IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2020, accepted July 9, 2020, date of publication July 14, 2020, date of current version July 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009217

Impact of Stemming and Word Embedding on Deep Learning-Based Arabic Text Categorization

HUDA ABDULRAHMAN ALMUZAINI AND AQIL M. AZMI
Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
Corresponding author: Aqil M. Azmi (aqil@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University through the initiative of DSR Graduate Students Research Support.

ABSTRACT Document classification is a classical problem in information retrieval, and plays an important role in a variety of applications. Automatic document classification can be defined as content-based assignment of one or more predefined categories to documents. Many algorithms have been proposed and implemented to solve this problem in general, however, classifying Arabic documents is lagging behind similar works in other languages. In this paper, we present seven deep learning-based algorithms to classify the Arabic documents. These are: Convolutional Neural Network (CNN), CNN-LSTM (LSTM = Long Short-Term Memory), CNN-GRU (GRU = Gated Recurrent Units), BiLSTM (Bidirectional LSTM), BiGRU, Att-LSTM (Attention-based LSTM), and Att-GRU. And for word representation, we applied the word embedding technique (Word2Vec). We tested our approach on two large datasets–with six and eight categories–using ten-fold cross-validation. Our objective was to study how the classification is affected by the stemming strategies and word embedding. First, we looked into the effects of different stemming algorithms on the document classification with different deep learning models. We experimented with eleven different stemming algorithms, broadly falling into: root-based and stem-based, and no stemming. We performed ANOVA test on the classification results using the different stemmers, which helps assure if the results are significant. The results of our study indicate that stem-based algorithms perform slightly better compared to root-based algorithms. Among the deep learning models, the Attention mechanism and the Bidirectional learning gave outstanding performance with Arabic text categorization. Our best performance is F-score = 97.96%, achieved using the Att-GRU model with stem-based algorithm. Next, we looked into different controlling parameters for word embedding. For Word2Vec, both skip-gram and bag-of-words (CBOW) perform well with either stemming strategies. However, when using a stem-based algorithm, skip-gram achieves good results with a vector of smaller dimension, while CBOW requires a larger dimension vector to achieve a similar performance.

INDEX TERMS Arabic document classification, deep learning, stemming strategies, word embedding, statistical significance.

I. INTRODUCTION

The Internet is full of information in many different forms, including millions of textual documents. This large volume of data posts a challenge, even for simple tasks, such as information retrieval (IR). A possible solution is to organize the textual data into different categories. Manual classification is out of question, and the alternative is to automate the task. Automatic document classification is used to discover the basic information of documents automatically, thus saving human time and effort. Automatic text classification (TC), or document classification which we use interchangeably, is the assignment of the document to a pre-determined set of categories based on its contents.

With a population of around 445 million, Arabic language users constitute the fastest-growing language group with regards to the number of Internet users. According to a study in (www.internetworldstats.com/stats7.htm), during the last two decades, Arabic language Internet users have grown by 9348%. In the same statistics, the next two language

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Zakarya.

user groups experienced a growth of 3653.4% and 3356% (for Russian and Indonesian languages, respectively) in the same period. While most of the research to date has tackled the problem for the English language, the work on Arabic document classification is lagging behind. There are many reasons for this, as we will see in Section II-B. Needless to say, many authors have concluded that constructing automatic Arabic TC is a challenge [1]–[3].

Developing a classification system for the Arabic language involves understanding the syntactic structure of the words, so that we can manipulate and represent the words in a way that makes their classification more precise. The pre-processing step (e.g., removing stop-words, or stemming) plays a critical role in many Arabic natural language processing (NLP) applications, including classification. Moreover, it helps reduce the dimensionality and thus reduce the classification time.

Deep learning (DL) is a sub-area of machine learning that uses multi-layered artificial neural networks to deliver high accuracy in diverse tasks such as object detection, speech recognition, natural language processing, etc. Word2vec is a word embedding technique that uses a shallow neural network [4]. Word2Vec maps words into continuous vectors, and as it turns out, it is one of the most popular models for word similarity tasks, and text classification. It has been successfully used for raw text classification in English [5]–[7], and other languages, such as Chinese [8], [9].

Given the importance of stemming and word embedding to TC, we would like to see their impact on textual classification using DL algorithms. This study involves seven different DL models: Convolutional Neural Network (CNN), CNN-Long Short Term Memory (CNN-LSTM), CNN-Gated Recurrent Units (CNN-GRU), Bidirectional LSTM (BiLSTM), Bidirectional GRU (BiGRU), Attention-based LSTM (Att-LSTM), and Attention-based GRU (Att-GRU); and eleven different stemming strategies broadly falling into root-based, stem-based, or no stemming. We are not aware of any comparable study that combined so many DL models along with so many stemming algorithms to assess a single Arabic NLP application. Summarizing our contributions:

- We explore how the TC using the different DL models for the documents in the Arabic language is affected by the different stemming strategies.
- We investigate which of the studied DL models is best suited for the task of Arabic TC.
- We studied the impact of word embedding, using Word2Vec, and how this improves TC.
- We use analysis of variance (ANOVA) test to confirm the significance of the TC results.

More specifically, we conducted two groups of experiments, one for stemming strategies, and the other to identify appropriate parameters for Word2Vec. When conducting an experiment on, for example stemming strategies, we fixed the parameters of word embedding. The same goes for the other experiment, where we fixed the stemming algorithm to one from each stemming strategy. As we will discuss later (see Section III) most of the studies limit their comparison by using a single candidate algorithm from each stemming approach. On the other hand, in this study, we compared the performance of TC using ten different stemming algorithms, five of which are root-based [10]–[14], five are light stemmer [15]–[19], and no stemming. As we stated earlier, stemming impacts the performance of NLP applications, TC included. We experimented using two large corpora, and among the different DL models, only CNN-LSTM and CNN-GRU where effected by the stemming algorithms. So, we have an interesting conclusion. Stemming had little or no impact on the performance of classifying Arabic documents when using some of the deep learning models. We achieved our best performance for Arabic TC using Att-GRU, BiGRU, and BiLSTM, in order.

The rest of this paper is organized as follows: Section II presents the background about text classification, and stemming algorithms. In Section III, we review related studies. In Section IV, we discuss our proposed system. Section V is a discussion of the experiments and results. Finally, Section VI concludes the paper, and outlines possible future work.

II. BACKGROUND

In this Section we briefly outline the textual classification, challenges facing the classification when using Arabic language, and stemming algorithms.

A. TEXTUAL CLASSIFICATION

Text classification (TC)—also text tagging, or text categorization—is the process of classifying text into organized groups. It uses NLP to automatically analyze text, and then assigns a set of pre-defined tags or categories based on the content. Some of the areas that may benefit from TC are: sentiment analysis, topic detection, and language identification etc.

Classifying a text involves three stages. First, the pre-processing step. This step usually requires cleaning the text (e.g., removing punctuation mark, stop word, numerals). Stemming is applied at this step. The second step involves representing the document in vector form to extract its features. Different techniques have been proposed, such as Latent Semantic Analysis [20], Bag-of-Words [21], and Word2Vec [4], or at character level using n-gram [22]. In the third step, we train and test the classifier. There are two approaches for training the classifier, the traditional method and the deep learning method. In the traditional method, following the conversion of the documents to feature vector, we use a typical classifier (e.g., Support Vector Machine, K Nearest Neighbors, Naïve Bayes). More recently, deep learning (DL) methods have received considerable attention for the classification task. In DL architecture, a multi-layer neural network are used, where the input is the document features vector. One drawback is that DL requires large training data to give satisfactory results. Examples of DL models that achieved remarkable classification results

TABLE 1. Example of an Arabic word ليفاوضونهم which has different affixes attached to a root word, "negotiate". The full meaning "to negotiate with them". Source: [28].

هم *hm*	ون *wn*	فاوض *fAwD*	ي *y*	ل *l*
pronoun "them"	termination of conjugation	negotiate	letter for tense and the person of conjugation	preposition "to"
Postfix	Suffix	Core	Prefix	Antefix

are: the CNN [5], character level CNN [23], and Deep Belief Networks (DBN) [24].

A subproblem of TC is hierarchical textual classification, where the document is classified into a predefined multi-level categories. The hierarchical TC aims at organizing the mass of information as a tree structure in which a document that belongs to a topic at a certain level also belongs to all of its parent topics, ancestors, etc [25]. For example, under a 3-level categorization, a document may be classified as "health → diseases → cancer", another document is classified as "health → diseases → heart" etc. There are advantages for such hierarchical classification, such as improving retrieval time. Hierarchical TC is outside the scope of our work.

B. CHALLENGES TO CLASSIFYING DOCUMENTS IN ARABIC

Developing an accurate system for categorizing text from the large number of Arabic documents accessible on the Internet is very challenging. These challenges arise from the ambiguity due to the lack of diacritical markings in Modern Standard Arabic (MSA), the Arabic language's rich and complex morphology, the wide spread use of synonyms, the nature of the language itself—Arabic is a highly inflectional and derivational language—etc.

The Arabic orthographic system uses small diacritical markings to represent different short vowels. There are a total of thirteen different diacritics, and these are used to clarify the sense and meaning of the word. In MSA, the written text is devoid of these markings, as it is assumed the reader will disambiguate the meaning. However, this is not true for the machines [26]. Just to give an idea, consider the undiacritized word عقد. It has more than one meaning depending on the diacritics, "necklace", "knots", "contract", "decade", "pact", and "complicated". Some of the words share the exact same diacritical marking, but have different meaning which can only be realized through context. For example, the word عَام means "year", but it may also mean "public".

