Simple but *not* Naïve: Fine-Grained Arabic Dialect Identification using only N-Grams

Sohaila Eltanbouly, May Bashendy, and Tamer Elsayed
Qatar University
Doha, Qatar
{se1403101, ma1403845, telsayed}@qu.edu.qa

Abstract

This paper presents the participation of Qatar University team in MADAR shared task, which addresses the problem of sentence-level fine-grained Arabic Dialect Identification over 25 different Arabic dialects in addition to the Modern Standard Arabic. Arabic Dialect Identification is not a trivial task since different dialects share some features, e.g., utilizing the same character set and some vocabularies. We opted to adopt a very simple approach in terms of extracted features and classification models; we only utilize word and character n-grams as features, and Naïve Bayes models as classifiers. Surprisingly, the *simple* approach achieved *non-naïve* performance. The official results, reported on a held-out testing set, show that the dialect of a given sentence can be identified at an accuracy of 64.58% by our best submitted run.

1 Introduction

The Arabic Language is one of the oldest languages in the world, which made Arabic dialects emerge over the years. Although Modern Standard Arabic (MSA) is the only standardized form of the Arabic language that has a predefined set of grammatical rules, it is only used in education, some media channels, and official written documents. This owes to the fact that people tend to use dialects more in their daily life. Those dialects deviate from the classical MSA in terms of morphology, phonology, lexicon, and syntax (Janet, 2007). For example, a morphological difference could be seen in the affixes that are appended to the verb to indicate its tense, like the prefix عَمْ which indicate the present tense in Jordanian dialect. The existence of many varieties of the Arabic dialects gave rise to the task of automatic identification of written Arabic dialects, since a prior identification of those dialects is essential to many applications, such as sentiment analysis, opinion mining, author profiling, and machine translation. Despite the significant differences between the dialects, they still share some similarities such as having common character/vocabulary sets and basic language rules which make dialect identification an interesting but challenging problem. Moreover, the closeness between some dialects that are within the same country makes it even more challenging to distinguish between them.

Unlike most of the previous work which targeted coarse-grained Arabic dialect identification, this work presents the participation of Qatar University team in the Multi Arabic Dialect Applications and Resources (MADAR) shared task (Bouamor et al., 2019) that addresses a fine-grained classification of 25 dialects of different Arabic cities in addition to MSA. We propose a *simple* classification approach that only utilizes word and character n-grams using Naïve Bayes learning model. While our approach is so simple (depending only on two categories of lexical features), it proved *not* to be *naïve*; the official testing results show that our best submitted run achieved reasonably-good F_1 scores across the different dialects, ranging from 0.52 to 0.84.

The rest of the paper is organized as follows. Section 2 outlines the data used in building/training our models. Section 3 details our proposed approach. Section 4 presents our runs and official testing results. Section 5 discusses and analyzes the performance of our best run. Finally, Section 6 concludes our work with some directions of future work.

2 Data

In this work, we used MADAR dataset (Bouamor et al., 2018) for training our models. The dataset consists of 2 corpora, namely corpus-

26 and corpus-6, that include sentences translated from the Basic Traveling Expression Corpus (BTEC) (Takezawa, 2007) into different Arabic dialects and MSA, however, we only used the first corpus for developing our models. Corpus-26 contains 2,000 sentences translated into 25 parallel dialects plus MSA. This corpus is divided into training, development, and testing sets of 1600, 200, and 200 sentences per dialect respectively.

Several tools were used to build our system and process our data. Two main Python libraries were used for Arabic processing: pyarabic[1] for tokenization and diacritics removal, and the natural language toolkit (NLTK) for stop-words removal. For classification, we used both Scikit-learn and sklearn Python libraries.

3 System

The aim of this work is to design a system that can identify 25 different Arabic dialects (classes) in addition to MSA. We have adopted a similar approach to the one proposed by Salameh et al. (2018). They trained a Multinomial Naïve Bayes classifier over a feature combination of word n-grams, character n-grams, language models per dialect, and sentence probabilities given by the language models, achieving an accuracy of 67.9%. In this section, the main blocks of our proposed system are presented.

