Classical Arabic Poetry: Classification based on Era

Mariam Orabi
Department of Computer Science
University of Sharjah
Sharjah, UAE
morabi@sharjah.ac.ae

Hozayfa El Rifai
Department of Computer Science
University of Sharjah
Sharjah, UAE
hozayfarifai@protonmail.com

Ashraf Elnagar
Department of Computer Science
University of Sharjah
Sharjah, UAE
ashraf@sharjah.ac.ae

2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA) | 978-1-7281-8577-4/20/$31.00 ©2020 IEEE | DOI: 10.1109/AICCSA50499.2020.9316520

Abstract—**This paper proposes a CNN-based deep learning model that classifies Arabic poems based on its era, which is not reported before. To build this model, constructing a dataset is the first step, so we propose an updated Arabic Poetry Dataset (2020). We use FastText word embeddings, based on the full corpus of poems (unlabeled). Two classifiers were trained, namely, a supervised deep learning classifier and a FastText-based classifier. We conducted several experiments. First, we implemented a polarity classifier of poems to modern and non-modern eras, which achieved highest accuracy and F1-score of 0.913 and 0.914, respectively, using a deep learning model without frequent terms. In the second experiment, we categorized poems into three eras. The classifier reported an accuracy and F1-score of 0.875 each. Last, the classification of poems into five different eras achieved highest accuracy and F1-score of 0.801 and 0.796, respectively.**

Index Terms—**Natural Language Processing, Text classification, Arabic language, Poetry, Convolutional neural networks, Deep learning, Arabic Poetry Dataset**

I. Introduction

Text classification, or categorization, is the task where a machine is given a piece of text and decides to which set of a predefined set of classes this piece of text belongs to [1]. It is one of the main tasks of Natural Language Processing which is used in spam detection [2], [3], sentiment analysis [4]–[10], topic labeling [11], dialects classification [12], Arabic numerals detection [13], and a lot more. Similarly, Arabic speech recognition has attracted some attention lately, [14]–[16].

Text classification is a challenging task especially if the language for which it is used has complicated inflections and lots of variations, such as Arabic language [17]–[20]. The latter is the fifth among the most popular languages, used by approximately 422M people [21]. Arabic poetry changed over time, topics of Arabic poems, used terms and the rhythm of the poems have changed [22]. The variation of topics and used terms over poetry eras can be an indicator to discriminate the eras of Arabic poems. This feature motivates using machine learning to classify Arabic poems into the different eras. The authors in [23] classified Arabic poems from the 5^{th} to the 15^{th} century to the following four eras: Pre-Islamic era, Umayyad era, Andalusian era and Abbasid era. The work used seven different shallow learning models to build classifiers without using trained word embeddings. The highest achieved $F1$-score was of 68.8% using Multinomial Naïve Bayes classifier.

However, up to the best of our knowledge, no work included Modern Era in their classifications, and no work used deep learning to classify Arabic poetry even though deep learning might be more promising than shallow learning. This paper classifies Arabic poetry into its corresponding era including Modern era. The aim of this paper is to use FastText [24] word embeddings to train a supervised deep learning model using Convolutional Neural Networks to classify a given piece of poem written in Classical Arabic into its origin era, and compare the performance with a baseline classifier without using trained word embeddings.

The inclusion of Modern Era in our classifications raised multiple difficulties. That is, the corpus of old eras (prior to modern) are small and cannot be enlarged, while the number of Modern Era poems is huge compared to any other era. This created an imbalanced dataset that is hard to train especially with deep learning models. Multiple approaches were used and compared to overcome this issue and find the best approach for Arabic poetry classifications based on eras.

