Arabic Dialect Identification based on Weighted Concatenation of TF-IDF Features

Mohamed Lichouri, Mourad Abbas, Khaled Lounnas, Besma Benaziz, Aicha Zitouni
Computational Linguistics Department / CRSTDLA, Algiers, Algeria
{m.lichouri,m.abbas,k.lounnas,b.benaziz,a.zitouni}@crstdla.dz

Abstract

In this paper, we analyze the impact of weighted concatenation of TF-IDF features for the Arabic Dialect Identification task. This study is performed for two subtasks: subtask 1.1 (country-level MSA) and subtask 1.2 (country-level DA) identification. The classifiers supporting our comparative study are Linear Support Vector Classification (LSVC), Linear Regression (LR), Perceptron, Stochastic Gradient Descent (SGD), Passive Aggressive (PA), Complement Naive Bayes (CNB), MutliLayer Perceptron (MLP), and RidgeClassifier. In the evaluation phase, our system gives F_1 scores of 14.87% and 21.49%, for country-level MSA and DA identification respectively, which is very close to the average F_1 scores achieved by the submitted systems and recorded for both subtasks (18.70% and 24.23%).

1 Introduction

Nowadays, the study of Arabic dialects has become the focus of many researches, and many workshops have been dedicated for this interesting topic. One of the noteworthy works are those featured in MADAR 2019 (Bouamor et al., 2019), we can mention (Ragab et al., 2019; Abbas et al., 2019; Přibáň and Taylor, 2019) where authors used machine learning techniques combined with resemble classifier or major voting rules to enhance the performance of the used models. The MADAR 2019 has been followed by NADI 2020 in which more dialects from 100 Arab provinces have been covered (Abdul-Mageed et al., 2020). In Talafha et al. (2020); El Mekki et al. (2020); Gaanoun and Benelallam (2020), authors adopted the combination of ngrams and TF-IDF features. An analysis of the impact of preprocessing and n-grams on automatic classification of Tweets that mention medications has been presented in Lichouri and Abbas (2020b). In (Lichouri et al., 2020), the authors studied the effect of TF-IDF features and morphological process on profiling fake news spreaders on Twitter. In some works, because of the imbalanced nature of the used dataset, there is a need to apply sampling techniques: upsampling the minority class as presented in Beltagy et al. (2020) using the Scikit-Learn library (Pedregosa et al., 2011), or the use of oversampling techniques from imblearn toolbox (Lemaître et al., 2017) like: Random Over-Sampler (ROS), Synthetic Minority Oversampling Technique (SMOTE) and Adaptive Synthetic (ADASYN) as described in Lichouri and Abbas (2020a). In this work, we combined multiple classifiers, using a combination of ngrams and TF-IDF as features (Lichouri et al., 2018) with the same concept as suggested in Abu Kwaik and Saad (2019); Touileb (2020) by using a weighted concatenation of these features. We also readapted the oversampling approach proposed in Lichouri and Abbas (2020a), in addition to the morphological preprocessing steps that we applied like stemming, lemmatization and pos tagging. This paper is structured as follows: In section 2, a description of the used dataset is presented. The applied cleaning steps and preprocessing are explained in section 3. In section 4, we present the proposed approach and discuss the findings. Finally, we conclude in section 5.

2 Description of the Dataset

The used dataset in this shared task of NADI 2021 (Abdul-Mageed et al., 2021) is an extension of the previous shared task (Abdul-Mageed et al., 2020). It is composed of two parts, the first one includes tweets written in Modern Standard Arabic (MSA), and the second one in the local Arabic Dialects (DA). These two subsets (MSA and DA) include tweets from 21 Arab countries and 100 provinces.

	MSA subset				DA subset			
	Train	Dev	Test	Total	Train	Dev	Test	Total
# sentences	21,000	5,000	5,000	31,000	21,000	5,000	5,000	31,000
# words	233.4k	65.4k	63.1k	408.2k	168.3k	45.7k	44.7k	258.7k
Max # word per sentence	49	58	57	-	51	58	59	-
Min # word per sentence	1	1	1	-	1	1	1	-
Max # char per sentence	592	382	285	-	268	279	283	-
Min # char per sentence	1	2	1	-	1	2	2	-

Table 1: Dataset statistics after applying some pre-processing steps

We divided these subsets into three parts: train, dev and test, for which we addressed some statistics in Table 1, after applying a number of pre-processing steps. As can be noticed from Table 1, the minimum number of words and characters per sentence in the training and development sets is 1. This information is useful to determine the number "n" of grams to be selected in the features extraction phase.

