Faheem at NADI shared task: Identifying the dialect of Arabic tweet

Nouf A. Al-Shenaifi
Department of Computer Science,
College Comput. & Info. Sciences,
King Saud University,
Riyadh 11543, Saudi Arabia
noalshenaifi@ksu.edu.sa

Aqil M. Azmi
Department of Computer Science,
College Comput. & Info. Sciences,
King Saud University,
Riyadh 11543, Saudi Arabia
aqil@ksu.edu.sa

Abstract

This paper describes *Faheem* (adj. of understand), our submission to NADI (Nuanced Arabic Dialect Identification) shared task. With so many Arabic dialects being understudied due to the scarcity of the resources, the objective is to identify the Arabic dialect used in the tweet, at the country-level. We propose a machine learning approach where we utilize word-level n-gram ($n = 1$ to 3) and *tf-idf* features and feed them to six different classifiers. We train the system using a data set of 21,000 tweets—provided by the organizers—covering twenty-one Arab countries. Our top performing classifiers are: Logistic Regression, Support Vector Machines, and Multinomial Naïve Bayes (MNB). We achieved our best result of macro-$F_1 = 0.151$ using the MNB classifier.

1 Introduction

Arabic language users constitute the fastest-growing language group when it comes to the number of Internet users. According to (www.internetworldstats.com/stats7.htm), Arabic language Internet users have grown by 9,348% followed by Russian with a growth of 3,653% during the last twenty years. There is a serious lack of Arabic contents in the Internet. As one author put it, searching for "How to make ice cream?" in Arabic yields 99,100 results, while the same search in English yields 37,600,000 hits. Contrast this with twenty-two Arab countries, holding 420M speakers, vs 260M native English speakers worldwide (Pangeanic.com, 2013).

The majority of the Arabic content in the web is in Modern Standard Arabic (MSA), and though it is understandable to most of the Arabs, the mass majority do not use it in their daily communication. Rather they switch to colloquial or informal Arabic, which was originally confined to the domain of verbal communication (Azmi and Aljafari, 2018). Social media gave masses the opportunity to write in whatever language suites them, and this led to a rise in writing informally. The dialectical Arabic differs from region to another, and the vocabulary of some dialects overlaps with MSA by as much as 90% (Cadora, 1976); however, differences include some very common words such as those meaning "see", "go", and "not", as well as phonology, syntax and morphology rules (Habash and Rambow, 2006).

Dialect identification is the task of automatically identifying the dialect of a particular part of text (Zaidan and Callison-Burch, 2014). Arabic dialects differ from one region to another, and there are no available dictionaries for their vocabulary nor written rules. The range of Arabic dialects is much more varied than the range of dialects that are typically considered to comprise European languages such as English and French (Azmi and Aljafari, 2018). Even with the lack of written rules there are common set of conventional orthography guidelines for Arabic dialect (Habash et al., 2018; Eryani et al., 2020). There guidelines may give clue to the dialect.

The Nuanced Arabic Dialect Identification (NADI) shared task is associated with the Fifth Arabic Natural Language Processing Workshop (WANLP 2020) (Abdul-Mageed et al., 2020). The shared task follows the efforts of Arabic Natural Language Processing (NLP) workshops and conferences that have taken place in the recent years (Bouamor et al., 2018; Bouamor et al., 2019). Bouamor et al. (2018)

Proceedings of the Fifth Arabic Natural Language Processing Workshop, pages 282–287
Barcelona, Spain (Online), December 12, 2020

describes the first effort to collect a large-scale corpus and lexicon for dialectal Arabic. The MADAR corpus is what was used to build the MADAR Shared task. Salameh et al. (2018) is another important work where the authors worked on city-level dialect identification. The NADI shared task consists of two subtasks. The goal of the first subtask is to identify the origin country of the tweet based on the dialect. While the second subtask focused on province-level dialect, where the goal is to predict the country province of the tweet. The organizers provided a data set of 21,000 labeled tweets covering 21 Arab countries with country and province labels. In addition, they provided 10M unlabeled tweets covering different Arab countries.

We briefly summarize the challenges as: (a) the maximum length of a tweet is 280 characters which includes hashtags, special characters, URLs, etc. This does not leave enough information—in term of words—to accurately predict the dialects; (b) some of the tweets are written in MSA rather than country dialect which complicates the classification task; (c) certain similarity between dialects and the mixing between MSA and dialect in a given text (Biadsy et al., 2009); and (d) some of the tweets are re-tweets possibly from another different country with a different country dialect.