This work has three main milestones; first, to build a model that can binary classify a poem into Modern or Non-Modern eras. Second, to build a tri-class classifier that can classify a given poem to one of the three following eras: Pre-Islamic & Islamic, Umayyad & Andalusian and Modern eras. Third, to build a model that can classify a poem into one of the five eras: Pre-Islamic era, Islamic and Umayyad era, Andalusian era, Abbasid era and Modern era. The contributions of this paper are as follows:

- A cleaned dataset of all five eras of classical Arabic poetry that shall be made publicly available for the research community on Mendeley.
- Arabic poetry is classified based on all five eras of Arabic poetry using a baseline classifier and a deep learning classifier with word embedding. However, the only related work, [23], excluded Modern Era from its classifications and used only shallow learning models without word embeddings.
- Multiple approaches were used to deal with the imbalanced dataset issue of the Arabic poetry eras. Approaches were compared to find the most suitable method to avoid possible over-fitting.

The rest of this paper describes the following: related works are reviewed and the gap that is filled by this paper is shown in section II. Dataset collecting, cleaning, and preprocessing of labeled dataset is described in section III, in addition to generating the word embeddings. The architecture of the used classifiers is presented in section IV. Finally, after training and testing the classifiers, the results are analyzed in section V, a discussion about the results is presented in section VI and conclusions are reported in section VII.

II. Related Works

Recently, multiple Text Classification task were done on Arabic poems, such as classifying Arabic text as poem or not by Almuhareb et al. [25] and extracting classical Arabic poem from Arabic text by Almuhareb et al. [26]. Alfalahi et al. [27] and Alfalahi, Ramdani and Bellafkih [28], [29] examined authorship contribution to Arabic poetry by identifying the authors of input poems, given some information about the style of poems for each poet. Al-Harbi et al. [30] classified Arabic poems to one of the following topic classes: wisdom, elegiac, love, praise, satire and description. Similarly, in his work Mohammad [31] classified Classical Arabic poetry into eight classes that represent the types of Arabic poetry, where multiple features are extracted, such as roots of the words. In a slightly similar approach, Alsharif, Alshamaa and Ghneim [32] classified Arabic poetry based on the to the following classes: elegiac, love, pride and satire. The best performance their classifier achieved was gained by using non-stemmed, non-rooted, mutually deducted feature vectors.

Yet, none of the above literature is related to Arabic poetry classification based on its era. A closely related work that classified Arabic poems from the 5th to the 15th century to the following four eras: Pre-Islamic era, Umayyad era, Andalusian era and Abbasid era is reported by Abbas, Lichouri and Zeggada [23]. Among seven different supervised models that were used, Multinomial Naïve Bayes classifier achieved the highest F1-score of 68.8%. This work excluded Modern Era from its classifications and used only shallow learning models without trained word embeddings, even though these approaches can be promising to enhance the performance of the classifier.

Thus, up to the best of our knowledge, no work used deep learning to classify poetry based on eras and none used trained word embeddings. Moreover, none have considered Modern Era in the classification of Arabic poetry based on its era. This paper addresses these gaps and provides a deep learning-based classifier for Arabic poetry based on era using FastText word embeddings [24]. It provides an updated publicly available dataset for Arabic poetry.

III. Dataset Preparation

A. Dataset Description

The dataset used was scraped from Adab website[1], which contains 60377 poems in Classical Arabic by 739 poets. This dataset extends the dataset used by *Abbas, Lichouri and Zeggada* [23] by about 2k poems, that is, the original dataset[2] was scraped in 2017.

TABLE I and TABLE II present the collected dataset as described in the Adab website; the poems are categorized by their chronological origins as follows: Pre-Islamic era, Islamic era, Andalusian era, Abbasid era and Modern era. The first four eras can be considered as Non-Modern era. Each two consecutive eras were merged to make eras less overlapping in terms of content and years, except the Modern era. The poems were broken such that each poem fragment contains no more than 20 lines of words (less than 200 words). The total number of poems fragments is approximately 86k poems fragments, we will call them poems for short.

