Simple vs Oversampling-based Classification Methods for Fine Grained Arabic Dialect Identification in Twitter

Mohamed Lichouri
Computational Linguistics Department
CRSTDLA, Algiers, Algeria
m.lichouri@crstdla.dz

Mourad Abbas
Computational Linguistics Department
CRSTDLA, Algiers, Algeria
m.abbas@crstdla.dz

Abstract

In this paper, we present a description of our experiments on country-level Arabic dialect identification. A comparison study between a set of classifiers has been carried out. The best results were achieved using the Linear Support Vector Classification (LSVC) model by applying a Random Over Sampling (ROS) process yielding an F1-score of 18.74% in the post-evaluation phase. In the evaluation phase, our best submitted system has achieved an F1-score of 18.27%, very close to the average F1-score (18.80%) obtained for all the submitted systems.

1 Introduction

Fine grained Arabic dialect identification is a very challenging topic due, in one hand, to the rarity of dialectal resources, and in the other hand, to the inter-closeness of Arabic dialects spoken by twenty two countries, as well as the intra-closeness of each sub-dialect spoken in many provinces of a country. One of the most recent scientific events dedicated to Arabic fine grained dialect identification was organized in the framework of the fourth workshop for Arabic natural language processing, MADAR'2019 (Bouamor et al., 2019), where a couple of works have been proposed. Indeed, in Abu Kwaik and Saad (2019), the authors proposed a system based on extracting and applying a feature union process on a set of features (TF-IDF word n-grams, TF-IDF character n-grams, TF-IDF character_with_boundary n-grams and TF-IDF skip n-grams) multiplied by a transformation weight. In fact, they applied empirical experiments to find the best values of "n" (n-grams) as well as the transformation weights. Other features have been added, like sentence length ratio for every sentence in the data. To classify these features, the authors used an ensemble hard voting classifier with three ML algorithms (Linear Support Vector Classifier "LSVC", Multinomial Naive Bayes "MNB" and Bernoulli Naive Bayes "BNB"). They obtained a macro F-score of 67.32%. Based on the work done by (Salameh et al., 2018), (Ragab et al., 2019) have suggested an ensemble of 5 classifiers (MNB, SVC, BNB, k-nearest-neighbours "KNN" and a weak dummy classifier based on prior probabilities of each dialect), in addition to 96 language models on word and char-level trained using KenLM (Heafield, 2011) from Moses, with default parameters. This setup has given an F1-score of 66.7%. In a more refined experiment, (Abdul-Mageed et al., 2019) presented a very large scale dataset covering 319 cities from all 21 Arab countries. They also introduced a hierarchical attention multi-task learning (HA-MTL) approach for dialect identification exploiting the data at the city, state, and country levels. They also evaluated the use of BERT on the three tasks, while comparing it to the MTL approach. More recently, (Abdelali et al., 2020) built a new corpus and called it QADI (18 countries). They experimented the AraBert model (Baly et al., 2020) on both QADI and MADAR corpus (25 cities + MSA). The score they achieved was 60.6% and 29% on QADI and MADAR, respectively.

In this paper, we present our contribution based on our system presented in (Lichouri et al., 2018) and (Abbas et al., 2019) to solve the challenge introduced in the first subtask, i.e. identifying 21 Arabic countries' dialects. In section 2, a description of the used dataset is presented. The applied cleaning steps and preprocessing are explained in section 3. In sections 4 and 5, we exposed the two adopted approaches (both simple and oversampling-based classification), respectively. Finally, the findings and discussion are presented in section 6.

Proceedings of the Fifth Arabic Natural Language Processing Workshop, pages 250–256
Barcelona, Spain (Online), December 12, 2020

2 Description of the Dataset

In this section, we describe the dataset used in our current (i.e., NADI) shared task (Abdul-Mageed et al., 2020). This dataset is collected from 21 Arab countries from the Twitter domain, covering a total of 100 provinces. The dataset is divided to three sets: train, dev and test, for which we addressed, in table 1, some statistics after applying a simple punctuation removal. As can be noticed from table 1,

	Train	Dev	Test	Total
# sentences	21,000	5,000	5,000	31,000
# words	350,191	80,224	84,929	515,344
Max # word per sentence	228	248	249	-
Min # word per sentence	1	1	1	-
Max # char per sentence	488	673	535	-
Min # char per sentence	3	3	1	-

Table 1: NADI dataset statistics

the minimum number of words and characters per sentence in the training and development sets are 1 and 3, respectively. This information was useful to determine the number "n" of grams to be selected in the features extraction phase. In figure 1, we note that the numbers of words for the 21 dialects are differential in training dataset. As an example, we mention the case of Saudi Arabia versus Sudan or Iraq versus Libya. We also note that the ratio of the numbers of words to the number of sentences is more considerable for some dialects as Egyptian, Iraqi and Algeria.

