Arabic Dialect Identification Using BERT-Based Domain Adaptation

Ahmad Beltagy   **Abdelrahman Wael**   **Omar ElSherief**

Faculty of Engineering, Alexandria University
{a.beltagy97, abdelrahman.abouelenin}@gmail.com,
C.m_elshrief@hotmail.com

Abstract

Arabic is one of the most important and growing languages in the world. With the rise of the social media platforms such as like Twitter, Arabic spoken dialects have become more in use. In this paper, we describe our our approach on the NADI Shared Task 1 that requires us to build a system to differentiate between different 21 Arabic dialects, we introduce a deep learning semi-supervised fashion approach along with pre-processing that was reported on NADI shared Task 1 Corpus. Our system ranks 4th in NADI's shared task competition achieving 23.09% F1 macro average score with a simple yet an efficient approach on differentiating between 21 Arabic Dialects given tweets.

1   Introduction

Arabic dialect classification is a task of identifying the dialect of the writer given an input text. This task has been an active field of research the past few years due to the rise of Arabic corpora which are made available (Bouamor et al., 2018). However, NADI's Arabic dialect corpus (Abdul-Mageed et al., 2020) have been quite challenging and intriguing. NADI's corpora has 21 different country dialects from which some of them are quite similar to each other in terms of morphology. Some of the data provided had some English words, others had Quran verses generated from third party apps which makes it extremely difficult for not just models, but also human cannot differentiate between the dialects if the tweet is a Quranic verse, since Quran compromises classical Arabic not dialectal Arabic. This is noise in the dataset that a model can not fix. Also, the imbalance of training data introduced other challenges and difficulties.

Previous work in this task involved the use of traditional ML algorithms, RNN with their variants, and hybrid approaches like in (Salameh et al., 2018). However, we were inspired by self-attention technique (Vaswani et al., 2017). Due to the huge success of transformers in many classification tasks, we opted for approach of using pretrained BERT (Devlin et al., 2018) in semi-supervised deep learning as it is proved by Gururangan et al. (2020) that fine-tuning pretrained model to specific domain is an efficient solution. We fine-tuned pretrained BERT transformer on "AraBERT" (Antoun et al., 2020) Arabic text, which is trained on 23B GB of data, using masked language modelling with Huggingface interface (Wolf et al., 2019). This process was done after doing data pre-processing and data augmentation in order to alleviate the problems of having class imbalance and lack of training data.

Transfer learning has proven to be an efficient approach in classification tasks, especially when we do not have enough training data. We leveraged the unlabeled data provided by NADI (Abdul-Mageed et al., 2020) through twitter API's, for fine-tuning using language modelling to achieve domain adaptation. Where we adapt pretrained AraBERT to the domain of tweets. That is because of similarity of morphological and semantic features that both the unlabeled data and target data share. Pretrained BERT model "AraBERT" was fine-tuned on NADI's unlabeled corpus using masked language model to let it learn better features. These features are going to be used to classify target labeled training examples provided by

262

NADI, which have the same distribution as the unlabeled ones. Our results confirm that such a technique have improved our baseline performance significantly by 3% F1 macro average score.

The rest of the paper is organized as follows. Section 2 presents the data that were supplied for training our model. Section 3 is a description of our system in detail and what infrastructure we used to produce the results. In Section 4, we present our results using various techniques. Section 5 is a discussion about task and common errors that occured. Section 6 concludes the work we have done as well as suggested future work.

2 Data

The dataset was one of the factors that made this problem quite challenging. NADI shared task organizers (Abdul-Mageed et al., 2020) provided a corpus of Arabic tweets from Twitter platform having 21 class labels corresponding to countries {Egypt, Iraq, Saudi Arabia, Algeria, Oman, Emirates, Libya, Syria, Morocco, Yemen, Tunisia, Lebanon, Jordan, Kuwait, Palestine, Qatar, Bahrain, Djibouti, Mauritania, Somalia and Sudan}. The corpus is divided into training set, dev set and test set to report our final results on. The number of examples in the 3 sets is, 21,000 tweets, 4,957 tweets and 5,000 tweets respectively. To aid in the training and model building processes, the organizers also provided additional 10 million unlabeled tweets IDs from the same distribution of the labeled tweets, to be obtained using a provided python script.

