Voting Classifier vs Deep learning method in Arabic Dialect Identification

Dhaou Ghoul
STIH Lab, Sorbonne University
dhaou.ghoul@sorbonne-universite.fr

Gael Lejeune
STIH Lab, Sorbonne University
gael.lejeune@sorbonne-universite.fr

Abstract

In this paper, we present three methods developed by the SORBONNE Team for the NADI shared task on Arabic Dialect Identification for tweets. The first and the second method use respectively a machine learning model based on a Voting Classifier with words and character level features and a deep learning model at the word level. The third method uses only character-level features. We explored different text representation such as TF-IDF (first model) and word embeddings (second model). The Voting Classifier was the most powerful prediction model, achieving the best macro-average F1 score of 18.8% and an accuracy of 36.54% on the official test. Our model ranked 9 on the challenge and in conclusion we propose some ideas to improve its results.

1 Introduction

The Arabic language is one of the most widely spoken language in the world, currently considered as the fifth language (Chung, 2008) with more than 330 million Arabic speakers. It is the official language of more than 22 countries. In its written form, commonly referred as Literary Arabic, it usually is divided into two categories: Classical Arabic and Modern Standard Arabic (MSA). However, in their daily life Arabic speakers mostly use dialects which are a linguistic variant of classical Arabic with their own features, varying with respect to the country or the region. One can say that MSA is only used for written and official communication while dialects are used for oral communication as well as for many device mediated communication forms: email, SMS, chat or blogs. Therefore, Arabic dialect identification (DID) has become a very important data preparation step that attracts many attention in NLP research. Indeed, the knowledge about the dialect of an input text is useful in several NLP tasks such as sentiment analysis (Al-Twairesh et al., 2016), machine translation or document classification. This paper describes our submission to the NADI shared task on Arabic fine-grained dialect identification cover a total of 100 provinces from all 21 Arab countries (Abdul-Mageed et al., 2020). Two subtasks were proposed:

- **Subtask 1:** Country-level dialect identification: 21,000 tweets covering 21 Arab countries.
- **Subtask 2:** Province-level dialect identification: the same tweets but with province label.

Our submission is dealing with the first subtask. In Section 2, we describe the dataset and in Section 3 we present our three methods. In Section 4, we detail our results and give future directions in Section 5.

2 Dataset

Arabic has a widely varying collection of dialects. Many of these dialects remain understudied due to rarity of resources. The dataset used in this work is made up of a collection of tweets shared by users in different Arab countries. 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. The dataset for the shared task contains 25,957 tweets divided in training, development and test set as shown in Table 1.

The distribution of tweets among countries is very unbalanced. The Egyptian dialect is best represented in te dataset as shown in Figure 1. Around 21% (5543 tweets) of the tweets belong to Egypt while only 0.8% belong to Bahrain (218 tweets) and Djibouti (220 tweets).

Proceedings of the Fifth Arabic Natural Language Processing Workshop, pages 243–249
Barcelona, Spain (Online), December 12, 2020

Datasets	Train	Dev	Test
# lines	21,000	4,957	5,000
# words	269,644	60,467	64,211
# characters	2,798,691	633,134	667,597

Table 1: Size of the Train, Dev and Test sets

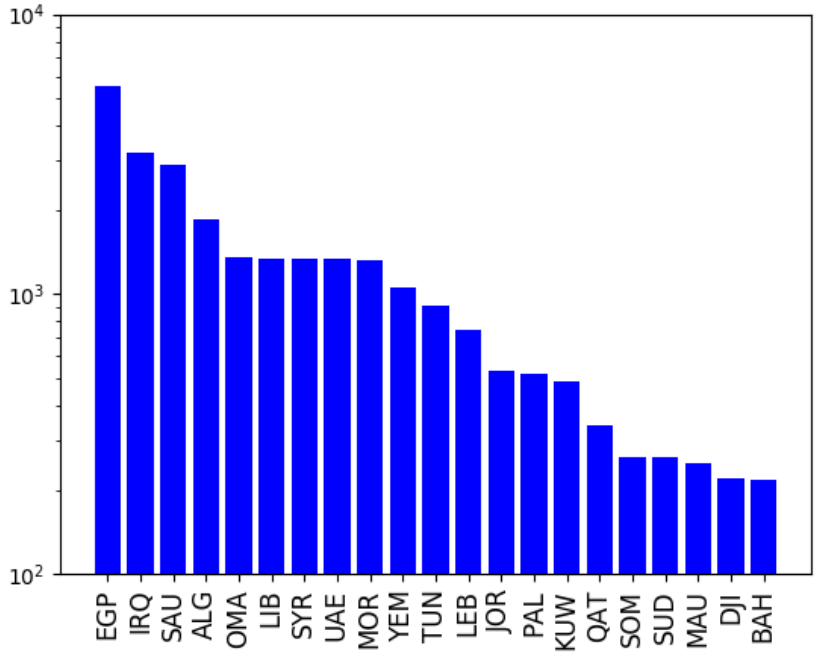

Figure 1: The distribution of the 21 Arabic dialects in NADI Twitter dataset (logarithmic scale)