In this paper, we develop a machine learning model to identify and predict the dialect of a tweet over 21 different Arabic dialects (NADI shared task subtask 1). Dialect identification can be seen as a multi-class text classification task, where we predict the probability of a dialect given a tweet (set of words) (Talafha et al., 2019). Our proposed system use word n-grams and *tf-idf* as features, and six different classifiers (e.g., Multinomial Naïve Bayes, Logistic Regression). In developing the system we relied solely on the data provided by the shared task organizers, no external data was involved. We assume the data is free of spelling error, and/or context-sensitive error. The latter error occurs when a user types a correctly spelled word when another is intended (Azmi et al., 2019), e.g. "I want a peace (piece) of cake".

The rest of the paper is organized as follows. In Section 2, we present the data set used in building our machine learning models. We describe the proposed system in Section 3. Section 4 discusses system performance and the results. We conclude with suggestions for future research in Section 5.

2 Dataset and evaluation

All the registered participants to the NADI shared task received the data set. The data set is divided into Train, Development (Dev), and Test sets (see Table 1). The Training set contains 21,000 Arabic tweets, with each tweet is associated with its dialect label. The labels mark the country the dialect is used in. Table 2 lists the 21 Arab countries which are covered by the Training set, along with their size. The Dev set contains 4,957 tweets with corresponding Arabic dialect labels, while the Test set contains unlabeled 5,000 tweets. This blind Test data set will be used to evaluate the output of the participating teams.

Data set	No. of tweets
Training	21,000
Dev	4,957
Testing	5,000

Table 1: Details of the NADI shared task data set.

We have an imbalanced class distribution in the Training data set, tweets are not equally divided among the countries (see Table 2). Egypt has the highest number of tweets in the Training set ($\approx 21\%$), while only 1% of the tweets are labeled Sudan, Somalia, Mauritania, Bahrain, and Djibouti.

The evaluation metrics includes precision (P), recall (R), F-score, and accuracy. The official metric is the macro-averaged F-score, which computes the metric independently for each country and then take the average (hence treating all countries equally). Let $C = \{\textit{Egypt}, \textit{Iraq}, \textit{Saudi}, \ldots\}$ be the list of countries. If we denote F-score of classifying the tweet into country c by $F_1(c)$, then we can define the macro-average F-score as,

Country	# tweets	%	Country	# tweets	%
Egypt	4,473	(21.30)	Lebanon	639	(3.04)
Iraq	2,556	(12.17)	Jordan	426	(2.03)
Saudi Arabia	2,312	(11.00)	Palestine	420	(2.00)
Algeria	1,491	(7.10)	Kuwait	420	(2.00)
Oman	1,098	(5.23)	Qatar	234	(1.11)
UAE	1,070	(5.10)	Sudan	210	(1.00)
Syria	1,070	(5.10)	Somalia	210	(1.00)
Morocco	1,070	(5.10)	Mauritania	210	(1.00)
Libya	1,070	(5.10)	Bahrain	210	(1.00)
Yemen	851	(4.05)	Djibouti	210	(1.00)
Tunisia	750	(3.57)			

Table 2: Distribution statistics of the Training data set. Numbers inside bracket is the size as percentage.

$$\text{macro-}F_1 = \frac{1}{n} \sum_{c \in C} F_1(c).$$

3 System overview

Our goal is to accurately classify the tweets to one dialect out of 21 Arabic dialects (corresponding to twenty-one Arab countries). To develop our system, we used python, the machine learning library Scikit-learn[1] (Pedregosa et al., 2011), and NLTK (Natural Language Toolkit) library (Loper and Bird, 2002). There are three main stages in our system: pre-processing the data, feature extraction, and finally classification.

3.1 Data pre-processing

We pre-process all the data sets (i.e. Train, Dev, and Test). The pre-processing includes tokenizing over the white spaces, removing URLs and user mentions from tweets, removing punctuation and special characters, removing of non-Arabic words. No stemming was applied. Stemming is useful on MSA text but not for the dialect. There is no single stemmer that can be applied on all Arabic dialects, and so we need to apply a different stemmer for each dialect (e.g. stemmer for Arabian Gulf dialect, stemmer for Egyptian dialect, etc). As the dialect is not known beforehand, so it does not make sense to apply it. None of the participants in MADAR shared task 2019 (Bouamor et al., 2019) used stemming. Stopword removal is another task that is common in Arabic NLP applications. Again, we did not remove the stopwords. Simply, it is the same argument behind not using stemming.

3.2 Feature extraction

The *tf-idf* (short for Term Frequency-Inverse Document Frequency) is a weighting factor intended to reflect how important a word is to a document (tweet) in a corpus (Leskovec et al., 2014). The *tf-idf* value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word. In our system we build a model that depends on extracting a word-level n-grams *tf-idf* features for ($n = 1$ to 3), and character-level n-grams *tf-idf* features. According to Mishra and Mujadia (2019), combining word and character level n-gram *tf-idf* performs better than individual word or character *tf-idf* vectors. The word n-grams helps in differentiating between dialects as some words are confined to a single dialect. In contrast, the character-level n-grams help in finding morphological (prefixes and suffixes) characteristics of the dialects.