In addition, the plural, dual, and singular forms in Arabic vary according to gender. In Arabic, there are linguistic rules for each type, and some words have irregular plural forms. Moreover, the letter و *waw* at the beginning of an Arabic word poses a challenge, since it may be the proposition "and" or an original letter part of the word. For example, the letter waw is proposition in وجلس "and sat", while it is original lexeme in وقف "stood up".

C. STEMMING ALGORITHMS

Commonly referred to as stemmers. Stemming is a computational procedure which reduces all words with the same root (or the same stem, in case prefixes are left untouched) to a common form, usually by stripping each word of its derivational and inflectional suffixes. A stemming algorithm reduces the words "chocolates", "chocolatey", or "choco" to the root word, "chocolate"; and the words "retrieval", "retrieved", "retrieves" are reduced to the stem "retrieve". Here, we want to reduce different forms of a word to a core root or stem. This provides more convenience when handling words that share the same core meaning, thus playing an important role in the field of information retrieval (IR). In IR, grouping words with the same root (or stem) increases the success with which documents can be matched against a query [27].

For the English language, a simple stemming that involves the stripping of suffixes is sufficient for the purpose of IR. For Arabic, however, the stripping of suffixes alone would not be sufficient [29]. In Arabic, there are four kinds of affixes: antefixes, prefixes, suffixes and postfixes that can be attached to words [28]. Table 1 provides an example of a complex Arabic word with all affixes. There are two main stemming approaches in Arabic: root-based stemming, and stem-based (or light) stemming.

In the root-based stemming technique we perform heuristic and linguistic morphological analysis to extract the root of a word. This technique can be further divided into three categories: dictionary-based, nondictionary-based, and hybrid (see Figure 1). An example of dictionary-based is the Khoja stemmer [11], which uses—as the name implies—a dictionary file of Arabic roots. The nondictionary-based algorithms are further classified into three different approaches: pattern-based, statistical-based, and rule-based. The pattern-based algorithm uses the Arabic pattern (or a template) to match a word, then extract the root. For instance, the word مدرسة "school" matches the pattern مفعلة, resulting in the triliteral root د ر س: d r s. Some of the algorithms that fall under this approach are [12], [30], [31]. Ref [32] developed an algorithm that is statistical-based stemmer. The algorithm uses the idea of assigning weights to the letters in order to extract the root without consulting lists of prefixes, suffixes, patterns, or roots. Then, we have algorithms that utilize linguistic rules to extract the root, such as those in [10], [14], [33], [34]. The last approach under the root-based stemmer

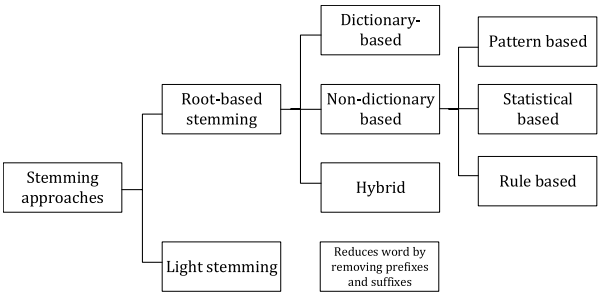


FIGURE 1. Stemming approaches.

is the hybrid method. In hybrid methods, we extract the root using a combination of rules, patterns, and/or lookup dictionary of roots, e.g. [13], [35], [36].

In light stemming we reduce the word by removing prefixes and suffixes. This method does not deal with patterns and infix (letters added within the root). Occasionally, the resultant word may not be a valid one, but it is sufficient for the objective of IR. For example, دراسات ''studies'' whose stem is دراس, is not a valid Arabic word. Works that tackle light stemming includes [15]–[19], [37], [38]. The author in [17] uses snowball, a programming language dedicated for stemming in different human languages. Ref [18], on the other hand, used lexicon resources to improve the stemming.

III. RELATED WORK

For the Arabic language, most of the stemming algorithms have been tested on IR, but few works have looked at their impact on automatic TC. Arabic is a morphologically rich language, and from a single root we can derive many different words. For example, from the root letters د ر س:d r s we can drive the words درس ''study'', مدرسة ''school'', مدرس ''teacher'' (masc.), مدرسات ''teachers'' (fem.), مدرسين ''teachers'' (masc.) etc. An analysis of Arabic text in a newspaper indicates that there are more words occurring once and there are more distinct words than those found in English text of identical size, when no stemming is involved [39]. Given that, extracting the roots of the words found in a document will reduce the dimensionality, and may improve the accuracy of IR. However, many word variants do not share the same semantic meaning even though they may share the root. Thus, in the pre-processing stage for IR, the root extraction methods may increase a word's ambiguity, in which case the light stemming methods may be a better choice. Ref [15] investigated the effectiveness of two different stemming techniques of Arabic texts on IR. The first technique was root-based, and the second technique, was light stemming. For the first technique, they used a modified Khoja's algorithm [11] to extract the roots; and for the second technique the authors used their own light stemmer. The latter technique relied on removing the most frequently occurring suffix and prefix, and normalization. The authors evaluated the performance of no stemming, light stemming, and root-based stemming algorithms. For an Arabic query system, they reported a degraded performance when no stemming is involved, while light stemming significantly boasted the system performances compared to one using the root-based stemmer.

In [40], the authors investigated the impact of the root-based stemmer vs light stemmer for text mining tasks. In the pre-processing step, the authors applied [11] and [37] stemming algorithms, respectively. They also used the Latent Semantic Analysis (LSA) model for measuring the semantic similarity between Arabic words. The experiments demonstrated that using light stemming improved the performance compared to using the root-based stemming algorithm. The authors attributed this to the occasional loss of sense when words were reduced to their root form.

Ref [1] studied the effectiveness of light stemming vs heavy stemming (root-based) for Arabic text categorization. For the experiment, the authors used a dataset of 15,000 documents classifying them into three categories. Using the K-Nearest Neighbors (KNN) classifier, the authors experimented with both the heavy stemmer [32] and the light stemmer [15]. They concluded that better accuracy was achieved when light stemming was used as opposed to heavy stemming.

Reference [24] proposed a three-stage technique for classifying Arabic documents into multiple categories. A root extraction algorithm was applied in the pre-processing stage. Then, a combination of Markov and fuzzy C-means was used for clustering. In the third stage, the DBN was used to build the Arabic classification model for each resulting cluster. The experiment was conducted on 12,000 randomly selected documents from two different datasets. The authors reported an F-score of 91.02%.

Reference [41] studied the effect of stemming techniques on Arabic document classification. For the pre-processing step, the authors picked three stemmers, one root-based [30], and two stem-based [19], [42]. They used traditional classification algorithms, namely, Naïve Bayesian (NB), Support Vector Machines (SVM), and KNN. The experiments were performed on open source Arabic corpora (OSAC) [43]. The corpus consists of 5,070 documents divided into six categories. The best performance of micro-$F_1 = 94.64\%$ was reported using SVM with stemmer in [19].

To encode documents for classification, [44] utilized the Restricted Boltzmann Machine (RBM). For the pre-processing step, the authors applied Khoja's stemmer [11], and [45] a light stemmer. After training the document representations using RBM, they classified the documents using Decision Tree (DT), NB, and SVM. They used OSAC corpus [43], the authors reported their best accuracy of 75.1% using light stemmer.

Learning effective document representation can enhance document classification. In [46], the authors proposed a technique that combines document embedding representation with Arabic WordNet to learn the word sense disambiguation. For the pre-processing step, the root of the words was extracted using the Khoja stemmer [11]. After learning

the documents representation, the documents were classified using the multi-layer perceptron classifier. This proposed method yield an F-score of 90%.

The proposed technique in [47] focused on feature selection for Arabic TC. They applied the modified Khoja's stemmer, followed by four different feature selection metrics (Chi-square, information gain, mutual information, and improved Chi-square) to select the best features. The size of features ranged between 20 and up to 1400. For classification they used DT and SVM. The authors tested their scheme on OSAC corpus [43], and reported their best performance using improved Chi-square feature selection, achieving F-score of 90.50%. This was achieved when using 900 features. When less or more features are selected, the F-score drops. For instance, the F-score $=$ 83.8% and 88% for 100 and 1400 features respectively, using improved Chi-square.

Some recent researches explored the effects of different stemming approaches on TC and text mining tasks. In [48] compared three different stemmers [11], [16], [49] on TC task. For classification the authors used SVM and NB. The system was tested on 2,000 documents collected from royanews.com. They reported the best result of F-score $=$ 92% using the stemmer [49].

The study in [50] used the stemmer [51] to assess Arabic TC task. They used TF-IDF to extract the features from the documents. For classification, the authors used Logistic Regression, SVM, and CNN. The system was tested on 111,728 documents collected from three different online newspapers. They reported their highest accuracy score of 92% using the CNN model. While [52] compared two stemmers [49], [53] to see which one is better suited for Arabic TC. Using two different classifiers, SVM and NB, and a dataset of 1000 news articles from alghad.com, they reported their best performance of F-score $=$ 90% when used with the stemmer [49].

In [54] devised a TC model to detect violence in Arabic tweets using different feature reduction methods. For classification they used KNN, Bayesian boosting, and bagging SVM. In addition, the authors used two different stemmers (a heavy and a light stemmer), and n-gram words without stemming. They collected a total of 12,500 tweets covering four different regions of Saudi Arabia. Experimentally, the authors reported their highest accuracy of 86.61% using SVM bagging with tri-gram. A further boosting of accuracy to 90.59% was achieved using information gain and some reduction features.

Reference [55] studied the impact of stemming on sentiment analysis using SVM, NB, and Maximum Entropy. For the sake of comparison, the authors used different stemmers [11], [16], [30], [42], [56]. Their highest reported precision was 90%.