3.1 Data Pre-processing

Arora et al. (2012) introduced the phrase "Garbage In, Garbage Out" to indicate that the data quality greatly affects the classification task. In our system, the data was pre-processed by tokenizing over white spaces, removing Arabic stop words, and removing punctuation.

3.2 Feature Extraction

Extracting a set of discriminative features from the data helps in differentiating between the different classes. In our proposed models, only two categories of features were considered: word n-grams and character n-grams.

- **Word n-grams**: The word unigrams and bigrams are extracted and used as features. This category helps in distinguishing between the different dialects since some words are uniquely found in specific dialects, capturing their lexical variations at the *word* level.
- **Character n-grams**: The character n-grams, ranging from 2-grams to 5-grams, are extracted and used as features. This category captures the morphological characteristics of the dialects by capturing prefixes and suffixes that distinguish some dialects at the *character* level.

[1]https://pypi.org/project/PyArabic/

Classifier	Accuracy
Multinomial Naïve Bayes	**63.21%**
Bernoulli Naïve Bayes	**63.37%**
Stochastic Gradient Descent	52.79%
Gaussian Naïve Bayes	49.37%
Perceptron (one versus all)	46.92%
Perceptron (one versus one)	46.79%

Table 1: Performance of different classifiers on the development set using word unigrams features only.

3.3 Feature Selection

Feature selection is an optimization technique that narrows down the feature space by selecting a subset of the most important features from the original set. In this work, we used Random Forest algorithm to select the top features. It is an ensemble learning algorithm that is based on combining a number of de-correlated decision trees in which the tree-based structure is naturally used to *rank* the features.

3.4 Training Classifiers

We have experimented with a number of classifiers: Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Stochastic Gradient Descent, Gaussian Naïve Bayes, and Perceptron.

To choose the best classifiers for this task, we initially trained all classifiers only on word unigrams features. Table 1 shows that Multinomial Naïve Bayes classifier (MNB) and Bernoulli Naïve Bayes classifier (BNB) had, by far, the highest (approximately equal) performance on the development set. Therefore, we only use those two in the rest of the experiments. Some previous studies also used Naïve Bayes classifiers for dialect identification, e.g., (Sadat et al., 2015).

Next, we focus on the performance of MNB and BNB classifiers with different combinations of features. We used word unigrams and bigrams,

Features		Accuracy	
word	character	MNB	BNB
1	-	63.21%	63.37%
1	2	57.40%	57.21%
1	3	62.52%	62.21%
1	4	65.67%	65.65%
1	5	65.85%	65.23%
1+2	-	63.50%	62.56%
1+2	2	58.56%	59.15%
1+2	3	62.85%	62.71%
1+2	4	66.00%	65.86%
1+2	5	65.75%	64.85%
1	2 + 3	59.42%	58.77%
1	3 + 4	64.37%	64.90%
1	3 + 5	65.35%	65.27%
1	4 + 5	**66.25%**	**65.90%**
1	3 + 4 + 5	65.33%	65.52%
1	2 + 3 + 4 + 5	64.98%	64.42%

Table 2: Performance of MNB and BNB classifiers on the development set using different combinations of features.

Classifier	#Features	Time	Accuracy
MNB	228,585	1:42h	64.71%
MNB-FS	203,200	1:31h	64.13%

Table 3: Training time and accuracy for MNB (trained over all character n-grams) and MNB-FS classifiers on the development set.

and character 2-grams to 5-grams. Table 2 shows the performance of different combinations of those features for both classifiers. The combination of the word unigrams with the character 4-grams and 5-grams achieved the highest accuracy.

Based on the performance of different classifiers and different combinations of features (shown in Tables 1 and 2), we chose the following models to represent our runs in the shared task:

1. **MNB**: In this run, a MNB classifier is trained with features that are obtained from combining word unigrams, character 4-grams and character 5-grams. No feature selection is performed here.