3 Data Cleaning and Preprocessing

Dealing with texts collected from Twitter necessitates a cleaning process before doing analysis and processing. Hence we have applied two simple cleaning steps:

1. Emoji removal: We removed the emoticons, symbols & pictographs, transport & map symbols and flags (iOS).

2. Arabic and Latin punctuation removal: We removed a list of Arabic punctuation symbol that we prepared manually, whereas we used the Latin punctuation symbol list provided by the string library (string.punctuation).

For preprocessing, we explored the following steps: punctuation removal, normalization of Arabic letters, stop words removal, lemmatization, stemming, and part of speech tagging. For the lemmatization process we used Farasa[1] developed by (Abdelali et al., 2016), whereas for stemming and part of speech tagging, we used ISRI Arabic Stemmer and PosTagger of NLTK[2] (Bird et al., 2009). To investigate the impact of these six text functionalities, we combined them in different ways (see tables 2 and 3). We used different combinations of n-grams as features which resulted in 200 experiments. Nevertheless, we kept those which yielded the best performance, namely: 3-grams word, 5-grams word, 3-grams char, 5-grams char, 3-grams char_wb, 5-grams char_wb (see tables 2 and 3). These features have been transformed using the sklearn tfidf vectorizer (Pedregosa et al., 2011) as presented in (Abbas et al., 2019).

4 Simple Classification Model

This classification model is simply based on three linear classifiers, namely: Linear Support Vector Classification (LSVC), Ridge Classifier (RDG) and Stochastic Gradient Descent (SGD). The parameters we used for these classifiers are as follows: Linear SVC (C=0.01, penalty="l1", dual=False), Ridge Classifier (tol=1e-4, solver="sag") and SGD (alpha=.0001, max_iter=100, penalty='l2'). So as to avoid overfitting, we have applied a K-Fold Cross Validation (K=10) by using the Stratified K-Folds cross-validator (Pedregosa et al., 2011).