TABLE I
Number of Poems of Each Era (Polarity)

Era	Period	Number of Poems
Modern	1798 CE till now (2020)	44925
Non-Modern	478 CE to 1492 CE	41136
Total		*86061*

TABLE II
Number of Poems of Each Era (Multiclass)

Era	Period	Number of Poems
Modern	1798 CE till now (2020)	44925
Andalusian	750 CE to 1492 CE	8914
Abbasid	750 CE to 1258 CE	25857
Islamic & Umayyad	625 CE to 750 CE	4592
Pre-Islamic	478 CE to 624 CE	1773
Total		*86061*

B. Cleaning and Preprocessing

After collecting the data, cleaning was required where only Arabic alphabets is kept and all punctuation, symbols, numbers and Arabic diacritization are removed. The total number of poems of each era for polarity classification and multi-class classification are described in TABLE I and TABLE II, respectively.

It was shown by *Abbas, Lichouri and Zeggada* [23] that stop words removal would enhance the classification of Arabic poetry for Non-Modern eras. To verify this claim, we created a copy of the dataset for which the stop words are filtered out. The list of Arabic stop words is collected from NLTK library [33]. Additionally, other experiments were applied where terms frequencies were computed and terms that appeared in more than one quarter of the poems were removed from the dataset. This was applied as an alternative option instead of removing predefined stop words.

A further step of cleaning was applied after finding that some poem fragments consist of only few meaningless words. Therefore, poem fragments of less than 8 words (almost one full line of poetry) were dropped from the dataset at the time of training and testing; to make sure that the classifier will not

[1]Adab website: urlhttp://www.adab.com/

[2]Dataset: https://www.kaggle.com/fahd09/arabic-poetry-dataset-478-2017

give extra weight to words of such meaningless fragments. Moreover, the fragments that include more than 200 words were truncated to fit in the vectors that represent the poems. Note that the poems are already broken into fragments of no more than 20 lines, this is just an extra step to be sure that no error will occur while transforming poems into vectors.

For polarity classification, the dataset is almost balanced and does not require further modification. On the other hand, the dataset for multiclass classification is not balanced. Both datasets may need either modifications to the dataset or to the classifier. The solutions that were considered to handle this issue through modifying the dataset are over-sampling or under-sampling. Another solution in which the model is modified is through assigning weights based on the sizes of the classes in the dataset and feeding these weights to the classifier to balance weights of each class. However, this issue will be discussed in section IV-B

C. Word Embedding Models

Word embeddings are extracted from the unlabeled corpus of Arabic poems using FastText [24], where the words are broken into several n-grams instead of feeding individual words to the network. It works better than Word2Vect when dealing with terms that are out of the vocabulary of the embedding. Therefore, FastText word embeddings work better for Arabic language, as it is rich in morphological terms. After preparing the dataset to be compatible with FastText, the embeddings were trained in an unsupervised manner on the unlabeled corpus using skipgram model [24], were the dimensions of the vectors is set to be 200, number of epochs is 5 and the minimum and maximum length of character n-grams are 3 and 6, respectively. Two models were obtained, one for the full dataset and one for the dataset without stop words.

IV. CLASSIFIERS

After the dataset was collected, cleaned, preprocessed, and word embeddings were extracted and trained, the supervised models were trained to build classifiers. The deep learning classifier used in this project is the Convolutional Neural Network. This classifier is chosen because of its ability to detect patterns, and in this case, we are looking for topic terms patterns. Mainly, three classifiers were built in this work. First, polarity classifier where Arabic poems are classified into two main eras: Modern era and Non-Modern era. Second, a tri-class classifier in which each two consecutive eras are grouped together, except for the modern era. Third, multiclass classifier where Arabic poems are classified into five eras: Pre-Islamic era, Umayyad era, Andalusian era, Abbasid era and Modern era. All classifiers are described in the following subsections:

A. Polarity Classifiers

Two polarity classifiers were built, namely, FastText-based classifier [24] and CNN classifier.The reason to build two classifiers is to compare the results of the CNN classifier with the well-known FastText-based classifier. Our aim is to surpass the classification performance of previous similar work reported in [23]. The following describes FastText and CNN classifiers.