In Figure 1, we show the distribution of each class in the training examples. The figure clearly shows the imbalance of training examples where a class like Egypt has 4,473 example and a class like Sudan has only 210 examples. It also shows that most of the classes had under 1,000 example, which was quite challenging to solve.
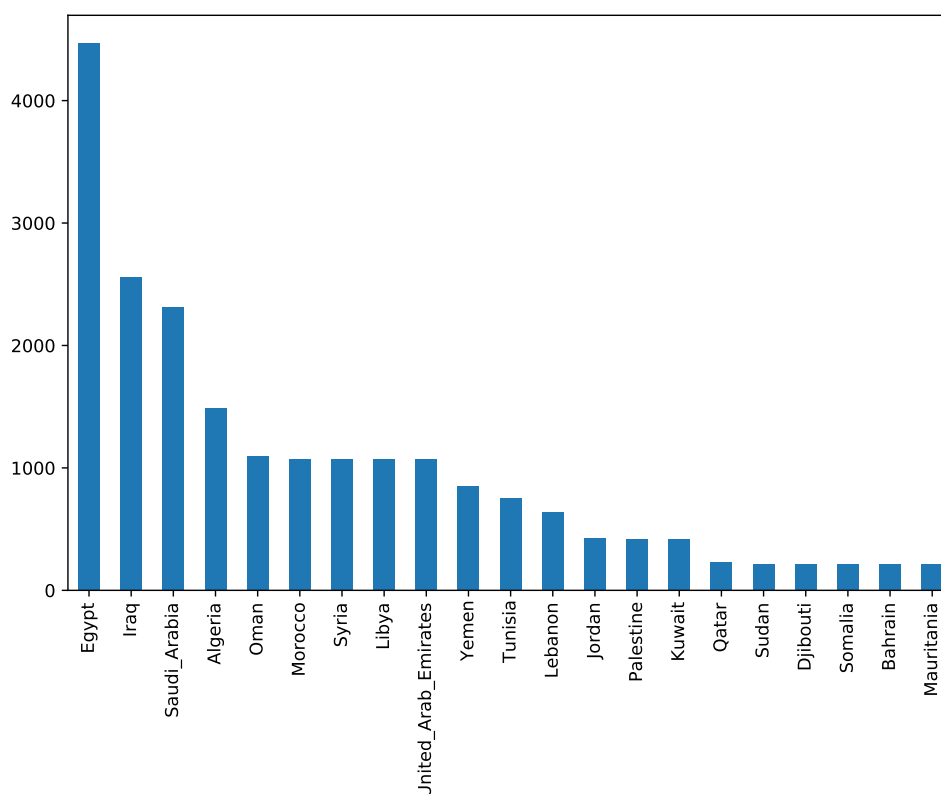

Figure 1: Distribution of 21 class labels across training data

2.1 Data Pre-processing

We start our pre-processing by cleaning the data and removing all the unnecessarily characters. We start the process of data cleaning by removing URLs, punctuation, mentions, email addresses, emojis and

other unknown Unicode characters using regex patterns. Next step we removed all English characters as it won't help in differentiating between Arabic dialects. Moreover, we used PyArabic library (Zerrouki, 2010) to strip tashkil from Arabic sentences. We stripped tashkil because we observed the inconsistent use of it and even wrong usage, which it would be a burden rather than an advantage to keep it. At this point, we have a corpus of only Arabic characters. The next step of cleaning is to remove elongations ex: كملت لو اهخ متععهههههه والله . Here we won't need متععههههه as it represents noise, we remove the repeated characters so the result متعه . Next we deal with the words that have same semantics but different syntax, those words differ not in the core of the word it self, but in the suffix and prefix, for example the word بعرفكيش, which means I don't know you, can be written in different forms with different dialects معرفكيش، ما بعرفك، لا أعرفك، مش هعرفك, so in order to provide more information to our model. We separate the suffix and prefix from the core of the word, for this step we use Farasa Segmenter (Abdelali et al., 2016), which is an Arabic NLP toolkit that serves as a sentence segmentation toolkit. Farasa takes a word and splits it into suffix and prefix, which completes our data processing step, example: امين يارب العالمين this is transformed to 'امين يارب ال'+'عالم'+'ين'

2.2 Data Augmentation

Because of the imbalance of data as shown in figure 1, some minority classes like {Lebanon, Jordan, Kuwait, Palestine, Qatar, Bahrain, Djibouti, Mauritania, Somalia and Sudan} had less than 750 examples. Our proposed solution to this problem is to upsample the minority classes to 750 examples using scikit-learn library (Pedregosa et al., 2011). We chose this specific number after many trials with different numbers. We observed that above 750 examples we do not observe any increase in terms of accuracy. Other techniques for data augmentation were suggested in (Fares et al., 2019; Ibrahim et al., 2018; Ibrahim et al., 2020)

3 System

In this section, we describe our proposed approach used in NADI shared task 1 Corpus submission. All of our experiments were based on AraBERT (Antoun et al., 2020) which is an Arabic version BERT model trained on 23GB of Arabic text with 3B words having vocab size of 64,000. We present the building blocks of our system, then we go over to explain another experiment that we have tried.