3 Data Cleaning and Preprocessing

In most NLP tasks, dealing with tweets necessitates cleaning before analysis and processing. Hence we applied some simple surface cleaning steps including Arabic Letter Normalizer (ALN) and the removal of: punctuation (RP), emojis (i.e., emoticons, symbols & pictographs, transport & map symbols and flags (iOS)) (RE), stop words (RSW), Arabic diacritics (RAD), Latin letter (RLL) and words (RLW), repeated words (RRW) and chars (RRC). We used also three morphological preprocessing steps: WordNetLemmatizer (Lem), ISRI Arabic Stemmer (Stem) and PosTagger (PosTag) of NLTK[1].

After a series of experiments in order to investigate the impact of all the possible combinations of the different aforementioned cleaning processes, we kept the best configuration which is composed of emojis removal, diacritics removal, repeated words removal, in addition to the stemming process. Furthermore, we selected the features which yielded the best performance after testing a huge number of n-grams combinations (200 experiments). These features have been transformed using the TF-IDF vectorizer and concatenated with (see Table 2) and without (see Table 3) a weighted union function.

4 Proposed system

It should be noted from Figure 1, that we first combined all the aforementioned preprocessing steps to generate all the possible textual presentations, after that we applied a second combination between n-grams (n=1 to 10) and tokenizer (word, char, char with boundary, union of the three) to generate all the possible vector features. Then we fed these features to the best oversampling procedures (i.e., ADASYN) which were applied before training our three best classifiers (LSVC, RDG, SGD). Finally we used an resemble classifier (i.e., majority voting rule) which takes as input the three predictions and applies a majority voting rule to predict the dialect. We divided our work into three setups. In the first setup, we conducted an empirical comparison of some classifiers (Pedregosa et al., 2011), namely: Linear Support Vector Classification (LSVC), Linear Regression (LR), Perceptron, Stochastic Gradient Descent (SGD), Passive Aggressive (PA), Complement Naive Bayes (CNB), MutliLayer Perceptron (MLP), and RidgeClassifier. We used these classifiers in their default setup except for the SGD classifier for which we examined the performance in function of runs number. In the first setup, we used three types of features independently, it is about the TF-IDF features of ngrams for words, chars and char_wb. Using different values of n (ngrams) resulted in diverse feature vectors using the eight aforementioned classifiers. We then ranked the best "features+classifiers" combinations according to the obtained F_1 scores, in order to use them in the following setups.

The second setup is based on the first one, since we concatenate the best "features+classifiers", mentioned above, without weighting. The best performance is obtained by three classifiers, as shown in Table 2.

[1]https://www.nltk.org/index.html

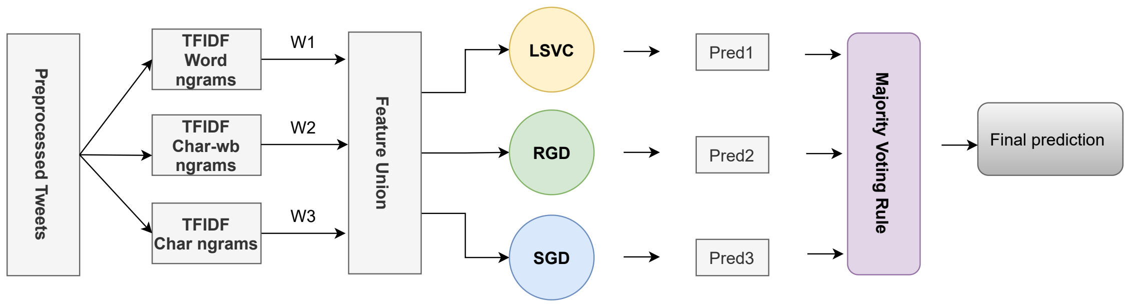

Figure 1: Proposed system based on weighted concatenation approach

	MSA		DA	
Model	Configuration	F_1	Configuration	F_1
SVC	unionVect(1, 5, 2, 10, 2, 8)	14.70	unionVect(1, 3, 1, 4, 2, 4)	21.81
SGD	unionVect(1, 2, 1, 4, 1, 9)	15.28	unionVect(1, 3, 1, 4, 2, 4)	21.85
RDG	unionVect(1, 2, 1, 5, 1, 9)	14.73	unionVect(1, 2, 1, 4, 2, 4)	20.68

