Mawdoo3 AI at MADAR Shared Task: Arabic Tweet Dialect Identification

Bashar Talafha **Wael Farhan** **Ahmed Altakrouri** **Hussein T. Al-Natsheh**
Mawdoo3 Ltd, Amman, Jordan
{bashar.talafha, wael.farhan, ahmed.altakrouri, h.natsheh}@mawdoo3.com

Abstract

Arabic dialect identification is an inherently complex problem, as Arabic dialect taxonomy is convoluted and aims to dissect a continuous space rather than a discrete one. In this work, we present machine and deep learning approaches to predict 21 fine-grained dialects form a set of given tweets per user. We adopted numerous feature extraction methods most of which showed improvement in the final model, such as word embedding, Tf-idf, and other tweet features. Our results show that a simple LinearSVC can outperform any complex deep learning model given a set of curated features. With a relatively complex user voting mechanism, we were able to achieve a Macro-Averaged F1-score of 71.84% on MADAR shared subtask-2. Our best submitted model ranked second out of all participating teams.

1 Introduction

In recent years, an extensive increase in social media platforms usages, such as Facebook and Twitter, led to an exponential growth in the user-base generated content. The nature of this data is diverse. It comprises different expressions, languages, and dialects which attracted researchers to understand and harness language semantics such as sentiment, emotion, dialect identification, and many other Natural Language Processing (NLP) tasks. Arabic is one of the most spoken languages in the world, being used by many nations and spread across multiple geographical locations led to the generation of language variations (i.e., dialects) (Zaidan and Callison-Burch, 2014).

In this paper, we tackle the problem of predicting the user dialect from a set of his given tweets. We describe our work on exploring different machine and deep learning methods in our attempt to build a classifier for user dialect identification as part of MADAR (Multi-Arabic Dialect Applications and Resources) shared subtask-2 (Bouamor et al., 2018) (Bouamor et al., 2019). The task of user dialect identification can be seen as a text classification problem, where we predict the probability of a dialect given a sequence of words and other features provided by the task organizers. Besides reporting the results from different models, we show how the provided dataset for the task is not straightforward and requires additional analysis, feature engineering, and post-processing techniques.

In the next sections, we describe the methods followed to achieve our best model. Section 2 lists previous work done, Section 3 analyses the dataset, while Section 4 describes models and different approaches. Section 5 compares and discusses empirical results and finally conclude in Section 6.

2 Related Work

Recent work in the Arabic language tackles the task of dialect identification. Fine-grained dialect identification models proposed by Salameh et al. (2018) to classify 26 specific Arabic dialects with an emphasis on feature extraction. They trained multiple models using Multinomial Naive Bayes (MNB) to achieve a Macro-Averaged F1-score of 67.9% for their best model.

In addition to traditional models, deep learning methods tackle the same problem. The research proposed by Elaraby and Abdul-Mageed (2018), shows an enhancement in accuracy when compared to machine learning methods. In Huang (2015), they used weakly supervised data or distance supervision techniques. They crawled data from Facebook posts combined with a labeled dataset to increase the accuracy of dialect identification by 0.5%.

In this paper, we build on top of methods from Salameh et al. (2018) and Elaraby and Abdul-

Proceedings of the Fourth Arabic Natural Language Processing Workshop, pages 239–243
Florence, Italy, August 1, 2019.

	Train	Dev	Test
Available	195227	26750	43918
Unavailable	22365	3119	6044
Total	217592	29869	49962
Retweet	135388	17612	29868

Table 1: Distribution of the Train, Dev and Test sets used in our experiments.

Mageed (2018), by exploring machine and deep learning models to tackle the problem of fine-grained Arabic dialect identification.

3 Dataset

The dataset used in this work represents information about tweets posted from the Arabic region, where each tweet is associated with its dialect label (Bouamor et al., 2018) (Bouamor et al., 2019). This dataset is collected from 21 countries which are Algeria, Bahrain, Djibouti, Egypt, Iraq, Jordan, Kuwait, Lebanon, Libya, Mauritania, Morocco, Oman, Palestine, Qatar, Saudi_Arabia, Somalia, Sudan, Syria, Tunisia, United_Arab_Emirates, Yemen.

As shown in Figure 1, the distribution of tweets among countries is unbalanced. Around 35% of the tweets belong to Saudi Arabia, where only 0.08% belong to Djibouti.