3 Data Preparation and Classification Methods

3.1 Pre-processing

It is intended to help the classifiers to generalize and to remove potential biases. Our pre-processing is limited to a few steps of cleaning up, applied to each tweet. After several tests, we have kept this cleaning process in the following order: removing punctuation signs, digits, non Arabic words, repeated characters and finally emoticons.

Often, pre-processing also involves removing Arabic stop-words, diacritics and normalizing (Ayedh et al., 2016). It is known that for language identification tasks stop-words are very useful (McNamee, 2005) but we also noticed that removing diacritics and normalizing text reduces the identification as well.

3.2 Models

3.2.1 Run1: Voting Classifier

Our first model is based on a traditional machine learning approach. To build and train the model, we use *FeatureUnion* in SCIKIT-LEARN (Pedregosa et al., 2011) which allows to combine easily different n-gram representations at the word level and the character level as shown in Figure 2. To train this model, we concatenate three vectors with the following features (weigthed with TF-IDF): word n-grams (1 to 5-grams), character n-grams (1 to 4-grams) and character n-grams (1 to 5-grams) with word boundaries explicitly marked with a space.

We use a set of classifiers based on a voting technique using the SCIKIT LEARN implementation of *VotingClassifier* in order to build an ensemble voting classifier with hard voting, where it uses predicted

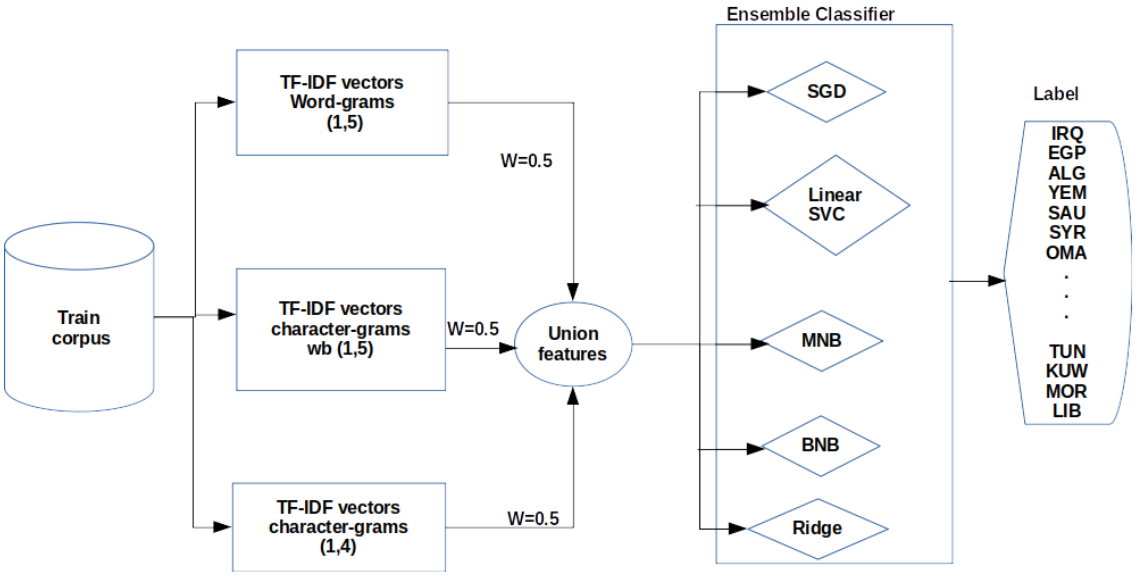

Figure 2: Model of Run1

class labels for majority rule voting. This ensemble is a combination of the following classfiers:

- SGDClassifier with `alpha` = 0.00001 and `penalty` = 'l2'
- LinearSVC with `penalty` = 'l2' and Tolerance for stopping set to 0.001
- Multinomial Naive Bayes with `alpha` = 0.01
- Bernoulli Naive Bayes with `alpha` = 0.01
- Ridge Classifier with `alpha` = 1

The results for the dev and test set with pre-processing are reported in Table 2.