1) Baseline Classifier: The implementation of this classifier was done using the python library for FastText. This classifier was built based on the default parameters of the model, which gave the best results. Dimensionality of the word vectors was the only parameter to change, which was set to 200 to be more compatible with the provided dataset. The default value for learning rate is 0.1, the loss function is softmax and the number of epochs is 5.

2) Deep Learning Classifier: The CNN classifier was built using Keras . A sequential model is used to create the layers of the model. The layers of the model has the following characteristics:

- An embedding layer, for which, the tokenized words will be represented by their word vectors, where the word embeddings are feed to the network.
- A one-dimensional dropout layer follows the embedding layer for regularization reasons.
- The convolutional layer of 512 neurons follows, with a 33 filter matrix and a pooling layer to reduce the number of parameters after the convolutional layer.
- Then, a hidden dense layer is added with dimensionality of 512 nodes, followed by a dropout layer.
- The activation for the convolutional layer and the dense layer is Rectified Linear Activation (ReLU) which proved to work well in neural networks. The output of this activation will be in range $[0, +\infty[$.
- Finally, the output layer is a dense layer of dimensionality equal to the number of output classes (2, 3 or 5 classes).
- The activation function for output layer is softmax, as we want to get a probability distribution and choose the era that has maximum probability.
- The loss function (function to minimize) is Sparse Categorical Cross-entropy, as it is suitable for multiclass classification, as well as, for binary class classification.
- The optimizer used is Adam , which allows adaptive learning rate for every parameter.
- The number of epochs is 20; chosen based on experiments. The validation split is 0.1 and the metric to be followed is validation accuracy where the model will be overwritten during epochs only if the validation accuracy has increased.

B. Multiclass Classifiers

The classifier is similar to what was done for polarity classifiers, both multiclass classifiers were built using the same methods as described in section IV-A. Minimal edits were applied to the CNN classifier to adapt to the multi-classes in the dataset. For the CNN classifier, the output layer has 3 nodes in the tri-class and 5 nodes in the multi-class case. Moreover, to overcome the imbalance in the data, approaches were applied as described below:

- A classifier with class weights was created. These class weights alter the amount of penalty for the mistakes of each class. We defined class weights to be proportional to the ratio of the number of instances of each class compared with the full dataset. Higher class weights are assigned to classes of small sizes and lower-class weights are assigned to larger classes.

- Random under-sampling was implemented, where the classes of larger sizes are resampled and only the number of samples in the smallest class is kept. This technique was applied in two different scenarios, the usual scenario where all the classes except the smallest class are under-sampled. The other scenario in the classification of five eras, where only the largest two classes, namely, Modern era and Abbasid era, are under-sampled to be of size similar to the Andalusian era. Then, class weights are assigned to balance the resulting classes. This technique was motivated by the idea of applying under-sampling only for the classes that cause very large imbalance in the dataset.
- The last approach is random over-sampling, where the classes of smaller sizes will be resampled to match the size of the larger classes. This must be implemented only on the testing set, otherwise the classifier will see samples from training set replicated from the testing set. Naïve over-sampling had bad influence on the classification of five eras, as the Pre-Islamic era was replicated about 40 times. Therefore, a mixture of under-sampling and over-sampling was applied. The method is like under-sampling with class weights, but over-sampling was used instead of class weights.

V. EXPERIMENTAL RESULTS

The classifiers were built and tested in Python using Jupiter Notebook. To compare between the performances of the different classifiers, $F1$-score is used. To understand the performance of our deep learning classifiers, Confusion Matrices (CM) is used. Unlike accuracy, these metrics can precisely describe the performance of the classifiers with imbalanced datasets. The results for the polarity classifiers and multiclass classifiers are presented in the following subsections.