The dataset contains 6 features for each user; username of the Twitter user, tweet ID, the language of the tweet as automatically detected by Twitter, a probability scores of 25 city dialects and MSA (Modern Standard Arabic) for each tweet obtained by running the best model described in (Salameh et al., 2018) and most importantly the tweet text.

Each user has at most 100 tweets, labeled with the same dialect, and exists in one set. For example, if a user is listed in the training set then that user will not exist in development nor test set. Moreover, the maximum length of each tweet is 280 characters including spaces, URLs, hashtags, and mentions which makes it challenging to identify the dialects automatically (Twitter).

Another challenge of the dataset is that around 61% of the tweets are retweets, as shown in Table 1. This means that the majority of the tweets are a re-post of other Twitter users. For example, the

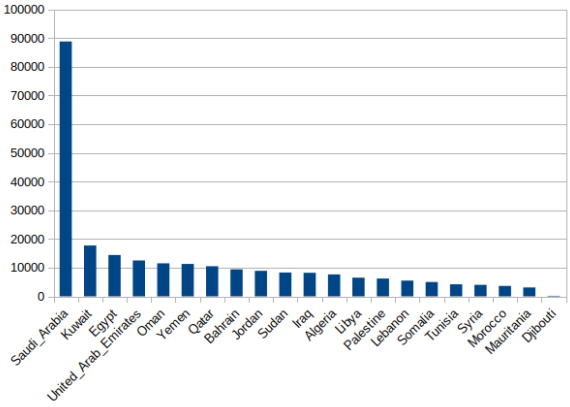

Figure 1: The distribution of 21 Arabic dialects in MADAR Twitter corpus

tweet "RT @Bedoon120: المخرج عاوز كدا https://t.co/sIKqXCUSAn for the user @abushooooq8 is an Egyptian tweet but it has a label of Kuwait because the user who retweeted is Kuwaiti (i.e. @abushooooq8), where the original author is Egyptian (i.e. @Bedoon120).

Table 1 shows the distribution of available and unavailable data across different sets. It is also worth mentioning that around 10% of the data is missing; some tweets are not accessible because they were deleted by the author or owner account was deactivated.

4 Models

In this section, we explain our feature extraction methodology then we go over the various experimented approaches.

4.1 Feature Extraction

As a preprocessing step, normalization of Arabic letters is common when it comes to deal with Arabic text. We adopted the same preprocessing methodology used in (Soliman et al., 2017).

Aravec: A pre-trained word embedding models proposed by (Soliman et al., 2017) for the Arabic language using three different datasets: Twitter tweets, World Wide Web pages, and Wikipedia Arabic articles. Those models were trained using Word2Vec skip-gram and CBOW (Mikolov et al., 2013). In our experiments, we used the 300-dimensional Twitter Skip-gram AraVec model.

fastText: An extension to Word2Vec model proposed by (Bojanowski et al., 2017). The model feeds an input based on sub-words rather than passing the entire words. In our experiments, a model with 300 dimension was trained

on a combination of 5 different datasets: AutoTweet (Almerekhi and Elsayed, 2015), AraTweet (Zaghouani and Charfi, 2018), DART (Alsarsour et al., 2018), PADIC (Parallel Arabic DIalectal Corpus) (Harrat et al., 2014), MADAR shared tasks (Bouamor et al., 2018) (Bouamor et al., 2019).

Tf-idf: It has been proven that Tf-idf is efficient to encode textual information into real-valued vectors that represent the importance of a word to a document (Salameh et al., 2018). One of the drawbacks of Tf-idf vectorized representation of the text is that it looses the information of the word order (i.e., syntactical information). Considering n-grams, for both levels word and characters, reduces the effect of that drawback. Accordingly, unigram and bigram word level Tf-idf vectors were extracted in addition to a character level Tf-idf vectors with n-gram values ranging from 1 to 5.

Features specific to tweets: There are features that are unique to Twitter such as user mentions, (e.g., @abushooooq8) and emojis. It has been found that using username as a feature can help the model understand the user dialect, for instance, it can easily find that the users @7abeeb_ksa, @a_ksa2030 @alanzisaudi have a Saudi_Arabia dialect. Character level unigram Tf-idf has been extracted from each of the mentioned features.

4.2 Classification Methods

We used a range of classification methods starting from traditional machine learning methods into more complicated deep learning techniques.