Trained on	Tested on	Precision	Recall	Accuracy	Macro avg F-score
Train Set	Dev Set	22.82	17.02	36.33	17.69
Train Set	Test Set	22	15.9	31.9	16.06
Train+Dev	Test Set	**24.87**	**18.05**	**36.54**	**18.8**

Table 2: Results with pre-processing for Run1 with different training configurations

3.2.2 Run2: Deep Learning approach

To build the deep learning model for Run2, we used two pre-trained word embedding models:

- **Aravec:** proposed by *Soliman et al*, it is composed of three models trained trained with Word2Vec skip-gram and CBOW (Mikolov et al., 2013) on three datasets in Arabic: tweets, web pages and Wikipedia articles (Soliman et al., 2017). We used the 100-dimensional Twitter N-Grams model.
- **Unlabelled-tweetsVec:** Word2Vec skip-gram (300-dimensions) trained on NADI unlabelled tweets

In addition to this, we added features specific to tweets in the input layer of our neural network. When analyzing the train set, we found that there some features where useful to differentiate the dialects : user mentions and words specific to Twitter data. For instance, people tagging the user *@Mowafaglibya* mostly use Lybian dialect, so here the user mention is decisive. Some words as "شنو"(chnowa), "برشَا" (barcha) to identify the Tunisia dialect. These words are specific for the Tunisian dialect, they respectively mean 'what' and 'many'.

The deep neural network developed for this run is composed as follows: the input layer, an embedding layer, two LSTM layers and two Dense layers. To prevent over-fitting we add a dropout layer after each LSTM layer and the first dense layer. We use two word embeddings representation : AraVec and our own model trained on 10M unlabelled tweets. The final output is passed into one hidden layer and followed by a softmax output layer. To train these two models, we used the "NADI" train corpus with 10 epochs and evaluate it by "NADI" development corpus and. The results are reported in Table 3. We can see that, contrary to Run1, this model suffers from over-fitting when trained on both train and dev set.

Word Embedding	Trained on	Tested on	Precision	Recall	Macro avg F-score	Accuracy
AraVec	Train Set	Dev Set	21.99	**16.24**	**16.68**	**35**
	Train Set	Test Set	**22.54**	16.13	16.18	32.56
	Train+Dev	Test Set	20.4	15.03	14.94	33.7
Unlabled-tweetsVec	Train Set	Dev Set	14.92	14.26	14.2	29.03
	Train Set	Test Set	13.98	14.76	13.96	27.98
	Train+Dev	Test Set	17.22	15.28	15.06	29.94

Table 3: Results with pre-processing for Run2

3.2.3 Run3: character n-grams

This run uses MICHAEL the method developed for the 2019 MADAR challenge (Ghoul and Lejeune, 2019), this method does not use any pre-processing in order to act as a simple baseline. It is a character n-gram model with various range of n-gram size. The rationale behind this kind of model is to get a simplified representation without having to tackle the problem of tokenization, which is particularly difficult for languages like Arabic. The best configuration for this baseline was a Logistic Regression classifier with character 5-grams as features. For this baseline, the dataset has been used as it is without any pre-processing.

4 Results and Discussion

The results obtained by our different models for the test set with and without pre-processing are presented in table 4. In reality, there are similar dialects or with a minor difference. In other words, a short sentence can belong to several dialects because it is constructed with a reduced number of words. That is why, it is also impossible for Arabic speakers to detect the dialect from a very short sentence with 100% accuracy. Before analyzing our results, we wanted to briefly check for annotation errors. We focused on the 750 tweets annotated as Tunisian, divided into four provinces (Sousse: 212 tweets, Mahdia: 212 tweets, Ariana: 212 tweets and Kairaoun: 114 tweets). After this verification, we found out that the 212 tweets labeled as Mahdia are wrongly labelled. The majority of them belong to the Libyan dialect and the rest are Sudan tweets. For instance, this tweet (Train-4235): "تي متزعلش بكل تبي نشريلك نيتفليكس" (Don't get angry, do you want me to buy you a netflix) is annotated as Tunisian (region of Mahdia) but is Libyan. Our verification was peformed by consulting the user profile in some cases and from the lexicon used in the tweets in other cases. The last example contains words from the Libyan lexicon like "بكل" and "تبي". The imbalances in the dataset obviously makes the task harder. The imbalance ratio, defined as the ratio of the number of instances in the majority class to the number of examples in the minority class (ratio=0.047 and ratio=0.007). The Egyptian dialect is more represented with 5543 tweets (Train+Dev set). On the other hand, the Mauritania, Somalia, Sudan, Bahrain and Djibouti dialects are less represented with 210 tweets and the Qatari dialect with 234 tweets. All this shows that this shared task was very difficult.