Some of the research studies did not use stemming at all, such as [57], who instead utilized the Part of Speech (POS) feature. After several experiments, they concluded that a higher number of features allowed them to reach a higher classification score. The POS method achieved a classification accuracy of 91% when the number of features was 2,000.

Reference [25] used Markov chains to solve the problem of hierarchical Arabic TC into three-level deep categories (see Section II-A). The top level had eight categories. The authors used a corpus containing 11,191 documents compiled from Alqabas newspaper. All the documents were at last 800 characters long. The corpus was split into 9711 documents ($\approx$ 86.8%) for training, and 1480 documents for testing. The authors reported an accuracy of 90.29% for the first level categorization, with subsequent levels having accuracy of 77.09% and 63.33% (respectively).

Table 2 summarizes all the reviewed works. The above studies confirm that the preprocessing step is a challenging and a crucial stage when dealing with Arabic documents. Stemming is likely to impact the quality of classification. Though there are so many stemmers for the Arabic language, with new ones consistently being devised, in the end most of the published works that studied the effect of stemming picked a single candidate algorithm, one from each approach. A root-based stemming, which is typically Khoja [11], and another for light stemming, mostly Light10 [16].

IV. PROPOSED SYSTEM

One of our objectives was to automatically classify the Arabic documents into a set of pre-defined categories. For the pre-processing step, we experimented with eleven different stemmers, five of which are root-based stemmers, five are stem-based (light) stemmers, and the no stemming. Three of the five root-based stemmers were recommended by [58] as they are known to outperform others. These three are [10]–[12]. The other two, namely those by [13], [14], were proven to perform well in different NLP tasks. For light stemming we picked those algorithms that were developed recently, such as [15]–[19]. For our experiment we contacted the authors of all the aforementioned stemmers to share the source code. Only three agreed and shared the source, [11], [17], [18] for which we are grateful. ARLSTem [19], source is also freely available, but as it is in python, we re-implemented it in java. As for the other six stemming algorithms, we implemented the stemmers based on the description found in their respective published works, namely [10], [12]–[16].

Algorithm 1 Framework of Our Proposed System

Input: Raw document D

```
begin
    Clean document D (remove special symbols,
    stop-words, numerals, etc)
    Apply stemming algorithm on D
    Create a list of distinct words in document D
    Use word embedding by transforming D into feature
    vector using Word2Vec model
    Classify D using a deep learning model
end
```

TABLE 2. Summary of related works along with the list of stemmers used therein.

Ref.	Stemmers evaluated	Application	Classifier	Data set	Results
[1]	Root-based stemmer [32], and light stemmer [15]	Arabic TC	KNN	15K documents, 3 categories	Light stemming yield better classification
[40]	Root-based stemmer [11], and light stemmer [37]	Text mining	Latent semantic analysis	Two data sets both from SPA website with 252 and 257 documents	Reducing the word to its root form affects the semantic of the word
[24]	Root extraction algorithm using letter weight	Arabic TC	Deep belief network	12K documents from Al-Jazeera and SPA	Deep learning model with root extraction outperforms traditional ML models without root extraction
[41]	Root-based stemmer [30], and Light stemmer [19], [42]	Arabic TC	SVM, NB, and KNN	5,070 documents from OSAC, 6 categories	SVM + length stemmer [19] outperformed other models
[44]	Root-based stemmer [11], and light stemmer [45]	Arabic TC	SVM, DT, and NB	5,070 documents from OSAC, 6 categories	Best accuracy reported for SVM + light stemmer
[46]	Root-based stemmer [11]	Arabic TC	SVM, and multi-layer perceptron	22,428 documents from OSAC, 10 categories	Word2Vec with multi-layer perceptron outperform traditional method
[47]	Root-based stemmer, a modification of [11]	Arabic TC	SVM, and DT	5,070 documents from OSAC, 6 categories	Best accuracy reported for SVM with improved Chi-square
[48]	Root-based stemmer [11], and light stemmer [16], [49]	Arabic TC	SVM, and NB	2K documents from royanews.com	Light stemmer [49] beats the other two
[50]	Light stemmer [51]	Arabic TC	SVM, Logistic Regression, and CNN	111,728 documents from 3 online newspapers, 5 categories	CNN outperforms SVM
[52]	Light stemmer [49], [53]	Arabic TC	SVM, and NB	1000 news articles from alghad.com	Stemmer [49] beats [53]
[54]	Root-based stemmer [11], light stemmer [37], and no stemming with n-gram	Text mining	SVM, KNN, Bayesian boosting, and Bagging	12,500 tweets in 3 classes	Bagging SVM with tri-gram outperform others
[55]	Root-based stemmers [11], [30], [42], and light stemmer [56]	Sentiment analysis	SVM, NB, and Maximum Entropy	Three data set of 33K, 9K, and 40K tweets	Light stemming outperform root-based approach with majority voting of trained classifiers

For the Arabic document classification, we implemented seven different deep-learning models. For a thorough comparison of stemmers, we evaluate the model's classification performance using the different stemmers. Algorithm 1 summarizes our proposed system. There are three main steps: preprocessing, word embedding that transforms the document into feature vector using Word2Vec model, and finally a deep learning-based classifier. We go over each step in more detail below.

A. PREPROCESSING

In this step, we cleaned the text by removing punctuation marks, stop-words, numerals, non-Arabic words, and single letter words. Then, we applied the stemmer. Each word in the document was reduced to its root (or stem) form based on stemming algorithm. This was followed by creating a dictionary of all distinct words in the document. The dictionary was used to encode the documents in embedding model. Compared to light stemming, using the root-based stemmer results in a lower number of words in the dictionary. More words in the dictionary means a larger document representation. Table 6 shows the number of words in the dictionary using different stemming algorithms.

B. WORD EMBEDDING

Word embedding is a technique used to represent the words of the document, where each word—in the vocabulary—is represented by a real valued vector. In this representation, words with similar meaning will have a similar representation. For word embedding, we used Word2Vec [4]. It is an efficient algorithm in terms of space and time. Word2Vec is a two-layer neural network, where the input is the document and the output are a continuous feature vector of a pre-specified dimension. There are two main learning algorithms in Word2Vec: continuous skip-gram or continuous bag-of-words (CBOW). In the skip-gram method, we predicted the surrounding context words given the center word,

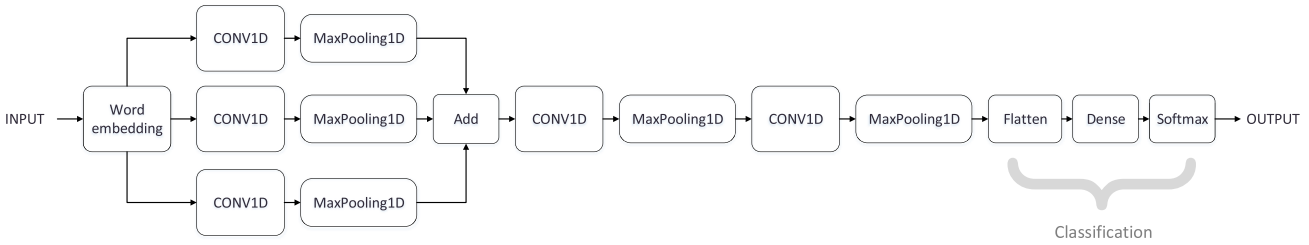


FIGURE 2. Neural network with three convolutional layers for document classification. Conv1D is the convolutional layer, and for pooling layer we use max pooling.

while in CBOW we predicted the current word using a window of surrounding words. For instance, in the CBOW model we maximized the probability of a word being in a specific content in,

$$\Pr(w_i \mid w_{i-d}, w_{i-d+1}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+d-1}, w_{i+d}), \tag{1}$$

where w_i is a word at position i and d is the size of the window. Thus, it yields a model that is contingent on the distributional similarity of words. The dimension of word embedding can vary based on different applications, and it usually ranges from 50 to 300.

C. DEEP-LEARNING MODELS

In this Section we will take a short glimpse at the different deep learning models that we used for Arabic TC. Altogether, there are seven different models. In designing the different models, the first layer is the word embedding layer (see Section IV-B).

Convolutional Neural Network (CNN). Figure 2 shows our proposed model, consisting of three layers of CNN, and max pooling. Typically, CNN is made up of a sequence of layers, where the output of a layer feeds into the next layer. The layers are: convolutional layer, pooling layer, and the fully connected layer [59]. The embedding layer is followed by the CNN layer with three filters (each with a varying window sizes). Then, we have two CNN layers with a single filter, which are passed to a dense (fully connected) layer with soft-max activation whose output is the probability distribution over labels.

CNN-Long Short-Term Memory (CNN-LSTM). It combines CNN and forward LSTM. See Figure 3 for our proposed model. LSTM is a kind of Recurrent Neural Network that can capture sequential information, such as the context of a sentence [60]. The first Layer following the embedding layer is a single CNN layer which is followed by the pooling layer. The CNN is used for feature extraction of the input text. Then, the resulted features are passed to the single LSTM layer with 100 memory units and tanh activation to support sequence prediction. We perform 20% dropout that is followed by a dense layer with soft-max activation.

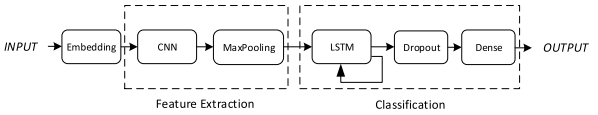


FIGURE 3. The CNN-LSTM (long short-term memory) model.

CNN-Gated Recurrent Units (CNN-GRU). The GRU is pretty similar to LSTM, but with less gates than a LSTM, making it a little speedier in the training process [61]. As we the mentioned in previous model, we utilize CNN as feature extraction layer with pooling. It is followed by the GRU layer with 60 memory cells and tanh activation, and finally, a dense layer with soft-max activation. There are no dropout in our implementation, since it works better without the dropout layer. Figure 4 illustrates our proposed model.