2. **Voting**: In this run, two Naïve Bayes classifiers are trained, the first one is MNB and the second is BNB. The classifiers are trained using the word unigrams and character 4 and 5 grams. The classification is done by voting between the two classifiers based on the higher probability. No feature selection is performed here.

3. **MNB-FS**: In this run, a MNB classifier is trained with character n-grams features ranging from 2-grams to 5-grams after feature selection. The main motivation behind this run is to improve the efficiency by reducing the feature space while maintaining good performance. The top 200 features of character bigrams, 3,000 of character 3-grams, and 200,000 of character 4-grams and character 5-grams were selected. As shown in Table 3, feature selection reduced the size of the feature space by about 11% which yields a drop in training time by about 10%. However, the performance is maintained across both models, where the difference in accuracy is only about 0.6%.

Run	Prec.	Recall	F_1	Acc.
MNB	0.6458	0.6440	0.6418	64.40%
Voting	**0.6499**	**0.6458**	**0.6445**	**64.58%**
MNB-FS	0.6292	0.6262	0.6232	62.62%

Table 4: Official performance of the 3 runs on test set.

3.5 Official Performance

Table 4 shows the overall performance on the test set for our three submitted runs. The results show that MNB run exhibited better performance than MNB-FS run. However, the Voting run, which exploits the predictions proposed by both MNB and BNB classifiers, got the best performance. This indicates that the two basic classifiers had some different predictions with different confidence (represented by the classification probabilities) that were better leveraged by the voting scheme.

4 Discussion

Figure 1 illustrates the F_1 score per dialect for the best run, where dialects of the same country are colored the same. We notice that F_1 scores range from 0.52 to 0.84. We also notice that performance on different countries within the same geographical regions is relatively consistent. For example, performance on Maghrebi group (Algeria, Morocco, and Tunisia) is relatively good, while on Levantine group (Syria, Jordan, and Lebanon) is relatively bad, and on Gulf group (Oman, KSA, and Qatar) is probably the worst. Performance

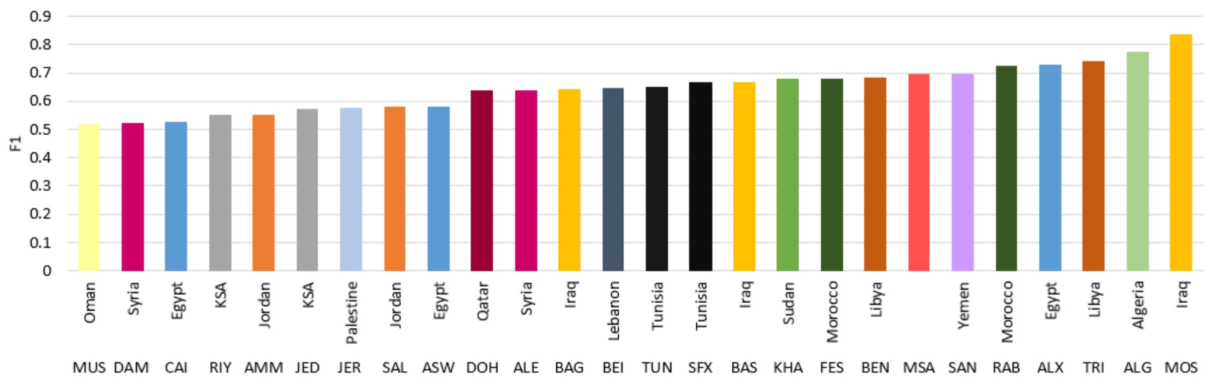

Figure 1: F_1 scores for the 25 dialects and the MSA for the best run. Same color indicates same country.

on Iraq and Libya is considered among the best. Moreover, Egypt has a noticeable but different observation; its 3 representative cities/dialects (almost) span the entire spectrum, with performance on Cairo and Alexandria (almost) make the extremes. Finally, performance on MSA is among the top third, which is expected.

