILI	Bert-base-cased	80
	mBERT	87
	IndicTransformers	**88**
AOC	Bert-base-cased	75
	mBERT	76
	AraBERT	**77**
	multi-dialect-ArabicBERT	76
ADI	Bert-base-cased	40
	mBERT	23
	AraBERT	**41**
	multi-dialect-ArabicBERT	40
NADI	**Bert-base-cased**	4.8
	mBERT	4.9
	AraBERT	20
	multi-dialect-ArabicBERT	**26**

Table C3: Comparisons(F1%) of the different pre-trained models in each dataset

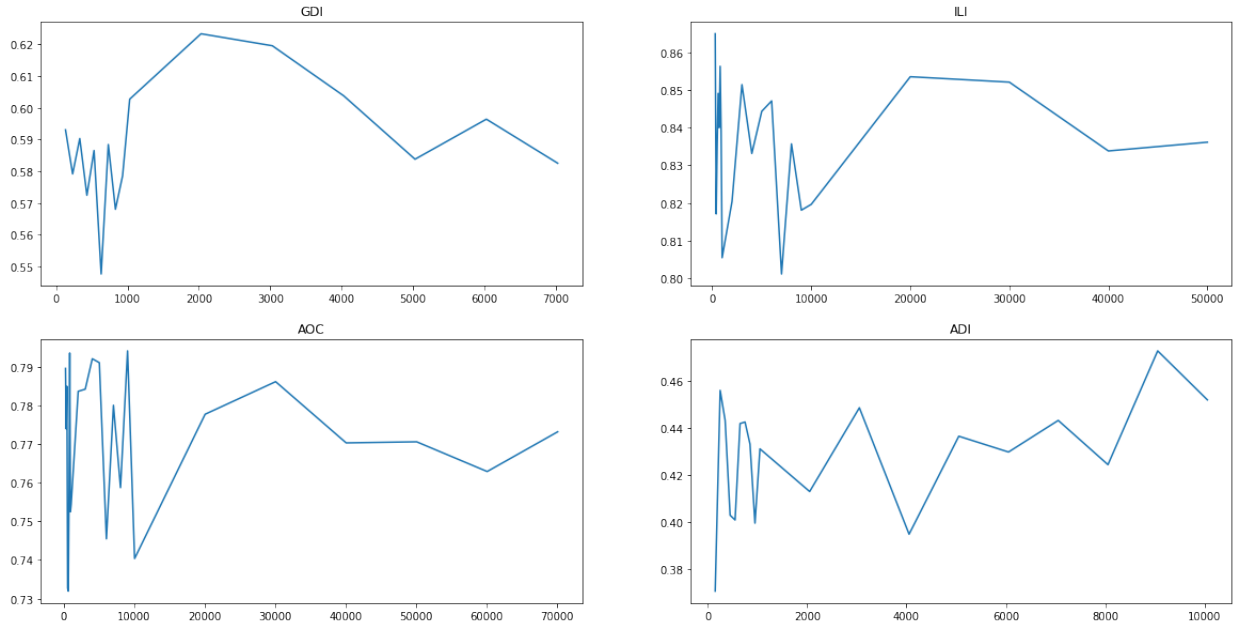

Figure D1: Variation of performances with respect to vocabulary sizes in Kim_CNN subword level unigram models across GDI, ILI,AOC and ADI datasets