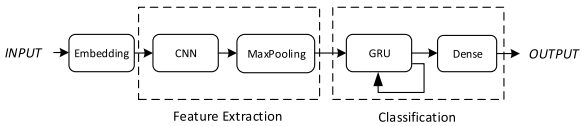


FIGURE 4. The CNN-GRU (Gated Recurrent Units) model.

Bidirectional LSTM (Bi-LSTM). The LSTM in CNN-LSTM is a forward LSTM, which can predict the class label based on the past (previous tokens). However, a word in the sentence is related to previous and next tokens. This makes it useful to learn the full context from both directions. The Bi-LSTM consists of two LSTMs, one to pass the text from left to right (forward LSTM), and another to pass the text from right to left (backward LSTM) [62]. This way, the model learns from past and future information. We implemented the Bi-LSTM using 100 memory units (forward and Backward LSTMs) after encoding the text with embedding layer. The output of those two LSTMs are fully connected to a dense layer with soft-max activation. For our proposed model, see Figure 5.

Bidirectional GRU (Bi-GRU). It is the same as Bi-LSTM but with the GRU layer instead of LSTM. Figure 6 shows our proposed model. After transforming the text to embedding representation, it is fed to the Bi-GRU

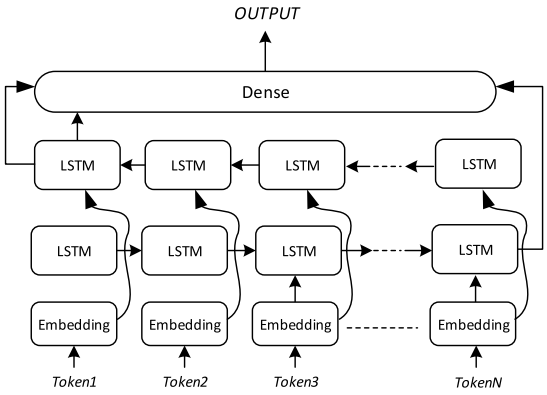


FIGURE 5. The bidirectional LSTM model. All inputs $Token_1, Token_2, \ldots$ to the model are stemmed.

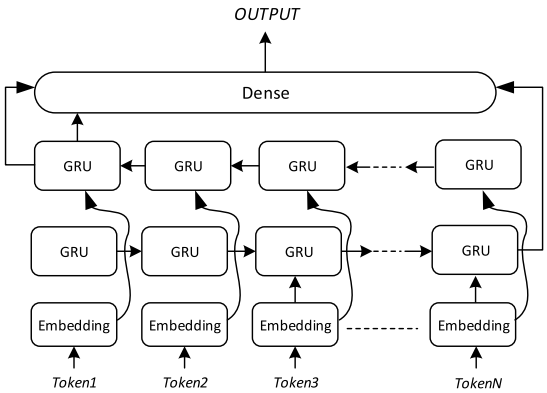


FIGURE 6. The bidirectional GRU model. All inputs are stemmed.

model with 100 memory cells connected to a dense layer with soft-max activation.

Attention-based LSTM (Att-LSTM). The attention is a great mechanism to concentrate on the useful information of the input data for a specific task, such as translation, visual identification of objects, text classification, etc. The attention model takes n arguments (which represent the hidden states of the LSTM h_s) and context vector c_t to generate the attention vector a_t. The attention vector a_t that contains the relevant part of the text is given by Eq. (2).

$$\begin{aligned} a_t &= f(c_t, h_t) = \tanh(W_c[c_t; h_t]), \\ c_t &= \sum_s \alpha_{ts} \bar{h}_s, \\ \alpha_{ts} &= \frac{\exp(score(h_t, \bar{h}_s))}{\sum_{s'=1}^{s} \exp(score(h_t, \bar{h}_{s'}))}, \end{aligned} \tag{2}$$

where h_t represents the last hidden state. The attention weights α_{ts} is calculated according to Luong's score [63] but with many-to-one attention (sequence to label instated of sequence to sequence in case of translation). Figure 7 is our implementation of the attention model described above. After the embedding layer, there is a single LSTM layer with 100 memory units and tanh activation. The resulted hidden states are connected to the attention model, and at the end there is a dense layer with soft-max activation.

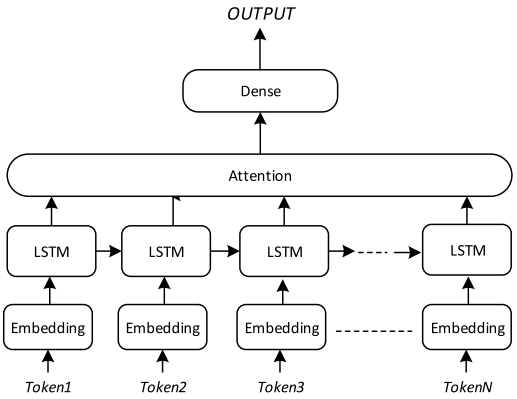


FIGURE 7. The attention-based LSTM model. All inputs are stemmed.

Attention-based GRU (Att-GRU). It is the same attention mechanism described in Att-LSTM. The GRU layer has 64 memory cells connected to the attention model. The output of the attention model is connected to a dense layer with soft-max activation, Figure 8.

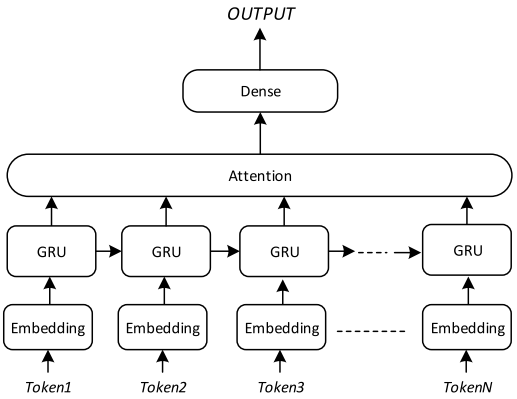


FIGURE 8. The attention-based GRU model. All inputs are stemmed.

V. EVALUATION AND RESULTS

For convenience we will break down this Section into subsections covering: (i) our data set, (ii) the setup for the experiments and the metrics used for the evaluation, (iii) experimenting with the different stemming algorithms, and (iv) experimenting with word embedding parameters.

A. THE DATA SETS

For the experiments we picked two different datasets. The first dataset is the ANT v1.1 (Arabic News Texts) Corpus [64],[1] and the second dataset is the Saudi Press Agency (SPA) corpus.[2] The ANT Corpus containing 10,161 documents containing a total of 1.474 million words,

[1] Compiled from Jawhara FM radio station in Tunisia. Free download from https://github.com/antcorpus/antcorpus.data/releases/tag/v1.1.

[2] The official Saudi Press Agency: http://www.spa.gov.sa. We downloaded the documents covering the four-month period starting in September 2018.

and is divided into nine categories: culture, diverse, economy, international news, local news, politics, society, sports, and technology. The SPA corpus covers six categories: general, culture, sports, economic, social, and politics. Table 3 summarizes the dataset used in this work. Though both corpora share some categories, each had its own. For example, the ANT corpus has the ''technology'' category, which is missing in the SPA corpus, while the latter has the ''general'' category which is not in ANT. Since the ANT corpus has nine categories versus six in SPA, we decided to drop one of the categories from ANT, and used only eight. In addition, ''local news'' is the largest category in the ANT corpus containing 4832 documents. We decided to use one-third of the documents in this category, in order to be in par with the ''international news'' category in the same corpus.

TABLE 3. Details of the ANT [64] and SPA corpora.

	Number of documents	
Category	ANT	SPA
Culture	124	2094
Economy	326	2400
International News	1260	—
Local News	1260	—
Politics	514	2558
Society	1087	1466
Sports	1460	2484
Technology	83	—
General	—	2400
Number of categories	8	6
Total # documents	6,114	13,402
Total # words	794,095	2,377,903
Mean # words/document	130	177

B. EXPERIMENTAL SETUP AND EVALUATION METRICS

We performed two sets of experiments: (a) evaluate the different stemmers, and (b) evaluate the word embedding. Further details follow.

(a) The first set of experiments looks at the effect of different stemming algorithms with the different deep learning models on the document classification task. The DL models are: CNN, CNN-LSTM, CNN-GRU, BiLSTM, BiGRU, Att-LSTM, and Att-GRU. For each DL model we conduct hyper parameter tuning experiment to find the best parameters to train the individual model. Then, we fix this parameter for that DL model. It will be the same parameter for all experiments involving the evaluation of the different stemming algorithms on that model. Table 4 summarizes the fixed parameters of each model. For learning the word vectors representation, we use a single setting for the Word2Vec model. The same setting is used for all DL models, which is: use the skip-gram method with window of size 5, and set the dimension of the feature vector to 60.

(b) The second set of the experiments looks at word embedding. In particular, we focus on the different parameters and study how they affect the classification. This experiment was applied on the CNN model only.

TABLE 4. The hyper parameter values used for each of the deep learning (DL) model.

Model	Optimizer	Learning rate	Batch size	No. epoch
CNN	RMSprop	0.001	50	15
CNN-LSTM	Adam [65]	0.001	100	25
CNN-GRU	Adam	0.001	50	25
BiLSTM	Adam	0.001	64	15
BiGRU	Adam	0.001	50	15
Att-LSTM	Adam	0.001	64	10
Att-GRU	Adam	0.001	64	10

All the experiments were performed using 10-fold cross-validation. Cross-validation (CV) is a statistical method used to assess the machine learning models. Generally, in k-fold CV, the dataset is randomly divided into k groups, or folds, of equal size (more or less). One of the folds is used as a validation set, and the other $k-1$ folds are used for training. The process is repeated k times, each time picking a different fold for validation [66, p. 181]. In practice, one performs k-fold CV using $k = 5$ or $k = 10$, as these are shown to yield test error rate estimates that balances between high bias and high variance [66, p. 184].