There are dialects which are not easy to detect even by an Arabic speakers as they do not contain any clue words, like Mauritania and Djibouti. These tweets are often written in standard Arabic. Figure 3 shows that our models are not able to correctly classify these tweets. Bahrain, Djibouti dialects were the most difficult to detect : we had no True Positive (TP).

Pre-processing	Features	Trained on	Precision	Recall	Accuracy	Macro avg F-score
With pre-processing	Union-features: TF-IDF	Train set	22	15.9	31.9	16.06
	Run1	Train+Dev set	24.87	**18.05**	36.54	**18.8**
	Embedding: AraVec	Train set	22.54	16.13	32.56	16.18
	Run2	Train+Dev set	20.4	15.03	33.7	14.94
Without pre-processing	Union-features: TF-IDF	Train set	25.10	17.22	36.02	17.49
	Run1	Train+Dev set	**26.72**	17.83	**36.56**	18.6
	Embedding: AraVec	Train set	18.78	14.69	32.6	14.89
	Run2	Train+Dev set	19.38	14.28	33.8	14.28
	Character 5-Grams	Train set	19.13	14.24	32.38	14.2
	Run3	Train+Dev set	18.56	14.81	32.7	14.59

Table 4: Results on the test set, models with and without pre-processing

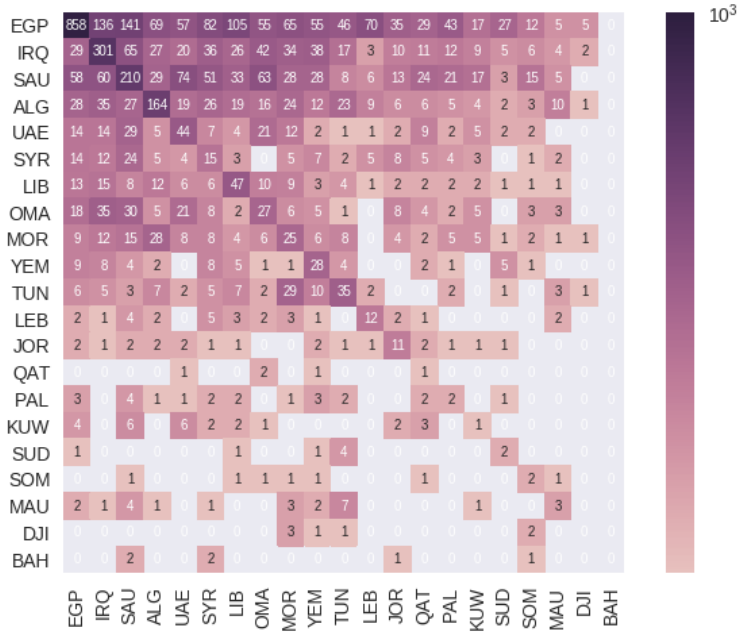

Figure 3: Confusion Matrix for Run1 (Voting Classifier) on the dev set

The confusion matrix in Figure 3 shows the numbers of actual and predicate labels for each dialect. There are some similar pairs of dialects for which the system made a lot of errors : (Iraq, Egypt) and (Saudi-Arabia,Egypt). We can also see on the confusion matrix that 70 tweets from Lebanon were classified as Egyptian. We manually checked the annotation of these 70 tweets: 63 of them are in fact from Egypt and as such are not classification errors. For exemple, "بس مش معَايَا انَا" (But, not with me) is annotated as Lebanon dialect but it is in reality Egypt dialect (after our verification). Our model predict this tweet as an Egypt dialect because it contains words belongs to the Egyptian lexicon. The table 5 presents some examples of tweets from the Train and Dev set and their transliteration. These examples shows the annoataion errors of the Train + Dev set.

For the evaluation purpose, we use the Macro averaged F-score which is retained as the official metric by the organizers of NADI shared task. In all the experiments, the best model is obtained for hard voting classifier with the previously mentioned algorithms (SGDClassifier, Linear SVC, MNB, BNB and RidgeClassifier) with Macro avg F-score of **18.8%** on the test set (line 3 in table 2). From the classification report which is produced from the dev set in Table 6, we find that some dialects were easier to detect than others, for example, the dialects Egypt, Algeria and Iraq with a lower score f respectively 0.58 , 0.41 and 0.46 compared to others such as: Bahrain (0.0), Djibouti (0.0), Kuwait (0.02) and Qatar (0.02).