Table 2: Obtained results in the dev phase while applying a concatenation without weighting. unionVect takes as inputs different values of n (ngrams)

	MSA		DA	
Model	weights	F_1	weights	F_1
SVC	0.8 0.6 0.5	14.89	0.8 0.4 0.8	21.25
SGD	0.8 0.8 0.8	14.84	0.8 0.5 0.8	18.98
RDG	0.8 0.7 0.8	14.85	0.8 0.5 0.8	19.76
Comb.	0.8 0.7 0.8	14.90	0.8 0.5 0.8	20.09

Table 3: Obtained results in the dev phase while applying a concatenation by weighting the three analyzers (char, char_wb, word). The selected values of n are (1, 5, 1, 7, 1, 9).

According to the results shown in Table 3, the three best classifiers are SVC, SGD and RDG, run with their default parameters. The best results for both MSA and DA are obtained by SGD classifer with an F_1-score of 15.28% and 21.85%, respectively. For the first subtask, the concatenation of: word bigrams (1, 2), char 4-grams (1, 4) and char_wb 9-grams (1, 9) gave the best results. Whereas, in the second subtask, the best performance was achieved using a concatenation of: word 3-grams (1, 3), char 4-grams (1, 4) and char_wb "bigrams, 3-grams and 4-grams" (2, 4).

In the third setup, we applied weighted concatenation of the features used in the second setup. We used multiple values of the weights: w1, w2 and w3 for word, char, and char_wb analyzers, respectively. The best performance is recorded for both MSA and DA subtasks using a majority voting rule on the three classifiers SVC, SGD, and RDG with an F_1-score of 14.90% and 20.09%, respectively. The best performance for the second subtask is achieved using the SVC classifier with an F_1-score of 21.25%. We summarize the results in Table 3.
As a final step, and in order to improve the results achieved in the third setup, we investigated the impact of oversampling approaches (i.e., ROS, SMOTE, ADASYN) to tackle the issue of imbalanced data (see Figure 1). It should be noted that the configurations reported in Table 5 are defined in Table 4.
For the first subtask (MSA), we obtained these performances by applying a set of surface and morphological preprocessing followed by a union of (word 5-grams, char (2,8) grams and char_wb (2,10)) multiplied by a vector weights (1.3, 1.3, 1), in addition to the intern parameters of SGD classifier(l1_ratio=0.25, loss='modified_huber',penalty='elasticnet'). We obtained F_1=16.30% (dev) F_1=14.50% (test).
For the second subtask (DA), we included the ADASYN oversampling technique to the configuration used in the first subtask while changing the preprocessing, ngrams range used by unionVect, vector weight as well as the parameters of

Configs	Preprocess	Union	Weights	Oversampling	SGD
Conf1	RSW, RAD, ALN, RRW, RLL, Stem, Lem	1, 5, 2, 8, 2, 10	1.3, 1.3, 0.9	N/A	l1_ratio=0.25, loss='modified_huber', penalty='elasticnet'
Conf2			1.3, 1.3, 1.0		
Conf3			1.3, 1.3, 1.9		
Conf4	RE, RAD, RRW, Stem	1, 3, 1, 4, 2, 4	1.8, 0.7, 1.8	ratio='minority'	max_iter=2000, shuffle=False
Conf5				ratio='minority', random_state=222	
Conf6				ratio='minority', random_state=333}	

Table 4: Configurations used in the 4th setup

Subtasks	Configuration	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
Subtask 1.1	Dev	15.86	16.28	16.30	N/A		
	Test	14.47	14.87	14.50			
Subtask 1.2	Dev	N/A			22.81	22.94	23.01
	Test				21.08	21.49	21.13

Table 5: Performance (F1) achieved using weighted concatenation and oversampling techniques for both subtasks.

the SGD classifier. We obtained performance equal to 23.01% and 21.13% for development and test phases, respectively.