[1] http://alt.qcri.org/farasa/

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

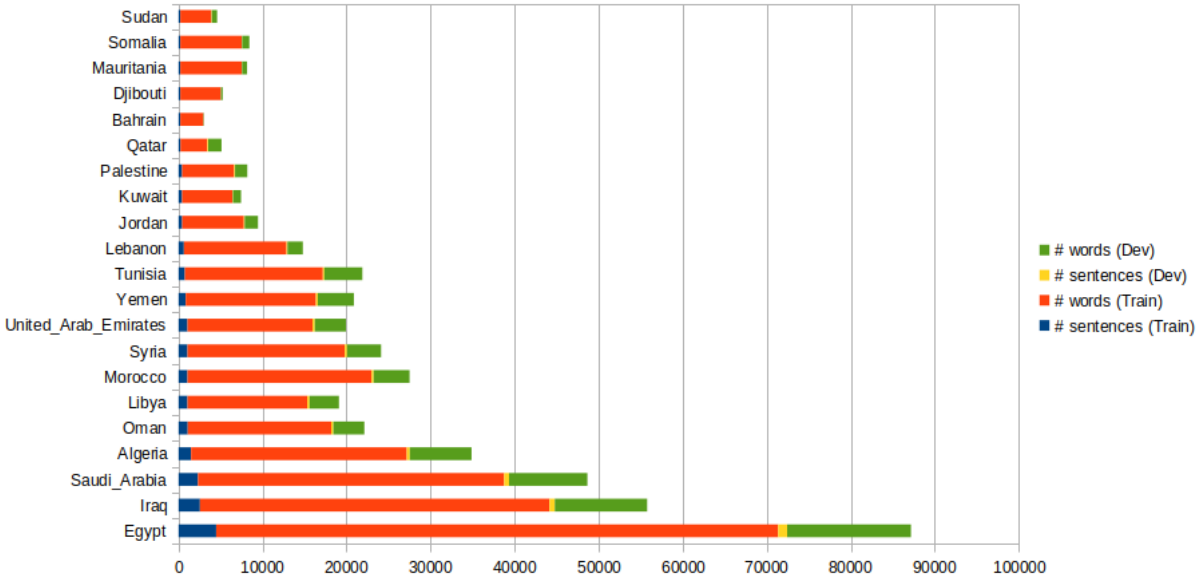

Figure 1: Dataset distribution: # sentences and words per country

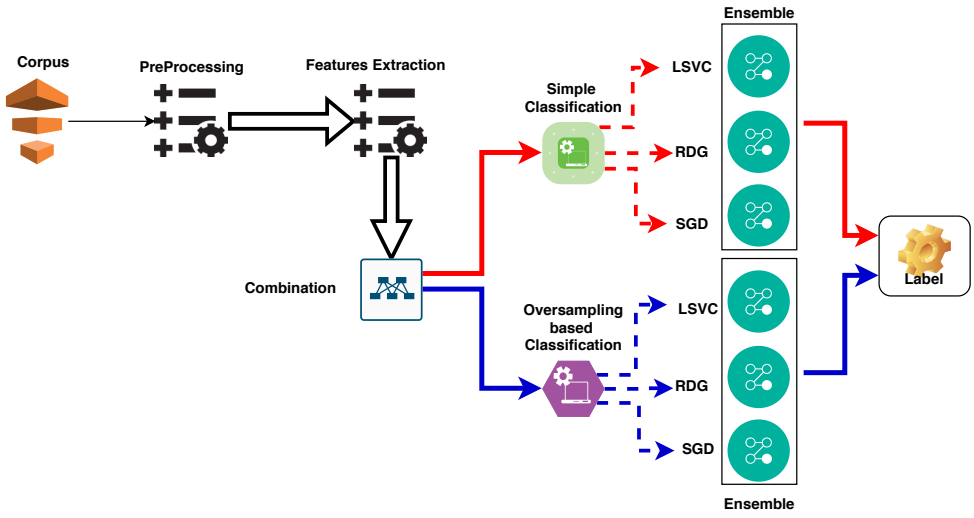

Figure 2: Classification system architecture of Arabic dialects.

5 Oversampling Based Classification

Since the dataset used in this task is imbalanced, as can be noticed clearly in figure 1, we used oversampling techniques to solve this problem (Lemaître et al., 2017). In the following, we briefly describe three different techniques that we used for oversampling:

Random Over-Sampler (ROS): This can be achieved by simply duplicating random examples from the minority class in the training dataset prior to fitting a model. This can balance the class distribution but does not provide any additional information to the model (Brownlee, 2020).

Synthetic Minority Oversampling Technique (SMOTE): It works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (default k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space (Brownlee, 2020).

Adaptive Synthetic (ADASYN): It is based on the idea of adaptively generating minority data samples according to their distributions using K nearest neighbors. The algorithm adaptively updates the distribution and there are no assumptions made for the underlying distribution of the data. The algorithm uses Euclidean distance for KNN Algorithm (Walimbe, 2017).

For these three models, we adopted the default configuration except two parameters which are: ratio='minority' and random_state=777. The full architecture of our system is presented in figure 2. It should be noted that we first combined all the six aforementioned preprocessing steps to generate all the possible textual presentations, after that we applied a second combination between n-grams (n=1,3,5) and tokenizer (word, char, char with boundary, union of the three) to generate all the possible vector features. Then we fed these features to our two proposed approaches: a simple classification model based on three classifiers (LSVC, RDG, SGD) and an oversampling-based classification approach based on the same three classifiers in addition to three oversampling procedures (ROS, SMOTE, ADASYN) which were applied before training. Finally we have used an ensemble classifier which will take as input the three predictions by the two approaches separately and apply a major voting rule to predict the dialect.