To shed more light on and gain more insights about the different performance on different dialects, we illustrate the confusion matrix in Figure 2. From the matrix, it is clear that some dialects were confused with other dialects on the same country or other countries that are relatively close to it. The highest confusion is between Tunis (TUN) and Sfax (SFX) where 49 examples of TUN examples are misclassified as SFX and 29 vice versa. For example, misclassification of the TUN sentence "شنوة الفيلم الي تنصح بيه؟" as SFX can be due to the fact that the word "شنوة" appeared 41 times in SFX examples and only 14 times in TUN examples in the training data. Also, for Egypt, we can see from the confusion matrix that many examples are misclassified between these three dialects: Alexandria (ALX), Aswan (ASW), and Cairo (CAI), most notably in CAI which achieved a low F_1 score of 0.53 because 35 examples misclassified as ASW and 20 as ALX. Similarly, there is a recognizable confusion between the countries in the same geographic area. For example, for the gulf area, 13 examples of Doha (DOH) were classified as Jeddah (JED) and 14 examples of Muscat (MUS) were classified as Riyad (RIY).

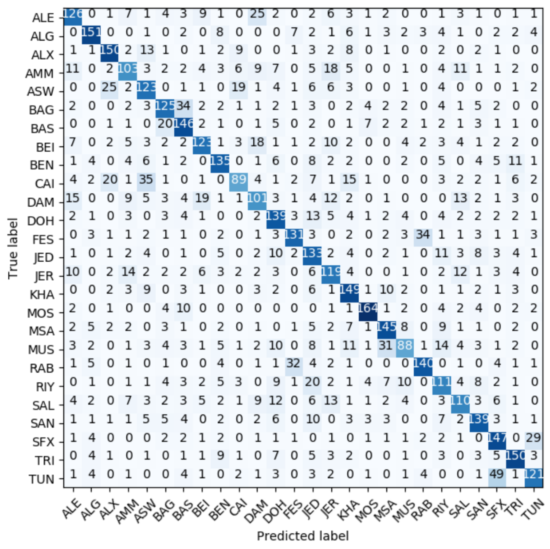

Figure 2: Confusion Matrix for the best run.

5 Conclusion & Future Work

In this work, we adopted a *simple* approach to classify the Arabic sentences into one of 25 dialects of different cities all over the Arab world, in addition to MSA, utilizing only the word and character n-grams features. Our best submitted run to MADAR shared task, that represents a voting scheme over two simple (both based on Naïve Bayes) classifiers, achieved an overall accuracy of 66.34% on the development set and 64.58% on the testing set.

That was indeed just the start. There are several directions that can potentially improve the performance of the system and address the limitations. First, extensive failure analysis has to be conducted to identify the major missclassification problems. For feature extraction, better term representation techniques, such as word and character embeddings, can be used to improve the quality of the features. For classification models, more traditional learning models (e.g., SVM) can be tried, in addition to the recently-hot deep learning models whenever applicable.

References

Alka Arora, P K Malhotra, Sudeep Marwah, Anshu Bhardwaj, and Shashi Dahiya. 2012. Data preprocessing techniques in data mining.

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. *Lrec*, pages 3387–3396.

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.

Watson Janet. 2007. *The Phonology and Morphology of Arabic*. Oxford University Press.

Fatiha Sadat, Farzindar Kazemi, and Atefeh Farzindar. 2015. Automatic Identification of Arabic Language Varieties and Dialects in Social Media. *Proceedings of the Second Workshop on Natural Language Processing for Social Media (SocialNLP) Dublin, Ireland*, pages 22–27.

Mohammad Salameh, Houda Bouamor, and Nizar Habash. 2018. Fine-Grained Arabic Dialect Identification. *Processdings of the 27th International Conference on Computational Linguistics Santa Fe, New Mexico, USA*, pages 1332–1344.

Toshiyuki Takezawa. 2007. Multilingual spoken language corpus development for communication research. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 303–324. The Association for Computational Linguistics and Chinese Language Processing.