We have different DL models, and different stemming algorithms. We need to answer the question, does the stemming impacts the classification results of a DL model. Some models performance will be indifferent to the stemming, i.e. whether we use stemming or not, the model's behavior remains the same, and other models will be affected by the stemming. For this, we will use ANOVA (analysis of variance), a statistical method which compares the samples on the basis of their means [67]. ANOVA uses F tests to statistically determine if the means are significantly different from each other (we have eleven groups corresponding to eleven different stemming algorithms classification results). Our null hypothesis will be ''the sample means are equal''. We then use the F statistic when deciding to support or reject the null hypothesis. In ANOVA, if the F value $>$ F critical (for a specific α) then we reject the null hypothesis. For the one-way ANOVA, the F value is given by Eq. (3),

$$F \text{ value} = \frac{\text{between-group variability}}{\text{within-group variability}}. \tag{3}$$

For all the experiments we fixed $\alpha = 0.05$. To calculate the F critical we use an F distribution Table.[3]

We report the performance of document classification using precision, recall, and F-score. These measures are defined using the confusion matrix. A confusion matrix is a table that is used to describe the performance of a binary classification model on a set of test data for which the true values are known. For classification tasks, the terms true

[3]See for example, https://www.stat.purdue.edu/~jtroisi/STAT350Spring2015/tables/FTable.pdf.

positive (TP), true negative (TN), false positive (FP), and false negative (FN) compare the results of the classifier with known judgments. 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. The four outcomes can be formulated in a 2×2 confusion matrix (see Table 5).

TABLE 5. Confusion matrix.

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

The precision (P) measures the exactness of a classifier, while recall (R) measures the completeness of a classifier. We can combine P and R to produce a single metric called F_1 (which has been referred to earlier as the F-score), which is the weighted harmonic mean of both measures. The three measures are given by Eq. (4),

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad R = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad F_1 = \frac{2PR}{P + R}. \tag{4}$$

However, when we have multiple class labels—as in our case—then we need to redefine the measures in Eq. (4). In this case, averaging the measures can give a better view of the general results. For instance, we have the micro-, the macro- and the weighted-averaged measures. The micro-average will aggregate the contributions of all the categories to compute the average metric, whereas the macro-average will compute the metric independently for each category and then take the average (hence treating all categories equally). However, we decided to use the weighted-average for all the measures. The weighted-average considers the class labels imbalance, as in our corpus. It is calculated by computing the metric independently for each category and then taking their average weighted by the number of true instances for each category. This may result in an F-score that is not between precision and recall. In subsequent discussion, weighted-F_1 or just plain F-score will refer to weighted average F-score.

C. EXPERIMENTING WITH DIFFERENT STEMMING ALGORITHMS

We conducted eleven experiments using each of the seven DL models on two different datasets, a total of 154 experiments. For each experiment, we used a different stemming algorithm (we had ten stemmers altogether), and no stemming for the last experiment.

In Section IV-A we mentioned that the number of words in the vocabulary file (dictionary) is less when a root-based stemmer is used as opposed to a light stemmer. Table 6 shows the number of tokens in the vocabulary file for each stemmer when applied to the SPA corpus. Some observations: although [10] is a root-based stemmer, it yields a larger vocabulary than other root-based stemmers. We attribute this to the fact that—by design—no root finding is attempted for non-Arabic words, e.g. الإلكترونية "electronic", which is left untouched. Furthermore, in the case of [18], a light stemmer, the number of tokens exceeds the case when no stemming is applied. This is because the stemmer often produces more than one stem. That is, in some cases the stemmer generates more than one stem for a single word. As we eluded to earlier, more words in the vocabulary means a larger document representation when using word embedding.

TABLE 6. Resultant number of tokens in the vocabulary file for the SPA corpus when using different stemmers.

Algorithm	Stemmer	Number of tokens
Baseline	No stemming	53,860
Root-based	[10]	20,029
	[12]	9,590
	[13]	10,075
	[11]	7,861
	[14]	6,018
Stem-based	[15]	35,350
	[16]	26,659
	[18]	76,308
	[17]	19,825
	[19]	21,102

Tables 7-8 summarizes the performance of classifying the documents in ANT and SPA corpora (respectively). All the values reported in the Tables are average weighted F-score. The performance of the document classifier depends on the stemming algorithm. Whereas the extraction of the wrong root or stem can result in the appearance of a wrong word in a wrong context, which may affect the representation of the word, and consequently the accuracy of the classification. However, among all the deep learning models, the light stemming algorithms generally yield better results compared to those that used root-based stemming algorithms in the context of document classification. In general, the best performance was achieved by BiGRU model for the ANT corpus with stem-based algorithms, while for the SPA corpus the Att-GRU model yield the best result irrespective of the stemming approach. Overall, the best classification result for the SPA corpus was weighted-$F_1 = 97.96\%$ (Att-GRU model with light stemmer [15]), while for ANT corpus it was weighted-$F_1 = 83.63\%$ (BiGRU model with stemmer [16]). Given that SPA has twice as many documents as ANT, this confirms the fact that deep learning algorithms scale well with the amount of data fed.

Performance wise, the weakest models were CNN-LSTM, and CNN-GRU on both copra. This shows that using CNN for feature extraction may not be as effective for Arabic TC. On the other hand, LSTM or GRU models performed well

TABLE 7. Classification results expressed using weighted-F_1 for classifying ANT corpus using various DL models with different stemmers.

		DL model						
Algorithm	Stemmer	CNN	CNN-LSTM	CNN-GRU	BiLSTM	BiGRU	Att-LSTM	Att-GRU
Baseline	No stemming	79.73%	78.67%	78.61%	80.47%	81.50%	80.40%	80.89%
Root-based	[10]	81.13%	77.37%	77.38%	80.58%	82.34%	81.35%	81.36%
	[12]	79.94%	77.19%	77.29%	80.31%	82.45%	81.69%	81.70%
	[13]	81.90%	77.13%	77.92%	79.73%	82.51%	80.61%	80.90%
	[11]	80.38%	76.05%	76.98%	79.01%	81.71%	80.26%	81.08%
	[14]	79.51%	75.94%	76.42%	79.69%	81.71%	80.75%	81.40%
Stem-based	[15]	79.01%	78.28%	78.33%	80.65%	82.88%	81.55%	81.30%
	[16]	81.68%	79.20%	78.47%	81.76%	83.63%	81.90%	82.15%
	[18]	81.31%	70.29%	77.69%	80.25%	83.12%	81.90%	81.61%
	[17]	81.60%	79.17%	80.11%	81.68%	83.23%	81.51%	81.82%
	[19]	80.42%	80.50%	79.55%	81.62%	83.15%	81.89%	83.22%

TABLE 8. Classification results for the SPA corpus.

		DL model						
Algorithm	Stemmer	CNN	CNN-LSTM	CNN-GRU	BiLSTM	BiGRU	Att-LSTM	Att-GRU
Baseline	No stemming	96.71%	94.56%	96.68%	97.15%	97.23%	97.15%	97.82%
Root-based	[10]	95.54%	96.24%	95.96%	97.06%	97.04%	97.04%	97.39%
	[12]	94.61%	95.50%	95.78%	96.91%	96.78%	96.29%	96.89%
	[13]	95.92%	96.79%	96.40%	97.44%	97.30%	96.71%	97.74%
	[11]	95.06%	93.11%	96.27%	97.03%	97.25%	96.06%	97.49%
	[14]	96.14%	97.01%	96.63%	97.28%	97.21%	97.20%	97.76%
Stem-based	[15]	96.59%	96.67%	96.78%	97.33%	97.35%	97.38%	97.96%
	[16]	96.42%	96.82%	96.57%	97.07%	97.12%	96.95%	97.48%
	[18]	96.49%	75.42%	52.26%	97.09%	97.37%	96.90%	97.65%
	[17]	95.91%	96.81%	96.38%	97.27%	97.21%	97.30%	97.73%
	[19]	96.58%	95.95%	96.29%	96.98%	97.16%	97.16%	97.51%

either with bidirectional or with attention mechanism. What surprised us was the degraded performance by CNN-LSTM and CNN-GRU on the SPA corpus when used with the light stemmer [18]. For CNN-LSTM, the performance dropped from mid 90's (using any of the other stemming algorithms) to 75.4%, and for CNN-GRU it is even worse where it drops to $\approx 52\%$. We attribute this to the stemming algorithm in [18] which often produced more than one stem for a word,[4] and this affected the word embedding learning and consequently the CNN feature extraction process. The same stemming algorithm performed well with other models, e.g. BiGRU and Att-GRU. This proves that learning the full context from both directions or using the attention mechanism to concentrate on the useful information efficiently does improve the result.