Id-tweet	Tweets	Label Before verification	Label After verification
Train-640	مَاهي السنه فعلًا مفروض تختم بكدَا mAhy Alsnh fElA mfrwD txtm bkdA	Lebanon	Egypt
Train-4280	لولَا طولَى كنت لبسته ع طول lwlA Twly knt lbsth E Twl	Lebanon	Egypt
Train-4842	ليلة عمَانية حمرَاء بَأدن الَله lylp EmAnyp HmrA' b¡*n Allh	Lebanon	Oman
Train-11283	صديقي وزميلي ف الَأبتدَائِي مش فَاضي بكل Sdyqy w zmyly f Al¡btdAŷ m$ fADy bkl	Tunisia	Lybia
Dev-2672	بعيد الشر ي ستي bEyd Al$r y sty	Lebanon	Egypt
Dev-3290	هّهّهّهّه اعمل ايه انتي الي مش فَاهمه hhhhh AEml Ayh Anty Al y m$ fAhmh	Lebanon	Egypt
Dev-4046	دَا طلع ف نَاس معندهَاش دم فعلًا dA TlE f nAs mEndhA$ dm fElAF	Lebanon	Egypt

Table 5: Examples of annotation errors from Train and Dev Set

	EGP	IRQ	SAU	ALG	UAE	SYR	LIB	OMA	MOR	YEM	TUN	LEB
Precision	0.45	0.43	0.28	0.36	0.34	0.24	0.12	0.17	0.14	0.35	0.29	0.29
Recall	0.80	0.49	0.36	0.46	0.19	0.16	0.05	0.10	0.10	0.15	0.21	0.11
F1-score	0.58	0.46	0.32	0.41	0.24	0.19	0.07	0.12	0.12	0.21	0.24	0.16
Support	1070	636	579	359	265	265	265	249	249	206	164	110
	JOR	QAT	PAL	KUW	SUD	SOM	MAU	DJI	BAH	**Macro**	**Micro**	**Acc.**
Precision	0.35	0.14	0.08	0.03	0.22	0.29	0.12	0.00	0.00	0.22	0.31	0.36
Recall	0.11	0.01	0.02	0.01	0.04	0.04	0.07	0.00	0.00	0.22	0.36	
F1-score	0.16	0.02	0.03	0.02	0.07	0.07	0.09	0.00	0.00	0.17	0.32	
Support	104	104	102	70	51	51	40	10	8	4957	4957	

Table 6: Detailed classification report for the Voting Classifier on the Dev set

Finally, we think that the class imbalance in our data set had a direct influence on the prediction of the less presented class in train data with a TP rate close to zero and FN rate close to 100% as shown in the Figure 3 (”Sudan”, ”Somalia”, ”Qatar”, etc).

5 Conclusion

In this paper, we described our system submitted to NADI shared task on country level dialect identification from Twitter data. We presented three methods for this shared task. Experimental results show that Voting Classifier was the most powerful prediction model, achieving the best macro-average F1 score than other machine learning and deep learning models. Despite the fact that there was a substantial amount of incorrectly annotated tweets in the dataset, we were still able to achieve an F1 score of 17.69% on the development set, 16.06% on the test set and 18.8% on the test set trained on the train and the development set. It would be interesting to correct the errors in training data in order to see the impact on the results. Another perspective would be to evaluate our model with other tokenizers like ”MADAMIRA” (Pasha et al., 2014) and to perform a deeper analysis of classification errors.

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 (WANLP 2020)*, Barcelona, Spain.

Nora Al-Twairesh, Hend Al-Khalifa, and Abdulmalik AlSalman. 2016. AraSenTi: Large-scale twitter-specific Arabic sentiment lexicons. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 697–705, Berlin, Germany, August. Association for Computational Linguistics.

Abdullah Ayedh, Guanzheng TAN, Khaled Alwesabi, and Hamdi Rajeh. 2016. The effect of preprocessing on arabic document categorization. *Algorithms*, 9:27, 04.

Wingyan Chung. 2008. Web searching in a multilingual world. *Commun. ACM*, 51(5):32–40, May.

Dhaou Ghoul and Gaël Lejeune. 2019. MICHAEL: Mining character-level patterns for Arabic dialect identification (MADAR challenge). In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 229–233, Florence, Italy, August. Association for Computational Linguistics.

Paul McNamee. 2005. Language identification: A solved problem suitable for undergraduate instruction. *J. Comput. Sci. Coll.*, 20(3):94–101, February.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1094–1101, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

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.

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. Arabic Computational Linguistics.