4.2.1 Machine Learning Approaches

Traditional models include linear and probabilistic classifiers with various feature engineering techniques. We used SVM classifier that implements LinearSVC from the Scikit-learn library (Pedregosa et al., 2011). We used the LinearSVC model to predict the dialect given the tweet text represented as Tf-idf, username features and language model probabilities as formulated in Equation 1:

$$\hat{y} = \underset{d \in \mathcal{D}}{\operatorname{argmax}} P(d|tfidf, tweet\_feat, lm) \quad (1)$$

where $\hat{y}$ is the predicted dialect, D is probability space of all dialects, *tfidf* is the Tf-idf features extracted from a given tweet, *tweet_feat* is the tweet features and *lm* is the language model probabilities.

4.2.2 Deep Learning Approaches

fastText Classification: The word embedding of the words in an input sentence that is fed into a weighting average layer. Then, it is fed to a linear classifier with softmax output layer (Joulin et al., 2017).

SepCNN: Stands for Separable Convolutional Neural Networks (Denk, 2017), and is composed of two consecutive convolutional layers. The first is operating on the spatial dimension and performed on channels separately, while the second layer convolutes over the channel dimension only. Word embedding of the sentences is looked up from AraVec(Soliman et al., 2017). Then, the embedding of each word in the sentence (i.e., the tweet) are passed into a number of SepCNN blocks followed by a max pooling layer.

Word-level LSTM: A traditional deep learning classification method. The word sequence is passed into an AraVec layer to look up word embedding and then fed into a number of LSTM layers. The final word sequence is used as an input to a softmax layer to predict the dialect (Liu et al., 2016).

Char-Level CNN: In this architecture, the input is represented as characters that are converted into 128 character embedding. Those embedding vectors are then passed into a number of one-dimensional convolutional layers. Each convolutional layer is followed by a batch normalization layer to optimize training and to add a regularization factor. The final output is then passed into one hidden layer and followed by a softmax output layer (Zhang et al., 2015).

Char-Level CNN and Word-level LSTM: A combination of the previous two methods. The output of word-level LSTM is concatenated with character-Level CNN before passing both of them into a hidden layer followed by a softmax output layer.

Char-Level CNN and Word-level CNN: In this network words are transformed into word embedding using AraVec, then concatenated with the output of character level CNN. The concatenated result is fed into the LSTM layer, which computes the final output. Then, passed into a hidden layer and a softmax output layer to make the final prediction (Zhang et al., 2017).

Model	Hyperparameters	Dev		Test	
		Acc	F1	Acc	F1
LinearSVC	C=1.0, penalty=L2, loss=hinge, tolerance=0.0001	54.26	38.95	-	-
fastText Classifier	emb_size = 100	48.5	31.12	-	-
SepCNN	filters=128, dense_units=256, emb_size=300, kernel=4, blocks=3	45.46	26.30	-	-
Word LSTM	embed_size=300, dense_units=256, lstm_size=512	44.91	26.89	-	-
Word Bi-LSTM	embed_size=300, dense_units=256, lstm_size=512	45.08	26.36	-	-
Word LSTM with fastText	embed_size=300, dense_units=256, lstm_size=512	50.59	34.65	-	-
Char CNN	dense_units=256, char_embed=128, filters=64, kernel_sizes= [3, 3, 4]	41.55	20.25	-	-
Char CNN and Word LSTM	Combining hyperparameters of Char CNN and Word LSTM models	47.12	30.32	-	-
Char CNN and Word CNN	embed_size=300, char_embed_size=128, char_filters=[6,5,4,3,2]	43.96	29.0	-	-
LinearSVC Combined	C=1.0, penalty=L2, loss=hinge, tolerance=0.0001	77.33	70.43	75.40	65.54
LinearSVC with User Voting	ranges=5, retweet_weight=8, unavailable_weight=1, saudi_weight=1	**81.67**	**71.60**	**76.20**	**69.86**
LinearSVC with Threshold	C=1.0, penalty=L2, loss=hinge, tolerance=0.0001, threshold=75%	80.02	70.72	78.00	67.75

Table 2: Final results on the development set for MADAR shared subtask-2

5 Results and Discussion

Two types of experiments were conducted to evaluate our models. At first, each tweet was treated independently with its corresponding label in the training and testing stages without grouping tweets for each user. All our experiments on MADAR shared subtask-2 were evaluated using the Macro-Averaged F1-score. Table 2 shows the accuracy and Macro-Averaged F1-score of the LinearSVC model. LinearSVC outperformed other traditional machine learning models hence we discarded reporting their results. On the other hand, deep learning models are known to generalize better on a large dataset, but unexpectedly it under-performed machine learning models.