Run	Model	Text PreProcessing	Morpho Features	Simple		CV=10	
				Dev	Test	Dev	Test
1	**LSVC**	RP	-	19.01	17.90	18.42	**18.29**
	RDG	RP	-	18.28	17.31	18.39	**18.28**
	SGD	RP	-	18.91	17.95	18.83	18.01
	Ensemble	RP	-	18.99	17.73	**19.23**	**18.46**
2	**LSVC**	RP + Norm	-	19.01	17.90	17.98	18.05
	RDG	RP + Norm	-	18.28	17.31	18.35	17.71
	SGD	RP + Norm	-	**19.55**	**18.27**	18.08	17.31
	Ensemble	RP + Norm	-	19.02	17.63	18.08	17.70
3	**LSVC**	RP + Norm + RSW	-	17.99	17.54	18.63	**18.48**
	RDG	RP + Norm + RSW	-	17.86	17.07	18.08	17.71
	SGD	RP + Norm + RSW	-	18.14	17.04	17.71	17.40
	Ensemble	RP + Norm + RSW	-	17.72	17.09	18.39	18.07
4	**LSVC**	RP + Norm	Stem	19.01	17.90	18.58	17.52
	RDG	RP + Norm	Stem	18.28	17.31	18.61	17.20
	SGD	RP + Norm	Stem	18.61	17.18	18.35	17.67
	Ensemble	RP + Norm	Stem	18.83	17.33	18.49	17.64
5	**LSVC**	RP + Norm	Lem	19.01	17.90	17.76	16.61
	RDG	RP + Norm	Lem	18.28	17.31	16.99	16.34
	SGD	RP + Norm	Lem	**19.36**	**18.24**	16.64	15.51
	Ensemble	RP + Norm	Lem	18.82	17.62	17.26	16.40
6	**LSVC**	RP + Norm	PosTag	19.01	17.90	15.62	15.47
	RDG	RP + Norm	PosTag	18.28	17.31	14.58	13.68
	SGD	RP + Norm	PosTag	18.20	17.39	13.90	13.80
	Ensemble	RP + Norm	PosTag	18.64	17.44	15.34	14.83
7	**LSVC**	RP + Norm	Stem + Lem	19.01	17.90	-	-
	RDG	RP + Norm	Stem + Lem	18.28	17.31	-	-
	SGD	RP + Norm	Stem + Lem	**19.75**	**18.00**	-	-
	Ensemble	RP + Norm	Stem + Lem	18.95	17.64	-	-
8	**LSVC**	RP + Norm + RLL	Stem + Lem	18.67	17.76	-	-
	RDG	RP + Norm + RLL	Stem + Lem	18.67	17.58	-	-
	SGD	RP + Norm + RLL	Stem + Lem	18.72	17.80	-	-
	Ensemble	RP + Norm + RLL	Stem + Lem	18.61	17.50	-	-

Table 2: The obtained results (macro average F1-score) in dev and test set for the simple classification model for the best 8 runs from the 200 run conducted.
RP: Removal of Punctuation, **Norm**: Arabic Letter Normalization, **RSW**: Removal of Stop Words, **RLL**: Removal of Latin Letters, **Stem**: Stemmer, **Lem**: Lemmatizer, **PosTag**: PosTagger.

6 Results and Discussion

After conducting more than 200 empirical experiments, in which we used different combinations of n-grams, we have reported the outputs of our 8 best configurations, as mentioned in tables 2 and 3, performed by the LSVC, RDG and SGD classifiers, in addition to the Ensemble classifier. Note that the oversampling-based method outperformed the simple classification model (F1-score = 18.74%). The impact of applying cross validation procedure k-CV (k=10) is also reported in tables 2 and 3. We should

Run	Model	Text PreProcessing	Morpho Features	OverSamp Methods	CV=10	
					Dev	Test
1	**LSVC**	RP	-	**ROS**	18.26	**18.74**
	RDG	RP	-	**SMOTE**	18.47	**18.27**
	SGD	RP	-	**ADASYN**	17.28	17.66
	Ensemble	RP	-	-	18.30	18.23
2	**LSVC**	RP + Norm	-	**ROS**	18.00	18.04
	RDG	RP + Norm	-	**SMOTE**	18.61	17.65
	SGD	RP + Norm	-	**ADASYN**	17.36	16.94
	Ensemble	RP + Norm	-	-	18.24	17.48
3	**LSVC**	RP + Norm + RSW	-	**ROS**	18.32	**18.60**
	RDG	RP + Norm + RSW	-	**SMOTE**	18.06	17.77
	SGD	RP + Norm + RSW	-	**ADASYN**	16.91	17.23
	Ensemble	RP + Norm + RSW	-	-	18.12	18.25
4	**LSVC**	RP + Norm	Stem	**ROS**	18.56	17.80
	RDG	RP + Norm	Stem	**SMOTE**	18.59	17.33
	SGD	RP + Norm	Stem	**ADASYN**	17.33	16.84
	Ensemble	RP + Norm	Stem	-	18.76	17.37
5	**LSVC**	RP + Norm	Stem	**ROS**	17.84	16.58
	RDG	RP + Norm	Stem	**SMOTE**	17.01	16.02
	SGD	RP + Norm	Stem	**ADASYN**	16.40	15.80
	Ensemble	RP + Norm	Stem	-	17.21	16.62
6	**LSVC**	RP + Norm	PosTag	**ROS**	15.85	15.94
	RDG	RP + Norm	PosTag	**SMOTE**	14.55	13.94
	SGD	RP + Norm	PosTag	**ADASYN**	13.17	13.46
	Ensemble	RP + Norm	PosTag	-	14.82	14.97