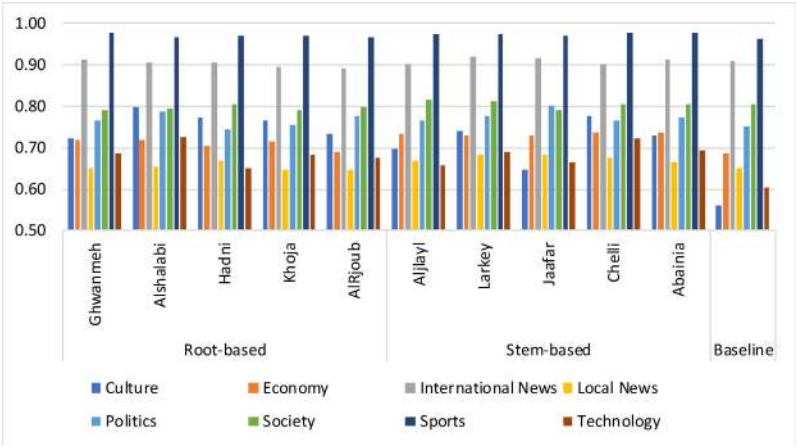
We now answer one of the questions raised in the paper. Does stemming impact the performance of Arabic TC when using a deep learning model? We will use ANOVA test on the 10-fold F-score results for all eleven different stemming algorithms. Table 9 lists the F value and F critical of the ANOVA test, recalling that we set $\alpha = 0.05$ (Section V-B). For the ANT corpus, the F value is greater than F critical for the four models: CNN-LSTM, CNN-GRU, BiLSTM, and BiGRU. We therefore reject the null hypothesis and claim that at least one of the stemming algorithms significantly affected the classification result. However, for the other three models (e.g., CNN), the stemming algorithms where of no use and its impact on the classification was insignificant. For the SPA corpus, there is a clear evidence that the stemming algorithms had an impact on the classification. This is true for all the models except for BiLSTM and BiGRU.

There is an interesting observation here. From Tables 7-8, we note that the range of F-score is narrow indeed, and this is true for both corpora and for all but two models (i.e. CNN-LSTM and CNN-GRU). In the case of the ANT corpus, the F-score ranges between 79.01% and 83.63%, while for the SPA corpus it ranges between 94.61% and 97.96%. What is more interesting is that the performance of no-stemming is somewhere in between. This goes counter to the general belief that stemming is a necessary step in any Arabic NLP application. We believe that when using deep learning algorithms and word embedding, the impact of stemming is minuscule, and we may still get a great performance without it. For instance, we get F-score $= 97.82\%$ using Att-GRU without resorting to any stemming. However, if we are concerned about the training time then it is better to do stemming as it reduces the size of the vocabulary. Note that the fastest training models were CNN-LSTM and CNN-GRU,

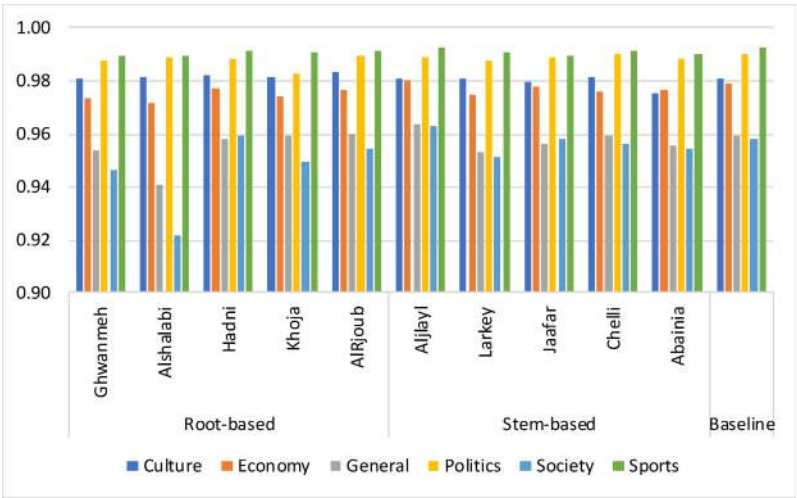
[4]There is a version that produces a single stem per word, but we were unsuccessful in contacting the authors.

TABLE 9. ANOVA test ($\alpha = 0.05$) for both corpora for the 10-fold result among all ten stemming algorithms.

	CNN		CNN-LSTM		CNN-GRU		BiLSTM		BiGRU		Att-LSTM		Att-GRU	
Dataset	F value	F critical	F value	F critical	F value	F critical	F value	F critical	F value	F critical	F value	F critical	F value	F critical
ANT	0.96	1.93	7.37	1.93	4.09	1.93	2.43	1.93	2.56	1.93	1.83	1.93	1.80	1.93
SPA	5.03	1.93	3.79	1.93	13.09	1.93	1.40	2.00	1.44	1.93	2.43	1.93	5.22	1.93



(a) ANT Corpus



(b) SPA Corpus

FIGURE 9. The F_1 score of classifying documents of each label using different stemmers on the two corpora. Baseline stands for "no stemming".

although they resulted in the worst performance among all the models. Whereas, the models BiLSTM and BiGRU had the largest training time. The CNN model had a reasonable training time and with a fairly well performance, followed by Att-GRU and Att-LSTM.

Tables 10-11 shows the classification results for each label for the ANT and SPA corpora (respectively), using the best performing model for each dataset. For each category, the results are reported in terms of F_1 score (see Eq. (4)). For the ANT corpus (Table 7) for example, using the stemmer [13] for the category "culture" the $F_1 = 77\%$, while for the category "international news" the score is $F_1 = 91\%$. The number of documents in category "culture" is 124 (see Table 3). The weighted-F_1 score for this stemmer is 82.5%.

Figure 9 plots the results presented in Tables 10 and-11, ordering them into the category (label). Within each label, we list the F_1 score of the document classification into that particular label using different stemmers: no stemming (baseline), [10], [12] etc. For the SPA corpus (Figure 9b), the results are very much consistent, and the classifier has done an equally good job in classifying documents into different categories. What is more interesting is that the use of

TABLE 10. Classification results of individual categories for the ANT corpus on BiGRU model. Values under each category are expressed using the F_1 score.

		Category										
Algorithm	Stemmer	Cul	Econ	Int	Loc	Pol	Soc	Spor	Tech	P	R	weighted-F_1
Baseline	No stemming	0.56	0.69	0.91	0.65	0.75	0.81	0.96	0.61	0.815	0.817	0.815
Root-based	[10]	0.72	0.72	0.91	0.65	0.77	0.79	0.98	0.69	0.824	0.824	0.823
	[12]	0.80	0.72	0.91	0.65	0.79	0.79	0.97	0.73	0.824	0.826	0.824
	[13]	0.77	0.70	0.91	0.67	0.75	0.81	0.97	0.65	0.825	0.826	0.825
	[11]	0.77	0.71	0.90	0.65	0.76	0.79	0.97	0.68	0.817	0.818	0.817
	[14]	0.73	0.69	0.89	0.65	0.78	0.80	0.97	0.68	0.816	0.818	0.817
Stem-based	[15]	0.70	0.74	0.90	0.67	0.77	0.82	0.97	0.66	0.828	0.831	0.829
	[16]	0.74	0.73	0.92	0.68	0.78	0.81	0.97	0.69	0.836	0.838	0.836
	[18]	0.65	0.73	0.92	0.68	0.80	0.79	0.97	0.67	0.831	0.832	0.831
	[17]	0.78	0.74	0.90	0.68	0.77	0.81	0.98	0.73	0.832	0.834	0.832
	[19]	0.73	0.74	0.91	0.67	0.77	0.80	0.98	0.69	0.831	0.833	0.832

TABLE 11. The classification results of individual categories for the SPA corpus using Att-GRU model.

		Category								
Algorithm	Stemmer	Cul	Econ	Gen	Pol	Soc	Spor	P	R	weighted-F_1
Baseline	No stemming	0.98	0.98	0.96	0.99	0.96	0.99	0.978	0.978	0.978
Root-based	[10]	0.98	0.97	0.95	0.99	0.95	0.99	0.974	0.974	0.974
	[12]	0.98	0.97	0.94	0.99	0.92	0.99	0.969	0.969	0.969
	[13]	0.98	0.98	0.96	0.99	0.96	0.99	0.977	0.977	0.977
	[11]	0.98	0.97	0.96	0.99	0.95	0.99	0.975	0.975	0.975
	[14]	0.98	0.98	0.96	0.99	0.95	0.99	0.978	0.978	0.978
Stem-based	[15]	0.98	0.98	0.96	0.99	0.96	0.99	0.980	0.980	0.980
	[16]	0.98	0.97	0.95	0.99	0.95	0.99	0.975	0.975	0.975
	[18]	0.98	0.98	0.96	0.99	0.96	0.99	0.977	0.977	0.977
	[17]	0.98	0.98	0.96	0.99	0.96	0.99	0.977	0.977	0.977
	[19]	0.98	0.98	0.96	0.99	0.95	0.99	0.975	0.975	0.975

TABLE 12. Comparison of our system's performance vs [24] (uses deep belief network). Both systems use a different root-based stemmer. Each system uses a different dataset. The performance measure of the other system is as reported by the respective author.

System	Technique	Dataset	P	R	F_1
[24]	DBN + Markov clustering	12,000 documents from Al-Jazeera news website, SPA and Saudi newspapers most likely from 2008	91.2%	90.9%	91.02%
Our system	Att-GRU + word embedding	13,402 documents from SPA from 2018	97.79%	97.78%	97.76%

the stemmer has little impact on the performance. However, for the ANT corpus (Figure 9a), there is a difference in performance between labels. We note the best performance has been for the ''sports'' and ''international news'' categories, while the worst performance was for the ''local news'' and ''technology'' categories. Even within a category, e.g. ''culture'', the performance varies between stemmers which ranges between 56% (no stemming) and 80% (e.g., [12]).

Finally, we wanted to compare the performance with Jindal [24], another deep-learning based classifier for Arabic documents. Table 12 summarizes the performance of both systems. For stemming, the author used a root-based stemmer, but did not provide any further detail. So, to be fair, we report the performance of our system using the root-based stemmer [14], though it is not our best reported performance. Clearly our system tops the performance of the other system by approximately 7%.