5 Conclusion

In this paper, we described our two submitted systems for identifying Arabic dialects, using twitter texts collected from 21 Arabic countries. We carried out an empirical comparison study by conducting 200 experiments in which we used several combinations of features, and preprocessing steps as stemming, lemmatization and part of speech tagging. We used a simple classification technique comprising three classifiers: LSVC, RDG and SGD. We addressed the issue of imbalanced data using a set of oversampling methods. Furthermore, We tried to investigate the impact of concatenating TF-IDF features with and without weight vector.

References

Mourad Abbas, Mohamed Lichouri, and Abed Alhakim Freihat. 2019. St madar 2019 shared task: Arabic fine-grained dialect identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 269–273.

Muhammad Abdul-Mageed, Chiyu Zhang, Houda Bouamor, and Nizar Habash. 2020. Nadi 2020: The first nuanced arabic dialect identification shared task. *arXiv preprint arXiv:2010.11334*.

Muhammad Abdul-Mageed, Chiyu Zhang, Abdel-Rahim 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 (WANLP 2021)*.

Kathrein Abu Kwaik and Motaz K Saad. 2019. Arbdialectid at madar shared task 1: Language modelling and ensemble learning for fine grained arabic dialect identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, Proceedings of the Fourth Arabic Natural Language Processing Workshop. Association for Computational Linguistics.

Ahmad Beltagy, Abdelrahman Wael, and Omar ElSherief. 2020. Arabic dialect identification using bert-based domain adaptation. *arXiv preprint arXiv:2011.06977*.

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*, pages 199–207.

Abdellah El Mekki, Ahmed Alami, Hamza Alami, Ahmed Khoumsi, and Ismail Berrada. 2020. Weighted combination of bert and n-gram features for nuanced arabic dialect identification. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 268–274.

Kamel Gaanoun and Imade Benelallam. 2020. Arabic dialect identification: An arabic-bert model with data augmentation and ensembling strategy. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 275–281.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Mohamed Lichouri and Mourad Abbas. 2020a. Simple vs oversampling-based classification methods for fine grained arabic dialect identification in twitter. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 250–256.

Mohamed Lichouri and Mourad Abbas. 2020b. Speechtrans@ smm4h'20: Impact of preprocessing and n-grams on automatic classification of tweets that mention medications. In *Proceedings of the Fifth Social Media Mining for Health Applications Workshop & Shared Task*, pages 118–120.

Mohamed Lichouri, Mourad Abbas, and Besma Benaziz. 2020. Profiling fake news spreaders on twitter based on tfidf features and morphological process.

Mohamed Lichouri, Mourad Abbas, Abed Alhakim Freihat, and Dhiya El Hak Megtouf. 2018. Word-level vs sentence-level language identification: Application to algerian and arabic dialects. *Procedia Computer Science*, 142:246–253.

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.

Pavel Přibáň and Stephen Taylor. 2019. Zcu-nlp at madar 2019: Recognizing arabic dialects. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 208–213.

Ahmad Ragab, Haitham Seelawi, Mostafa Samir, Abdelrahman Mattar, Hesham Al-Bataineh, Mohammad Zaghloul, Ahmad Mustafa, Bashar Talafha, Abed Alhakim Freihat, and Hussein Al-Natsheh. 2019. Mawdoo3 ai at madar shared task: Arabic fine-grained dialect identification with ensemble learning. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 244–248.

Bashar Talafha, Mohammad Ali, Muhy Eddin Za'ter, Haitham Seelawi, Ibraheem Tuffaha, Mostafa Samir, Wael Farhan, and Hussein T Al-Natsheh. 2020. Multi-dialect arabic bert for country-level dialect identification. *arXiv preprint arXiv:2007.05612*.

Samia Touileb. 2020. Ltg-st at nadi shared task 1: Arabic dialect identification using a stacking classifier. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 313–319.