3.3 Training classifiers

After the features are extracted, we need to predict the tweets' dialect labels. We used six different classifiers. Based on the performance of different classifiers, we picked the following three classifiers to submit our official entry into the shared task:

[1] Available at `https://scikit-learn.org`.

- Multinomial Naïve Bayes (MNB) classifier with Laplace smoothing parameter alpha $= 0.5$.
- Support Vector Machines (SVM) classifier with a linear kernel, and cost $C = 1$. The multi-class is handled using one-vs-the-rest strategy.
- Logistic Regression (LR). For the multi-class training we used one-vs-the-rest scheme.

The other three classifiers which we experimented with but failed to yield satisfactory performance are: Gaussian Naïve Bayes (GNB), Decision Tree (DT), and Random Forest (RF).

Moreover, we also experimented with deep learning-based classifier using python Keras API (Géron, 2019). However, deep learning models did not perform well on this task. This is due to the nature of the data set itself. Deep learning models require huge training data set which is not the case here. There are twenty-one different dialects, and that leaves a relatively small number of tweets per dialect.

4 Results and discussion

4.1 Selecting language model

Our model uses word-level n-grams. We need to determine what is the appropriate range for n. We trained the system using Training data, and tested the system on Dev data (Table 1). Table 3 reports the performance measures for different values of $n \in [1, 4]$. For all the classifiers we do not observe any improvement in the performance for $n > 2$. So, it suffices we use language model n-grams for $n = 1$ and 2. However, we wanted to be extra cautious and went for $n = 1$ to 3.

	$n = 1$			$n = 1$–2			$n = 1$–3			$n = 1$–4		
Classifier	P	R	macro-F_1	P	R	macro-F_1	P	R	macro-F_1	P	R	macro-F_1
SVM	0.84	0.83	0.82	0.85	0.84	0.83	0.85	0.84	0.83	0.83	0.82	0.80
MNB	0.79	0.77	0.76	0.81	0.79	0.78	0.78	0.76	0.74	0.78	0.76	0.74
LR	0.83	0.82	0.80	0.82	0.81	0.79	0.82	0.81	0.79	0.82	0.81	0.79
RF	0.71	0.70	0.69	0.71	0.70	0.69	0.71	0.69	0.67	0.70	0.68	0.66
DT	0.65	0.64	0.61	0.65	0.65	0.62	0.65	0.65	0.62	0.65	0.65	0.62
GNB	0.71	0.69	0.65	0.72	0.70	0.66	0.70	0.69	0.64	0.70	0.69	0.64

Table 3: Performance measurement on Dev data using different values of n for word n-grams.

4.2 System performance

We tested our system on Dev data set as well as the Training data set (Table 1). For the latter, we used 10-fold cross-validation, where the 21,000 tweets were randomly divided into ten equal folds. In each iteration a different fold is used for testing, while the other nine folds are used for training. The whole process is repeated ten times. Cross-validation (CV) generally results in a less biased or less optimistic estimate of the model than other methods, such as a simple train/test split. Table 4 summarizes the results of testing on the Training data using 10-fold CV, and Dev data using different classifiers. In both cases, the top three classifiers are SVM, MNB, and LR, with a slight edge in favor of SVM. The SVM achieved macro-$F_1 = 0.42$ and 0.83 on the Training and the Dev data (respectively).

	Training data (10-fold CV)			Dev data		
Classifier	P	R	macro-F_1	P	R	macro-F_1
Support Vector Machine	0.42	0.46	0.42	0.85	0.84	0.83
Multinomial Naïve Bayes	0.46	0.43	0.39	0.81	0.79	0.78
Logistic Regression	0.42	0.42	0.38	0.83	0.82	0.80
Random Forest	0.39	0.37	0.31	0.71	0.70	0.69
Decision Tree	0.31	0.33	0.31	0.65	0.65	0.62
Gaussian Naïve Bayes	0.33	0.36	0.33	0.72	0.70	0.66

Table 4: Summary of results using different classifiers on two data sets.

4.3 Official results

Based on our experiment, our best performing classifiers are SVM, MNB, and LR (Table 4). The organizers of NADI shared task (subtask 1) allow up to three submissions. Table 5 lists the official result of our submission on the Test data set as reported by the organizers. This time the Multinomial Naïve Bayes yield slightly better performance compared to SVM. However, the performance results of the three classifiers are close to each other.

Classifier	P	R	macro-F_1	Accuracy
Support Vector Machine	0.17303	0.14810	0.14613	0.3224
Multinomial Naïve Bayes	0.22342	0.14714	0.15095	0.3402
Logistic Regression	0.25430	0.14305	0.14911	0.3280

Table 5: Official results of our submitted run on the Test data set.