The second type of experiments were done by grouping predictions per user. The unifying approach was done by either combining all tweets together in one document per user or by applying voting per tweet. In the former, we applied LinearSVC on the combined data with averaging the language model scores for all the tweets per user. This model achieved results of 77.33% accuracy and 70.43% Macro-Averaged F1-score. In the latter, we took the output of the first model (Uncombined LinearSVC) and applied two voting techniques.

The first technique was user voting based on dialect weighting. This approach aims to give more emphasis on less frequent dialects by multiplying each predicted label with a weight associated for each dialect d_weight. Which is calculated as follows:

$$step = \frac{\sqrt[3]{max\_count} - \sqrt[3]{min\_count}}{5}$$

$$d\_weight = 6 - ceil(\frac{\sqrt[3]{d\_count} - \sqrt[3]{min\_count}}{step})$$

Where max_count is the number of tweets for the largest dialect (i.e., Saudi_Arabia), min_count is the number of tweets for the smallest dialect (i.e., Djibouti), $step$ is a range defined as inverse cubic difference between maximum and minimum dialect counts divided by 5. $dialect\_weight$ is an integer between 1 and 6 that defines dialect weight. Moreover, we found that increasing the weight of a retweet to 6 enhanced the accuracy of the model, and decreasing the weight of <UNAVAILABLE> tweets to 1 had a similar effect. The final user voting model achieved 81.67% accuracy and 71.60% F1-score which is the best model as shown in Table 2

Secondly, the other voting technique is based on majority voting with a penalty on the largest dialect. In this approach, we took the most frequent label from user tweets as the final label for that user. We impose selecting Saudi_Arabia only if 75% of the predictions were Saudi_Arabia for a given user. This approach achieved 80.02% accuracy and 71.84% Macro-Averaged F1-score.

6 Conclusion

This paper describes various methods applied on MADAR shared subtask-2 to predict an Arabic dialect from a set of given tweets, username, and other features. Experimental results show that LinearSVC was the most powerful prediction model, achieving the best Macro-Averaged F1-score than other machine learning models and deep learning ones. Despite the fact that there was a substantial amount of unavailable tweets in our dataset, yet we were able to achieve a relatively high F1-score of 71.60% on the development set and 69.86% on the test set, ranking second in the competition.

References

Hind Almerekhi and Tamer Elsayed. 2015. Detecting automatically-generated arabic tweets. In *AIRS*, pages 123–134. Springer.

Israa Alsarsour, Esraa Mohamed, Reem Suwaileh, and Tamer Elsayed. 2018. Dart: A large dataset of dialectal arabic tweets. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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 Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Houda Bouamor, Sabit Hassan, and Nizar Habash. 2019. The MADAR Shared Task on Arabic Fine-Grained Dialect Identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop (WANLP19)*, Florence, Italy.

Timo I. Denk. 2017. Text classification with separable convolutional neural networks.

Mohamed Elaraby and Muhammad Abdul-Mageed. 2018. Deep models for arabic dialect identification on benchmarked data. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 263–274.

Salima Harrat, Karima Meftouh, Mourad Abbas, and Kamel Smaili. 2014. Building resources for algerian arabic dialects. In *15th Annual Conference of the International Communication Association Interspeech*.

Fei Huang. 2015. Improved arabic dialect classification with social media data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2118–2126.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Mohammad Salameh, Houda Bouamor, and Nizar Habash. 2018. Fine-grained Arabic dialect identification. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1332–1344, Santa Fe, New Mexico, USA.

Abu Bakr Soliman, Kareem Eissa, and Samhaa R El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265.

Twitter. Api reference index twitter developers.

Wajdi Zaghouani and Anis Charfi. 2018. Arap-tweet: A large multi-dialect twitter corpus for gender, age and language variety identification. *arXiv preprint arXiv:1808.07674*.

Omar F Zaidan and Chris Callison-Burch. 2014. Arabic dialect identification. *Computational Linguistics*, 40(1):171–202.

Shiwei Zhang, Xiuzhen Zhang, and Jeffrey Chan. 2017. A word-character convolutional neural network for language-agnostic twitter sentiment analysis. In *Proceedings of the 22nd Australasian Document Computing Symposium*, page 12. ACM.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.