A. Polarity Classifiers

Multiple experiments were carried out to find the best results are shown in TABLE III.

The baseline classifier showed the best performance when stop words and frequent terms are not removed from the dataset, its performance slightly decreased when frequent terms are removed (frequent terms are terms that appear in more than one quarter of the poems).

The deep learning classifier outperforms the baseline classifiers. Removing stop words slightly enhanced the performance of the classifier, but it performs best when frequent terms are removed instead. This does not conflict with the results from [23], where keeping stop words for the classification of Non-Modern eras resulted in better classification results. That is, in their classification, Modern era was not included.

To compare the performance of the deep learning classifier with and without most frequent terms.The ability of the classifier to identify poems of the Modern Era increased when removing frequent terms, while it slightly decreased for the Non-Modern era.

TABLE III
PERFORMANCE OF POLARITY CLASSIFIERS

Classifier	Accuracy	$F1$-score
Baseline classifier		0.898
Baseline w/o stop words		0.890
Baseline w/o frequent terms		0.896
CNN	0.909	0.911
CNN w/o stop words	0.912	0.912
CNN w/o frequent terms	**0.913**	**0.914**

B. Multiclass Classifiers

As described in section IV-B , multiple classifiers were built to overcome the imbalance of the data in multi-class classifications, this resulted in many classifiers to evaluate. As per the previous experiments of polarity classifiers, removing frequent terms enhanced the classification more than removing stop words. Therefore, for multiclass classification we only compare the classification results while altering the existence of frequent terms only.

1) Classification on Three Eras: The results of the classification on three eras are shown in TABLE IV. The best results are achieved by the CNN classifier without any modifications. Removing frequent terms did not improve the classification; on the contrary, these terms have good value to the classification process. Assigning class weights performed well, but not better than the original classification. It is because the imbalance in this case is not extremely high. Over-sampling and under-sampling performed worse than assigning class weights.

The poems of every two consecutive eras are slightly misclassified. This is expected and can interpreted by the similar terms used in two consecutive eras. Terms that are used in Arabic poetry degrade with time.

2) Classification on Five Eras: It can be seen in TABLE V that the performance of the baseline classifier did not change with and without frequent terms. The CNN classifier has slightly lower $F1$-score than the baseline classifier, but removing frequent terms improved the CNN performance.

Also, assigning class weights improved the CNN classifier's performance, it has the best performance among the rest of the classifiers. Removing frequent terms does not make much difference for a classifier with predefined class weights, in fact, it slightly lowered the accuracy. This lines up with the findings from [23]. However, as keeping and removing frequent terms, performance of tri-class classifiers did not improve the classification, this approach was not used with under-sampling and over-sampling.

Even though under-sampling all classes except the smallest one resulted in worsening the performance, this approach has the closest confusion matrix to the unit matrix. Under-sampling the largest 2 classes and assigning weight to the rest resulted in slightly better performance. The CNN classifier needs large training dataset to achieve good performance. Under-sampling hinders this from happening by reducing the size of the set. Yet, applying both under-sampling for the largest 2 classes and over-sampling for the smallest 2 classes did not perform any better.

To get a closer look at the performances of multiclass classifiers, TABLE VI shows the $F1$-score for each class for each of the mentioned classifiers. The Pre-Islamic era has the lowest $F1$-score in most of the classifiers due to the small size of the set. This issue is hard to solve as poems from this era are few, because it is the oldest era that is more than 14 centuries ago.

Overall, assigning class weights gives the best accuracy and $F1$-score, which surpasses the best classification results in [23]. To get better classification, a larger dataset for Pre-Islamic, Islamic and Andalusian era are needed.