D. EXPERIMENTING WITH WORD EMBEDDING PARAMETERS

This is the second set of experiments where we looked into appropriate parameters for the Word2Vec word embedding algorithm. We confined this experiment to CNN model only. The parameters that can be controlled are: dimension of the vector, the learning method (skip-gram or bag-of-words), size of the window (number of neighboring words), and the cut-off occurrences for words in the vocabulary (less frequently occurring words are ignored). We conducted two experiments to identify which word embedding parameters affect the classification process. In the first experiment, we explored the two learning methods, skip-gram and bag-of-words (CBOW) for Word2Vec, with different vector dimensions. In the second experiment, we explored the size of the window. For both experiments, we picked two stemming algorithms, one root-based [10] and the other is stem-based [16], both yield

TABLE 13. Effect of vector dimensions on the *F*-score of document classification using the two learning methods for Word2Vec.

		Skip-gram				Bag-of-words (CBOW)			
Algorithm	Stemmer	50	60	100	300	50	60	100	300
Root-based	[10]	0.959	0.965	0.967	0.965	0.962	0.956	0.967	0.963
Stem-based	[16]	0.968	0.968	0.967	0.964	0.962	0.967	0.968	0.968

TABLE 14. Effect of window sizes on the *F*-score of document classification.

		Skip-gram			CBOW		
Algorithm	Stemmer	2	5	8	2	5	8
Root-based	[10]	0.965	0.967	0.961	0.964	0.967	0.962
Stem-based	[16]	0.968	0.968	0.964	0.967	0.967	0.967

good performance with CNN model. The experiments were performed using 10-fold cross-validation on the SPA corpus. All the reported performances are measured in terms of weighted-F_1.

The objective of the first experiment was to look at which of the two learning methods is more appropriate while—at the same time—trying different dimensions for the vector. Table 13 shows the results of the first experiment when using vectors of dimension 50, 60, 100, and 300 expressed in terms of weighted-F_1 score. For this experiment, we fixed the other two parameters as follows. Fixed the size of windows to 5 (a typical value) and set the cut-off frequency to one (no word was left out). From Table 13, we can observe that both methods (skip-gram and CBOW) yielded a comparable and good performance for Arabic document classification. If we had used the stem-based approach in the pre-processing step, the skip-gram method would have worked well with a smaller vector dimension. While CBOW requires a larger sized vector to achieve a high success rate. In case of the root-based method, we achieved the best values when the dimension of the vector was 100, and this was true for both methods. Although the small vector dimension in the skip-gram method works well with stem-based, it leads to ambiguity when using the root-based stemmer for both methods.

In the second experiment, we looked at the effect of window size on the classification process. Table 14 shows the effect of different window sizes in each method on the classification process, where we fixed the dimension of the vector to 60. We tested for windows of sizes 2, 5, and 8. As we mentioned earlier, windows of size 5 are typical. We note that using larger sized windows did not improve the classification. For stem-based, a smaller sized window is a good choice, but not so for the root-based stemmer. Evidently, the best choice is a size 5 window. It works well with both methods using either stemming approach.

VI. CONCLUSION

In this work, we studied the effect of stemming strategies and word embedding on the performance of Arabic document classification using various deep learning models. The following models were explored: CNN, CNN-LSTM, CNN-GRU, Bidirectional LSTM, Bidirectional GRU, Attention-based LSTM, and Attention-based GRU. And for the stemming strategies we investigated three of them: no stemming, root-based (five different algorithms), and stem-based (five different algorithms). While for word embedding, we used Word2Vec. For testing, we used two large datasets, one with six predefined categories and the other with eight. For testing we used 10-fold cross-validation.

For the first group of experiments, we conducted a total of 154 experiments (seven models $\times$ eleven stemming algorithms $\times$ two corpora). For these experiments we fixed the word embedding parameters. In the second group of experiments, we use the CNN model and picked two stemming algorithms that preformed well with CNN model (one root-based and the other stem-based), we then looked into the effect of different parameter settings controlling the Word2Vec used to classifying the documents. Summarizing the challenges and lessons learned (not in any particular order):

- There is a lack of research into Arabic NLP, and the absence of cooperation among researches in the field aggravates the problem. In this work, we had to implement many of the stemming algorithms ourself following the description in their respective papers.
- Deep learning models have a steep learning curve. And the size of the vocabulary affects the learning time, proportionally. We performed all the experiments in this work using 10-fold cross-validation. This contributed extra time to finish the experiments. For instance, the BiGRU model with stemmer [18] took over 75 hours on a system with Nvidia Tesla K80 GPU having 12GB memory.
- Label or class imbalance in the training set is a major issue in text classification task. This was the case of the publicly available ANT corpus. However, we kept this in mind when compiling our data set from the official SPA (Saudi Press Agency), where we tried to retain proportionate classes.
- Experimental results show that stem-based (or light stemmers) algorithms generally yield a slightly better performance compared to the root-based (or heavy stemmers).
- Looking at the small differences in the performance between the three stemming strategies (no stemming, root-based, or light stemming), we can safely claim that the stemming step is optional for Arabic text

classification task, as not much was gained from using stemming phase. One thing to keep in mind, stemming helps reduce the training time as the system has to learn less vocabulary.

- We believe in the fact that simple model is more appropriate for a rich morphological language such as Arabic. During the construction of different deep learning models, we found that adding more layers did not improve the results. This was specially true for the models based on LSTM and GRU.
- Using bidirectional learning or attention mechanism with LSTM and GRU significantly improves the classification result, whereas using CNN model as feature extraction layer before LSTM and GRU is ineffective. Nevertheless, using the standalone CNN model can achieve satisfactory classification results.
- For the embedding layer (Word2Vec), we found that it was better to use skip-gram with the stem-based algorithm, as we can get good results using a vector of dimension $50 \sim 60$. To achieve a comparable performance using CBOW (bag-of-words) and the root-based algorithm, we needed a vector of dimension 100. In addition, for Word2Vec, the best window size is 5.

For future work, we intend to experiment with Arabic multi-label classification problem using the proposed models in this work. We also intend to look into Graph Convolutional Networks (GCN), yet another successful deep learning architecture for NLP tasks, for Arabic document classification problem.

REFERENCES

[1] R. Duwairi, M. Al-Refai, and N. Khasawneh, "Stemming versus light stemming as feature selection techniques for arabic text categorization," in *Proc. Innov. Inf. Technol. (IIT)*, Nov. 2007, pp. 446–450.

[2] A. M. El-Halees, "Arabic text classification using maximum entropy," *Islamic Univ. J.*, vol. 15, no. 1, pp. 1–11, Dec. 2007.

[3] F. Harrag and E. El-Qawasmah, "Neural network for arabic text classification," in *Proc. 2nd Int. Conf. Appl. Digit. Inf. Web Technol.*, Aug. 2009, pp. 778–783.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781

[5] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Aug. 2014, pp. 1746–1751.

[6] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and Word2vec for text classification with semantic features," in *Proc. IEEE 14th Int. Conf. Cognit. Informat. Cognit. Comput. (ICCI*CC)*, Jul. 2015, pp. 136–140.

[7] M. Hughes, I. Li, S. Kotoulas, and T. Suzumura, "Medical text classification using convolutional neural networks," *Stud Health Technol. Inf.*, vol. 235, pp. 246–250, May 2017.

[8] Z.-T. Yang and J. Zheng, "Research on chinese text classification based on Word2vec," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 1166–1170.

[9] D. Zhang, H. Xu, Z. Su, and Y. Xu, "Chinese comments sentiment classification based on word2vec and SVMperf," *Expert Syst. Appl.*, vol. 42, no. 4, pp. 1857–1863, Mar. 2015.

[10] S. Ghwanmeh, G. Kanaan, R. Al-Shalabi, and S. Rabab'ah, "Enhanced algorithm for extracting the root of arabic words," in *Proc. 6th Int. Conf. Comput. Graph., Imag. Visualizat.*, Aug. 2009, pp. 388–391.

[11] S. Khoja and R. Garside, "Stemming arabic text," Dept. Comput., Lancaster Univ., Lancaster, U.K., Tech. Rep., 1999.

[12] R. Alshalabi, "Pattern-based stemmer for finding arabic roots," *Inf. Technol. J.*, vol. 4, no. 1, pp. 38–43, Jan. 2005.

[13] M. Hadni, A. Lachkar, and S. A. Ouatik, "A new and efficient stemming technique for arabic text categorization," in *Proc. Int. Conf. Multimedia Comput. Syst.*, May 2012, pp. 791–796.

[14] A. Al-Rjoub, "A new approach for Arabic root extraction," M.S. thesis, Dept. MSc Comput. Sci., Jordan Univ. Sci. Technol., Irbid, Jordan, 2007.

[15] M. Aljlayl and O. Frieder, "On arabic search: Improving the retrieval effectiveness via a light stemming approach," in *Proc. 11th Int. Conf. Inf. Knowl. Manage. CIKM*, 2002, pp. 340–347.

[16] L. S. Larkey, L. Ballesteros, and M. E. Connell, "Light stemming for Arabic information retrieval," in *Arabic Computational Morphology*. Springer, 2007, pp. 221–243.

[17] A. Chelli. (2016). *Arabic Stemmer*. [Online]. Available: https://www.arabicstemmer.com/

[18] Y. Jaafar, D. Namly, K. Bouzoubaa, and A. Yousfi, "Enhancing arabic stemming process using resources and benchmarking tools," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 29, no. 2, pp. 164–170, Apr. 2017.

[19] K. Abainia, S. Ouamour, and H. Sayoud, "A novel robust arabic light stemmer," *J. Experim. Theor. Artif. Intell.*, vol. 29, no. 3, pp. 557–573, May 2017.

[20] D. Laham, "Latent semantic analysis approaches to categorization," in *Proc. 19th Annu. Conf. Cognit. Sci. Soc.*, 1997, p. 979.