4.4 Discussion

Of the six classifiers, SVM, MNB, and LR where the top three on two different data sets (Dev and Training using 10-fold CV). Surprisingly, none of the three classifiers did well in the official shared task. Our best performance on the Test data set was macro-$F_1 = 0.15095$ using the MNB classifier (Table 5), which is a far cry from its behavior in Table 4.

	Actual judgment	
Classifier prediction	True	False
Positive	TP	FP
Negative	TN	FN

Table 6: Confusion matrix.

Suppose we randomly assign the dialects to a tweet. As each tweet could belong to any of the 21 dialects, then the probability of correct assignment (assume equal likelihood) is $1/21 = 4.76\%$. The precision P is defined as TP$/$(TP + FP), while recall R is given by TP$/$(TP + FN), where TP, FP, and FN stands respectively for true positive, false positive, and false negative. The terms true and false refer to whether that prediction corresponds to the actual judgment and the terms positive and negative refer to the classifier's prediction (see Table 6). For example, we may interpret FP as the number of tweets t that are falsely classified as belonging to dialect d; similarly, FN as the number of tweets t that are falsely classified as *not* belonging to dialect d. We expect FN to be $1 - 1/21 \approx 0.95$ which is close to 1.

From the above argument, we can state that any value for P and R over 4.76% means the system is doing better than randomly assigning the dialects. Thus, $P = R = 0.476$ should be the baseline. From Table 5 the value of R is almost the same for all three classifiers, while for P there is a noticeable difference. If we take MNB classifier, $R = 0.147$, substituting 0.95 for FN and solving, we get TP $= 0.1637$. This means 16.37% of the time we are identifying the dialect correctly. Substituting into precision and solving, we get the value FP $= 0.57$. Which means over half of the time (57%) we are wrongly assigning the dialect. The F-measure $F_1 = 2/(1/P + 1/R) = 2\text{TP}/(\text{TP} + \text{FP} + \text{TN} + \text{FN})$. This means if we want to improve macro-F_1 then we just need to boost TP.

5 Conclusion

In this paper we described our participation in the 2020 Nuanced Arabic Dialect Identification (NADI) shared task, where the goal is to build a system that can predict which of the 21 Arabic dialects the tweet is written in. We submitted the result of executing our system on a standard test data using three different classifiers: SVM, Multinomial Naïve Bayes, and Logistic Regression. Our official performance of macro-$F_1 \approx 0.151$ using the Multinomial Naïve Bayes classifier is not in par with our aspiration.

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.

Aqil M Azmi and Eman A Aljafari. 2018. Universal web accessibility and the challenge to integrate informal Arabic users: a case study. *Universal Access in the Information Society*, 17(1):131–145.

Aqil M Azmi, Manal N Almutery, and Hatim A Aboalsamh. 2019. Real-word errors in Arabic texts: A better algorithm for detection and correction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1308–1320.

Fadi Biadsy, Julia Hirschberg, and Nizar Habash. 2009. Spoken arabic dialect identification using phonotactic modeling. In *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*, pages 53–61.

H Bouamor, N Habash, M Salameh, W Zaghouani, O Rambow, D Abdulrahim, O Obeid, S Khalifa, F Eryani, A Erdmann, and K Oflazer. 2018. The MADAR Arabic dialect corpus and lexicon. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, 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 4th Arabic Natural Language Processing Workshop*, pages 199–207, Florence, Italy. Association for Computational Linguistics.

Frederic J Cadora. 1976. Lexical relationships among Arabic dialects and the Swadesh list. *Anthropological linguistics*, 18(6):237–260.

Fadhl Eryani, Nizar Habash, Houda Bouamor, and Salam Khalifa. 2020. A spelling correction corpus for multiple Arabic dialects. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4130–4138.

Aurélien Géron. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Nizar Habash and Owen Rambow. 2006. MAGEAD: a morphological analyzer and generator for the Arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688.

N Habash, F Eryani, S Khalifa, O Rambow, D Abdulrahim, A Erdmann, R Faraj, W Zaghouani, H Bouamor, N Zalmout, S Hassan, F Al-Shargi, S Alkhereyf, B Abdulkareem, R Eskander, M Salameh, and H Saddiki. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3628–3637.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge University Press.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. *arXiv preprint cs/0205028*.

Pruthwik Mishra and Vandan Mujadia. 2019. Arabic Dialect Identification for Travel and Twitter Text. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 234–238.

Pangeanic.com. 2013. Arabic content on the Internet: an unfilled gap. `https://www.pangeanic.com/knowledge_center/arabic-content-internet-unfilled-gap/`. Accessed: 2020-07-05.

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.

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.

Bashar Talafha, Wael Farhan, Ahmed Altakrouri, and Hussein Al-Natsheh. 2019. Mawdoo3 AI at MADAR Shared Task: Arabic Tweet Dialect Identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 239–243.

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