TABLE IV
PERFORMANCE OF TRI-CLASS CLASSIFIERS

Classifier	**Accuracy**	**Average Weighted $F1$-score**
Baseline classifier		0.856
Baseline w/o frequent terms		0.857
CNN	**0.875**	**0.875**
CNN w/o frequent terms	0.870	0.870
CNN with class weights	0.872	0.873
CNN w/o frequent terms with class weights	0.830	0.826
CNN with under-sampling	0.822	0.825
CNN with over-sampling	0.843	0.846

TABLE V
PERFORMANCE OF MULTI-CLASS CLASSIFIERS

Classifier	**Accuracy**	**Average Weighted $F1$-score**
Baseline classifier		0.779
Baseline w/o frequent terms		0.779
CNN	0.796	0.777
CNN w/o frequent terms	0.798	0.794
CNN with class weights	0.801	0.796
CNN w/o frequent terms with class weights	0.799	0.796
CNN with under-sampling	0.653	0.654
CNN (under-sampling largest 2 classes)	0.688	0.687
CNN with under-sampling and class weights	0.690	0.687
CNN with under- and over-sampling	0.671	0.672

VI. DISCUSSION

In the polarity classification, the classifier was confident in identifying poems from modern era. This shows that the poems of the Modern era use significantly different terms. In all other classifiers, poems of Modern era were also positively classified. The mislabeled poems in the tri-class classification are relatively few, and most of which are poems from the Pre-Islamic and Islamic eras that were classified as poems from Andalusian and Abbasid era. This is a result of two main reasons, first, that the dataset for the Pre-Islamic and Islamic eras is small. Second, because the terms used in these eras are significantly different from terms used in the Modern era.

TABLE VI
PERFORMANCE OF MULTI-CLASS CLASSIFIERS PER CLASS

Classifier	**$F1$-score for each era**				
	Pre-Islamic	Islamic	Abbasid	Andal-usian	Modern
Baseline classifier	0.006	0.405	0.745	0.526	0.894
Baseline w/o frequent terms	0.006	0.385	0.746	0.543	0.892
CNN	0.000	0.514	0.765	0.375	0.913
CNN w/o frequent terms	0.223	0.594	0.763	0.495	0.909
CNN + class weights	0.422	0.592	0.760	0.488	0.909
CNN + class weights w/o frequent terms	0.303	0.533	0.771	0.488	0.911
CNN with under-sampling	0.668	0.609	0.542	0.623	0.829
CNN + under-sampling the largest 2 classes	0.328	0.657	0.628	0.665	0.858
CNN + under-sampling & class weights	0.329	0.669	0.604	0.684	0.855
CNN + under-& over-sampling	0.433	0.601	0.626	0.599	0.865

The results of the multiclass classification are expected, due to the high imbalance in the classes. The confusion matrix triggers some interesting findings. Most of the poems from the Andalusian Era is mislabeled as Abbasid era, as seen in TABLE II These two eras are overlapping, and the only difference is the geographical location of the poems' origin. Every two consecutive eras, except Modern era, have some mislabeled poems. Examples of mislabeled poems are shown in TABLE VII. The Pre-Islamic and Islamic that are mislabeled share some words and both talk about social issues. Similarly, the Andalusian and Abbasid, both share the same topic and even exact words.

The results show that in polarity classification, the classifier is sound and could successfully identify if a poem belongs to the Modern Era due to the significant difference in terms used in the Modern Era compared to prior Eras. By zooming in to the Non-Modern Eras, results show that the classifier was slightly confused in determining the era of a poem of every two consecutive eras in the tri-class and five-class classifiers. A possible conclusion is that used words and topics are similar in every two consecutive eras. Removing stop words was not effective in multiclass classifications, while removing frequent terms slightly enhanced the performance of polarity classifier.