Table 3: Macro average F1-score for the oversampling classification approach -best 6 runs-.
RP: Removal of Punctuation, **Norm**: Arabic Letters Normalization, **RSW**: Removal of Stop Words, **RLL**: Removal of Latin Letters, **Stem**: Stemmer, **Lem**: Lemmatizer, **PosTag**: PosTagger.

note that the TF-IDF features used in all the 8 experiments are the union of (5-grams word, 5-grams char and 5-grams char with boundary). We summarize the results describing the best performance achieved with the two proposed approaches " simple classification (Table 2) and oversampling-based classification (Table 3)" where the impact of preprocessing on the different experiments (runs) is clearly noticeable. Let us first introduce the positive impact of punctuation removal and Arabic letter normalization. Indeed, one can see the improvement achieved in the second run, with an F1-score of 19.55% in dev (18.27% in test), while the stop words removal yielded the worst performance (table 2). In the the following runs (4-8), we kept the stop words and we applied stemming, lemmatization and part of speech tagging separately. This didn't lead to an improvement compared to the first three runs. However, using stemming and lemmatization jointly, we obtained better results using the SGD classifier with an F1-score of 19.75% (dev) and 18% (test). Ironically, when we removed the Latin letters in the last run, performance decreased, which shows apparently that the Latin words are useful for identifying Arabic dialects. Furthermore, we mention herein the results reported using the test set that outperformed our best submission system. For the simple classification model, we recorded the following F1-socres: 18.28% (run 1-RDG), 18.29% (run 1-LSVC), 18.46% (run 1-Ensemble) and 18.48% (run 3-LSVC). In the case of oversampling-based approach, using 10-CV, we obtained the following results: 18.27% (run 1-Ensemble), 18.60% (run 3-LSVC) and 18.74% (run 1-LSVC). These results have shown that the 10-CV is practical in the presence of new data (test set).

7 Conclusion

In this paper, we presented a description of our system for identifying Arabic dialects from twitter collected from 21 Arabic countries. We have performed an empirical comparison study where we conducted 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 model comprising three classifiers: LSVC, RDG and SGD, while we tried to resolve the problem of imbalanced data using the

oversampling methods. As expected, using a small and imbalanced dataset didn't help to get high scores, however, this task can be seen as the first try to identify high number of dialects with few resources.

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.

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy 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.

Ahmed Abdelali, Hamdy Mubarak, Younes Samih, Sabit Hassan, and Kareem Darwish. 2020. Arabic dialect identification in the wild. *arXiv preprint arXiv:2005.06557*.

Muhammad Abdul-Mageed, Chiyu Zhang, AbdelRahim Elmadany, Arun Rajendran, and Lyle Ungar. 2019. Dianet: Bert and hierarchical attention multi-task learning of fine-grained dialect. *arXiv preprint arXiv:1910.14243*.

Muhammad Abdul-Mageed, Chiyu Zhang, Houda Bouamor, and Nizar Habash. 2020. The Shared Task on Nuanced Arabic Dialect Identification (NADI). In *Proceedings of the Fifth Arabic Natural Language Processing Workshop (WANLP2020)*, Barcelona, Spain.

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. *ArbDialectID at MADAR Shared Task 1: Language Modelling and Ensemble Learning for Fine Grained Arabic Dialect Identification*, (Proceedings of the Fourth Arabic Natural Language Processing Workshop).

Fady Baly, Hazem Hajj, et al. 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.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

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.

Jason Brownlee. 2020. Smote for imbalanced classification with python. `https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/`. Accessed: 2020-06-23.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197. Association for Computational Linguistics.

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, 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.

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.

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.

Rohit Walimbe. 2017. Handling imbalanced dataset in supervised learning using family of smote algorithm. https://www.datasciencecentral.com/profiles/blogs/handling-imbalanced-data-sets-in-supervised-learning-using-family. Accessed: 2020-06-23.