[21] M. Davenport, "Introduction to modern information retrieval," *J. Med. Library Assoc. (JMLA)*, vol. 100, no. 1, p. 75, 2012.

[22] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," in *Proc. 3rd Annu. Symp. Document Anal. Inf. Retr. (SDAIR-94)*, 1994, pp. 161–175.

[23] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.

[24] V. Jindal, "A personalized Markov clustering and deep learning approach for arabic text categorization," in *Proc. ACL Student Res. Workshop*, 2016, pp. 145–151.

[25] F. S. Al-Anzi and D. AbuZeina, "Beyond vector space model for hierarchical arabic text classification: A Markov chain approach," *Inf. Process. Manage.*, vol. 54, no. 1, pp. 105–115, Jan. 2018.

[26] A. M. Azmi and E. A. Aljafari, "Universal Web accessibility and the challenge to integrate informal arabic users: A case study," *Universal Access Inf. Soc.*, vol. 17, no. 1, pp. 131–145, Mar. 2018.

[27] D. Harman, "How effective is suffixing?" *J. Amer. Soc. Inf. Sci.*, vol. 42, no. 1, pp. 7–15, Jan. 1991.

[28] Y. Kadri and J.-Y. Nie, "Effective stemming for Arabic information retrieval," in *Proc. Challenge Arabic NLP/MT Conf.*, Londres, U.K., Oct. 2006, pp. 68–74.

[29] I. A. Al-Kharashi and M. W. Evens, "Comparing words, stems, and roots as index terms in an arabic information retrieval system," *J. Amer. Soc. Inf. Sci.*, vol. 45, no. 8, pp. 548–560, Sep. 1994.

[30] K. Taghva, R. Elkhoury, and J. Coombs, "Arabic stemming without a root dictionary," in *Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC)*, vol. 2, Apr. 2005, pp. 152–157.

[31] A. Nehar, D. Ziadi, H. Cherroun, and Y. Guellouma, "An efficient stemming for arabic text classification," in *Proc. Int. Conf. Innov. Inf. Technol. (IIT)*, Mar. 2012, pp. 328–332.

[32] H. M. Al-Serhan, R. Al Shalabi, and G. Kannan, "New approach for extracting arabic roots," in *Proc. Arab Conf. Inf. Technol. (ACIT)*, 2003, pp. 42–59.

[33] H. M. Harmanani, W. T. Keirouz, and S. Raheel, "A rule-based extensible stemmer for information retrieval with application to Arabic," *The Int. Arab J. Inf. Technol.*, vol. 3, no. 3, pp. 265–272, 2006.

[34] M. Momani and J. Faraj, "A novel algorithm to extract tri-literal arabic roots," in *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl.*, May 2007, pp. 309–315.

[35] Q. Yaseen and I. Hmeidi, "Extracting the roots of arabic words without removing affixes," *J. Inf. Sci.*, vol. 40, no. 3, pp. 376–385, Jun. 2014.

[36] M. N. Al-Kabi, S. A. Kazakzeh, B. M. Abu Ata, S. A. Al-Rababah, and I. M. Alsmadi, "A novel root based arabic stemmer," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 27, no. 2, pp. 94–103, Apr. 2015.

[37] L. S. Larkey, L. Ballesteros, and M. E. Connell, "Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR '02)*, 2002, pp. 275–282.

[38] A. Chen and F. Gey, "Building an arabic stemmer for information retrieval," in *Proc. 11th Text Retr. Conf. (TREC)*, 2002, pp. 631–639.

[39] A. Goweder and A. D. Roeck, "Assessment of a significant Arabic corpus," in *Proc. Arabic NLP Workshop ACL/EACL*, 2001, pp. 73–79.
[40] H. Froud, "A comparative study of root -based and stem -based approaches for measuring the similarity between arabic words for arabic text mining applications," *Adv. Comput., Int. J.*, vol. 3, no. 6, pp. 55–67, Nov. 2012.
[41] Y. A. Alhaj, J. Xiang, D. Zhao, M. A. A. Al-Qaness, M. Abd Elaziz, and A. Dahou, "A study of the effects of stemming strategies on arabic document classification," *IEEE Access*, vol. 7, pp. 32664–32671, 2019.
[42] T. Zerrouki. (2017). *Tashaphyne Arabic Light Stemmer*. Accessed: Aug. 25, 2019. https://pypi.org/project/Tashaphyne/
[43] M. K. Saad and W. M. Ashour, "OSAC: Open source arabic corpora," in *Proc. 6th Int. Symp. Elect. Electron. Eng. Comput. Sci. (EEECS)*, 2010, pp. 1–6.
[44] F.-Z. El-Alami and S. O. El Alaoui, "An efficient method based on deep learning approach for Arabic text categorization," in *Proc. Int. Arab Conf. Inf. Technol. (ACIT)*, 2016, pp. 1–7.
[45] M. Sawalha and E. Atwell, "Comparative evaluation of Arabic language morphological analysers and stemmers," in *Proc. Int. Conf. Comput. Linguistics (COLING)*, 2008, pp. 107–110.
[46] F.-Z. El-Alami and S. O. El Alaoui, "Word sense representation based-method for arabic text categorization," in *Proc. 9th Int. Symp. Signal, Image, Video Commun. (ISIVC)*, Nov. 2018, pp. 141–146.
[47] S. Bahassine, A. Madani, M. Al-Sarem, and M. Kissi, "Feature selection using an improved chi-square for arabic text classification," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 2, pp. 225–231, Feb. 2020.
[48] T. Kanan, O. Sadaqa, A. Almhirat, and E. Kanan, "Arabic light stemming: A comparative study between P-stemmer, khoja stemmer, and light10 stemmer," in *Proc. 6th Int. Conf. Social Netw. Anal., Manage. Secur. (SNAMS)*, Oct. 2019, pp. 511–515.
[49] T. Kanan, R. Kanaan, O. Al-Dabbas, G. Kanaan, A. Al-Dahoud, and E. Fox, "Extracting named entities using named entity recognizer for Arabic news articles," *Int. J. Adv. Stud. Comput., Sci. Eng.*, vol. 5, no. 11, pp. 78–84, 2016.
[50] S. Boukil, M. Biniz, F. E. Adnani, L. Cherrat, and A. E. E. Moutaouakkil, "Arabic text classification using deep learning technics," *Int. J. Grid Distrib. Comput.*, vol. 11, no. 9, pp. 103–114, Sep. 2018.
[51] S. Boukil, F. El Adnani, A. E. El Moutaouakkil, L. Cherrat, and M. Ezziyyani, "Arabic stemming techniques as feature extraction applied in arabic text classification," in *Proc. Int. Conf. Adv. Inf. Technol., Services Syst.* Springer, 2017, pp. 349–361.
[52] M. Elbes, A. Aldajah, and O. Sadaqa, "P-stemmer or NLTK stemmer for arabic text classification?" in *Proc. 6th Int. Conf. Social Netw. Anal., Manage. Secur. (SNAMS)*, Oct. 2019, pp. 516–520.
[53] S. Bird, E. Klein, and E. Loper, *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit*. Newton, MA, USA: O'Reilly Media, 2009.
[54] H. ALSaif and T. Alotaibi, "Arabic text classification using feature-reduction techniques for detecting violence on social media," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 4, pp. 77–87, 2019.
[55] A. Oussous, A. A. Lahcen, and S. Belfkih, "Impact of text pre-processing and ensemble learning on arabic sentiment analysis," in *Proc. 2nd Int. Conf. Netw., Inf. Syst. Secur. NISS*, 2019, pp. 1–9.
[56] M. K. Saad and W. M. Ashour, "Arabic morphological tools for text mining," in *Proc. 6th Int. Conf. Electr. Comput. Syst. (EECS)*, 2010.
[57] A. Al-Thubaity, A. Alqarni, and A. Alnafessah, "Do words with certain part of speech tags improve the performance of arabic text classification?" in *Proc. 2nd Int. Conf. Inf. Syst. Data Mining ICISDM*, 2018, pp. 155–161.
[58] E. Al-Shawakfa, A. Al-Badarneh, S. Shatnawi, K. Al-Rabab'ah, and B. Bani-Ismail, "A comparison study of some arabic root finding algorithms," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 61, no. 5, pp. 1015–1024, May 2010.
[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[60] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[61] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.
[62] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.
[63] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.
[64] A. Chouigui, O. B. Khiroun, and B. Elayeb, "ANT corpus: An arabic news text collection for textual classification," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017, pp. 135–142.
[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[66] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning With Applications in R*, vol. 112. New York, NY, USA: Springer-Verlag, 2013.
[67] R. A. Fisher, "XV.—The correlation between relatives on the supposition of Mendelian inheritance," *Trans. Roy. Soc. Edinburgh*, vol. 52, no. 2, pp. 399–433, 1919.

HUDA ABDULRAHMAN ALMUZAINI received the B.Sc. degree in computer science from Qassim University, and the M.Sc. degree in computer science from King Saud University, Riyadh, Saudi Arabia, where she is currently pursuing the Ph.D. degree. She is also a Lecturer with the Department of Computer Science, Imam Mohammad ibn Saud Islamic University, Riyadh. Her research interests include natural language processing and deep learning.

AQIL M. AZMI received the B.S.E. degree in electrical & computer engineering (ECE) from the University of Michigan, Ann Arbor, MI, USA, and the M.Sc. degree in electrical engineering and the Ph.D. degree in computer science from the University of Colorado, Boulder, CO, USA. He is currently Professor with the Department of Computer Science, King Saud University, Saudi Arabia. His current research interests include natural language processing, computational biology, bioinformatics, image understanding and processing, and digital humanities specifically critical analysis of historical and religious texts.