VII. CONCLUSION

Arabic text classification is a challenging task because of the complicated inflections and several variations in Arabic language. In this paper, we use deep learning to build a model that can classify classical Arabic poems based on their era, which has not been done before. Besides, we present a refined dataset that accounts for Modern era poem. Three models

TABLE VII. MISCLASSIFIED EXAMPLES

Poem	Predicted era	Actual era
هيهات منك بنو عمرو ومسكنهم إذا تشتيت قنسرين أو حلبا	Pre-Islamic	Islamic& Umayyad
نحن بنو عمرو بن بكر بن وائل نحالفهم ما دام للزيت عاصر	Islamic & Umayyad	Pre-Islamic
لله خال على خد الحبيب له في العاشقين كما شاء الهوى عبث أورثته حبة القلب القتيل به وكان عهدي أن الحال لا يرث	Abbasid	Andalusian
يا مدع أن الغرام بقلبه أفنى تجلده وطار بلبه من كان في دعوى المحبة صادقا أخفى الحبيب ولن يبوح بحبه أيروم وصل محجب من دونه بيض تسل بأسود من هدبه هيهات مت كمدا بما قد ضم منك الحشا واخف الهوى أو ذع به	Andalusian	Abbasid

were built, a polarity classifier where poems are classified into Modern and Non-Modern eras. A tri-class classifier where every two consecutive eras are grouped, except the Modern era. And a multi-class classifier that classify the model into Pre-Islamic, Islamic and Umayyad, Abbasid, Andalusian and Modern eras. The polarity classifier achieved highest $F1$-score of 0.914 by removing frequent terms. The tri-class classifier achieved highest $F1$-score of 0.875, and the multiclass classifier surpassed the existing classifiers from previous works with highest $F1$-score of 0.796 by assigning class weights. Yet, this classifier struggles to categorize the classes of small size because of the high imbalance in the dataset. A good approach for future work can be exploiting the Arabic diacritization and the prosody of Arabic poetry as features.

REFERENCES

[1] R. Stuart and N. Peter, "Artificial intelligence-a modern approach 3rd ed," 2016.

[2] A. Al-Alwani and M. Beseiso, "Arabic spam filtering using Bayesian model," *Inter. Journal of Computer Applications*, 79(7), 2013.

[3] Y. Li, X. Nie, and R. Huang, "Web spam classification method based on deep belief networks," *Expert Systems with Applications*, vol. 96, pp. 261–270, 2018.

[4] A. Elnagar, "Investigation on sentiment analysis for arabic reviews," in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, 2016, pp. 1–7.

[5] A. A. Altowayan and A. Elnagar, "Improving arabic sentiment analysis with sentiment-specific embeddings," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 4314–4320.

[6] A. Elnagar and O. Einea, "Brad 1.0: Book reviews in arabic dataset," in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*. IEEE, 2016, pp. 1–8.

[7] A. Elnagar, Y. S. Khalifa, and A. Einea, "Hotel arabic-reviews dataset construction for sentiment analysis applications," in *Intelligent Natural Language Processing: Trends and Applications*. 2018, pp. 35–52.

[8] A. Elnagar, L. Lulu, and O. Einea, "An annotated huge dataset for standard and colloquial arabic reviews for subjective sentiment analysis," *Procedia computer science*, vol. 142, pp. 182–189, 2018.

[9] N. Boudad, R. Faizi, R. O. H. Thami, and R. Chiheb, "Sentiment analysis in arabic: A review of the literature," *Ain Shams Engineering Journal*, vol. 9, no. 4, pp. 2479–2490, 2018.

[10] A. Dahou, S. Xiong, J. Zhou, M. H. Haddoud, and P. Duan, "Word embeddings and convolutional neural network for arabic sentiment classification," in *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*, 2016, pp. 2418–2427.

[11] L. Al Qadi, H. El Rifai, S. Obaid, and A. Elnagar, "Arabic text classification of news articles using classical supervised classifiers," in *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*. IEEE, 2019, pp. 1–6.

[12] L. Lulu and A. Elnagar, "Automatic arabic dialect classification using deep learning models," *Procedia Computer Science*, vol. 142, pp. 262 – 269, 2018, arabic Computational Linguistics.