3.1 Tokenization and Encoding

First, we used the tokenizer corresponding to the model which mainly is used to split the sentence to tokens, example: خلص يبقى بعرفكيش this is transformed to 'ش','كي','رف','بع','يبقى','خلص'. Then we proceed to convert each word to it's appropriate ID and if the tweet is smaller than the expected dimension (64 word), then it's padded with an appropriate token " [PAD] ". If it exceed it, then it's truncated and a binary vector, also known as a mask, is returned to emphasize if certain token correspond to a word or padding token. Then we feed AraBERT the sequence of IDs along with it's corresponding mask vector and it's label corresponding to which class of the 21 mentioned in data section.

3.2 Feature Extraction

AraBERT is originally a pre-trained BERT (Devlin et al., 2018) specifically for the Arabic language. It has a vocab size of 64,000 word, and uses the same small Bert-Base configuration that has 12 transformer encoders. These transformers encoders learn about features in the input text and outputs 768 hidden dimension at the end. After each sentence is fed to the model, it is converted to a vector representing it using vocab dictionary and then fed to encoder layers of the model to form an output of (768).

3.3 Domain Adaptation Using Fine Tuning

We were inspired by power of domain adaptation in various Natural Language Processing tasks. It was proven by Gururangan et al. (2020) that adapting a large pre-trained model to another domain would gain a lot in terms of accuracy. We started this task by crawling some of the 10M unlabeled tweets using Twitter API python script provided by NADI organizers (Abdul-Mageed et al., 2020). We only utilized

2M tweets, as we did not observe any gain in accuracy when going beyond 2M tweets. We trained a masked language model using HuggingFace interface (Wolf et al., 2019), to let the model learn about semantic features of the new domain 'tweets'. The masked language model was trained using an adaptive learning rate starting with 2e-5 was used, and with masking rate of 15% of input text. We trained with only 1 epoch, which took around 20 hours to complete training. The rest of the hyper-parameters were the default ones provided by HuggingFace.

3.4   Training Classifiers

We experimented with 2 different settings to classify tweets. First, which was the one we submitted. It relied only on AraBERT to classify the 21 labels, achieving F1 macro average score of 24.433% on dev set. Second Trial was a mix between AraBERT and Naive-bayes achieving 22.3% F1 macro average score.

3.4.1   AraBERT classification head

After each tweet was encoded into 768 features, a simple BERT classification layer was added after multiple experiments. The layer would take in the 768 features and using fully connected network it converts them to output a vector of 21 entries. Where each entry uniquely correspond to one of the 21 Arabic dialect classes.

3.4.2   AraBERT with naive-bayes

One of the promising models and inspired by (Kowsari et al., 2017) was a hierarchical classification model. Having observed that deep learning doesn't perform well with classes with small number of tweets, we also observed how well naive Bayes performed classifying these classes, so we were motivated to build a hybrid model using AraBERT and naive Bayes.We The idea was simple, we used AraBERT to classify majority classes {Egypt, Iraq, Saudi Arabia, Algeria, Oman, Emirates, Libya, Syria, Morocco, Yemen, Tunisia, Lebanon, Jordan, Kuwait, Palestine} normally and combining minority classes {Qatar, Bahrain, Djibouti, Mauritania, Somalia and Sudan} into one class {Not majority} labelled C16 as shown in figure 2. If AraBERT classified a certain tweet to belong to this special class then this tweet is passed to naive Bayes to more accurately classify which class from the minority classes does this tweet belongs to. This model proved capable of achieving f1 score of 22% but still falls a bit short than a pure AraBERT.
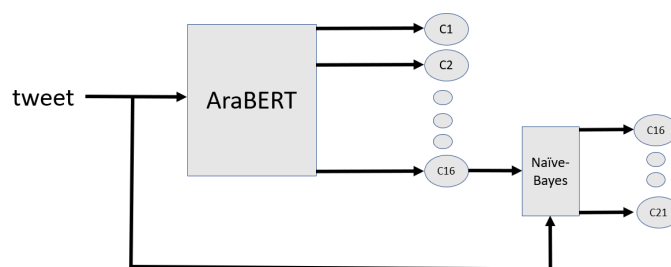

Figure 2: Visual of the classification head used in AraBERT with Naive Bayes. If AraBERT classifies a certain tweet to belong tominority class then it is passed to Naive-Bayes.