[13] A. Elnagar and S. Harous, "Recognition of handwritten Hindu numerals using structural descriptors" *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 15(3), pp. 299 – 314, 2003.

[14] A. Elnagar, R. Ismail, B. Alattas, and A. Alfalasi, "Automatic classification of reciters of quranic audio clips," in *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, 2018, pp. 1–6.

[15] A. Elnagar and M. Lataifeh, *Predicting Quranic Audio Clips Reciters Using Classical Machine Learning Algorithms: A Comparative Study*, 2020, pp. 187–209.

[16] M. Lataifeh, A. Elnagar, I. Shahin, and A. B. Nassif, "Arabic audio clips: Identification and discrimination of authentic cantillations from imitations," *Neurocomputing*, 2020.

[17] B. Elayeb, "Arabic word sense disambiguation: a review," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2475–2532, 2019.

[18] O. Einea, A. Elnagar, and R. Al Debsi, "Sanad: Single-label arabic news articles dataset for automatic text categorization," *Data in Brief*, vol. 25, p. 104076, 2019.

[19] A. Elnagar, O. Einea, and R. Al-Debsi, "Automatic text tagging of arabic news articles using ensemble learning models"," *Proceedings of the 3rd International Conference on Natural Language and Speech Processing*, pp. 59–66, 2019.

[20] A. Elnagar, R. Al-Debsi, and O. Einea, "Arabic text classification using deep learning models," *Information Processing & Management*, vol. 57, no. 1, p. 102121, 2020.

[21] E. Caplan, "What are the most spoken languages in the world?" Aug 2020. [Online]. Available: https://www.worldatlas.com/articles/most-popular-languages-in-the-world.html

[22] D. Shawqi, *History of Arabic Literature*. Islamic Books, 1982.

[23] M. Abbas, M. Lichouri, and A. Zeggada, "Classification of arabic poems: from the 5^{th} to the 15^{th} century," in *International Conference on Image Analysis and Processing*. Springer, 2019, pp. 179–186.

[24] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[25] A. Almuhareb, W. A. Almutairi, H. Al-Tuwaijri, A. Almubarak, and M. Khan, "Recognition of modern arabic poems." *JSW*, vol. 10, no. 4, pp. 454–464, 2015.

[26] A. Almuhareb, I. Alkharashi, L. A. Saud, and H. Altuwaijri, "Recognition of classical arabic poems," in *Proceedings of the Workshop on Computational Linguistics for Literature*, 2013, pp. 9–16.

[27] A. Ahmed, R. Mohamed, B. Mostafa, and A. Mohammed, "Authorship attribution in arabic poetry," in *10th International Conf. on Intelligent Systems: Theories and Applications (SITA)*. IEEE, 2015, pp. 1–6.

[28] A. Ahmed, R. Mohamed, and B. Mostafa, "Machine learning for authorship attribution in Arabic poetry," *Int. J. Future Comput. Commun*, vol. 6, no. 2, pp. 42–46, 2017.

[29] A. Al-falahi, M. Ramdani, M. Bellafkih, "Use smo svm, lda for poet identification in arabic poetry," in *International Conference on Advanced Information Technology, Services and Systems*. Springer, 2018, pp. 161–169.

[30] S. Al-Harbi, A. Almuhareb, A. Al-Thubaity, M. Khorsheed, and A. Al-Rajeh, "Automatic arabic text classification," 2008.

[31] I. A. Mohammad, "Naive bayes for classical arabic poetryclassification," *Al-Nahrain Journal of Science*, vol. 12, no. 4, pp. 217–225, 2009.

[32] O. Alsharif, D. Alshamaa, and N. Ghneim, "Emotion classification in arabic poetry using machine learning," *International Journal of Computer Applications*, vol. 65, no. 16, 2013.

[33] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[34] F. Chollet, "keras-team/keras," 2015. [Online]. Available: https://github.com/fchollet/keras