4   Results

We present our experiments results on the development set provided by organizers which consisted of 4,957 examples. In Table 1 we show our corresponding F1 macro average score on those models. We firstly show that our baseline model which consists of using AraBERT only resulted in 18.6% F1 macro average score. After cleaning process described in data section 2.1, we gained nearly 2% F1 macro average score which proved our pre-processing was efficient one. However, we noticed that our classifier

Model	Dev-set F1-score
baseline (AraBERT without augmentation and cleaning)	18.6%
AraBERT with cleaning	20.54%
AraBERT with cleaning + up sampling	21.33%
AraBERT with cleaning + up sampling + MLM fine-tuning	24.43%
AraBERT + Naive-Bayes with cleaning + up sampling	22.3%

Table 1: Our proposed models scores on dev-set.

is struggling with minority classes as described in data section 2.2, which gave us the idea of up sampling them. And that gained us a further 0.79% F1 macro average score. After further research we were fascinated by the power of fine-tuning approach of training a masked language model to improve classification task as described in system section 3.3 . This technique has given us the biggest improvement of 3.1% F1 macro average score. This was our submitted model in the task, which ranked 4th with 24.43% F1 macro average score in dev set and 23.09% on test set.

A second experiment which was not submitted was concluded based on hierarchical model of AraBERT with naive-bayes. However, this approach had an F1 macro average of 22.3%, which is still 2% F1 macro score less than a single transformer model.

5   Discussion

It was observed in our experiments, through analysis of data, that some of the tweets had noticeable noise that cannot be solved. Some users used to tweet through third party apps. We noticed that these tweets were mainly prayers or quran versus coming from various apps like {http://d3waapp.org, http://knzmuslim.com, http://Gharedly.com and http://du3a.org}. The main problem is that these tweets are given different labels according to the nationality of the user, and in fact they are not even differential by humans, which makes it very hard for a trained model. Another noticeable noise was that some tweets are actually a retweets. Where the retweeted content refers to a dialect, but it was given another label because the actual user retweeting is having a different dialect.

In future work, we plan on exploiting hierarchical models having transformer model, as it's backbone as we believe that regrouping our classes into different sets, would improve the results. Also another promising idea was to build a byte pair encoding tokenizer trained on extracting frequent sub words, which may prove to be helpful especially in dialect classification. We believe that the main differences between dialects are specific sub words that occurs in certain dialect more than others. We will also experiment ensemble techniques and how they can fit into our system.

6   Conclusion

We introduce a simple efficient single deep neural network model to classify 21 arabic dialects, which was based on the idea of fine-tuning approach using a pre-trained transformer model on similar domain after various cleaning and augmentation methods. We were able to achieve 4th best score in NADI shared task 1 competition with F1 macro average score of 22.03%.

References

Ahmed Abdelali, Kareem Darwish, and Hamdy Durrani, Nadir andz Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California, June. Association for Computational Linguistics.

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 (WANLP 2020)*, Barcelona, Spain.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding.

Houda Bouamor, Nizar Habash, Mohammad Salameh, Wajdi Zaghouani, Owen Rambow, Dana Abdulrahim, Ossama Obeid, Salam Khalifa, Fadhl Eryani, Alexander Erdmann, and Kemal Oflazer. 2018. The MADAR Arabic dialect corpus and lexicon. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May. European Language Resources Association (ELRA).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Youssef Fares, Zeyad El-Zanaty, Kareem Abdel-Salam, Muhammed Ezzeldin, Aliaa Mohamed, Karim El-Awaad, and Marwan Torki. 2019. Arabic dialect identification with deep learning and hybrid frequency based features. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 224–228, Florence, Italy, August. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*.

Mai Ibrahim, Marwan Torki, and Nagwa El-Makky. 2018. Imbalanced toxic comments classification using data augmentation and deep learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 875–878, Dec.

Mai Ibrahim, Marwan Torki, and Nagwa El-Makky. 2020. Alexu-backtranslation-tl at semeval-2020 task [12]: Improving offensive language detection using data augmentation and transfer learning. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 364–371.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Mohammad Salameh, Houda Bouamor, and Nizar Habash. 2018. Fine-grained arabic dialect identification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1332–1344.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Taha Zerrouki. 2010. pyarabic, an arabic language library for python.