Received December 27, 2018, accepted January 7, 2019, date of publication January 21, 2019, date of current version February 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2893460

Arabic Word Segmentation With Long Short-Term Memory Neural Networks and Word Embedding

ABDULRAHMAN ALMUHAREB, WALEED ALSANIE, AND ABDULMOHSEN AL-THUBAITY
National Center for Artificial Intelligence and Big Data Technology, King Abdulaziz City for Science and Technology, Riyadh 11442, Saudi Arabia

Corresponding author: Abdulrahman Almuhareb (muhareb@kacst.edu.sa)

This work was supported by King Abdulaziz City for Science and Technology.

ABSTRACT In this paper, we propose an Arabic word segmentation technique based on a bi-directional long short-term memory deep neural network. This paper addresses the two tasks of word segmentation only and word segmentation for nine cases of the rewrite. Word segmentation with a rewrite concerns inferring letters that are dropped or changed when the main word unit is attached to another unit, and it writes these letters back when the two units are separated as a result of segmentation. We only use binary labels as indicators of segmentation positions. Therefore, label 1 is an indicator of the start of a new word (split) in a sequence of symbols not including whitespace, and label 0 is an indicator for any other case (no-split). This is different from the mainstream feature representation for word segmentation in which multi-valued labeling is used to mark the sequence symbols: beginning, inside, and outside. We used the Arabic Treebank data and its clitics segmentation scheme in our experiments. The trained model without the help of any additional language resources, such as dictionaries, morphological analyzers, or rules, achieved a high F1 value for the Arabic word segmentation only (98.03%) and Arabic word segmentation with the rewrite (more than 99% for frequent rewrite cases). We also compared our model with four state-of-the-art Arabic word segmenters. It performed better than the other segmenters on a modern standard Arabic text, and it was the best among the segmenters that do not use any additional language resources in another test using classical Arabic text.

INDEX TERMS Arabic word segmentation, bi-directional long short-term memory, deep learning, neural network, word embedding.

I. INTRODUCTION

Word segmentation is a traditional problem in natural language processing. It focuses on segmenting certain morphemes, that is, affixes and clitics, from the beginning and end of words and stems. Arabic word segmentation is essential for various natural language processing and text mining tasks, such as machine translation [1], text classification [2], and parsing [3]. For instance, two words that are combined in a single sequence need to be identified before the sentence that includes them can be conveniently parsed. If a preposition is attached to a noun, then these two morphemes need to be split to match the grammar rule PP $\rightarrow$ IN NOUN. Without the two morphemes being split, this rule will not be applied.

Arabic morphology is known for its rich and complex inflection rules. Examples of such rules include the attachment of a preposition, for example, ب "b" or ل "l," to words; the attachment of object pronouns to the preceding words, for example, ه "h" in أعطيته ">ETyth"[1] (I gave him) is attached to أعطيت ">ETyt" (I gave); and the coordinating conjunction و "w," which is conventionally attached to succeeding words in formal Arabic orthography.

In Arabic phonology, there are phenomena in which some symbols in words are dropped when these words are attached to other words to make pronunciation easier. This is in turn reflected in the orthography. For instance, the first letter ا "A" of the determiner ال "Al" is dropped when it is attached after the preposition ل "l." An example of this phenomenon is the sequence للبيت "llbyt" (for the house), which is a combination of the preposition ل "l" and determined noun البيت "Albyt" (the house). When these words are split as a result of segmentation, the resulting segments are ل "l" and لبيت "lbyt" (for a house). However, the second segment is incorrect because it must be البيت "Albyt" (the house) and not لبيت "lbyt" (for a house), but the first symbol ا "A" was dropped when the two morphemes combined to make pronunciation smoother. This symbol needs to be added back after the segmentation to result in the correct word البيت "Albyt" (the house). This problem is referred to in the literature as *segmentation with rewrite*.

[1]Buckwalter Arabic transliteration

The proposed systems to solve the problem of Arabic word segmentation typically consider the problem either within part of speech (POS) tagging and morphological disambiguation or as a standalone problem (see Section II). Segmentation with rewrite is either ignored completely [4], considered for a limited number of rewrite [5], considered with the help of a morphological analyzer [6], considered with the help of dictionaries [7], or considered with the help of dictionaries and rules.

In this paper, we propose a deep learning approach based on a recurrent neural network (RNN), specifically bi-directional long short-term memory (Bi-LSTM), to solve the problem of Arabic word segmentation without rewriting and with rewriting. The contribution of this paper is twofold. First, we use a simple representation scheme and show its effectiveness for Arabic word segmentation. The common practice for word segmentation is to use the IOB labeling scheme to mark the symbols in the sequence. Label 'I' denotes 'inside,' which indicates the continuation of the sequence. Label 'O' denotes 'outside,' which indicates that the symbol is outside the sequence. Label 'B' denotes 'beginning,' which indicates the beginning of the sequence. Instead of using this scheme, we use a binary labeling scheme. In this scheme, '1' indicates a splitting position, which defines the beginning of a new morpheme that is attached to another morpheme in the original corpus, and '0' is used otherwise. To the best of our knowledge, this work is the first to use a binary labeling scheme and deep neural network (DNN) for this problem. Second, we consider a wider range of cases of segmentation rewriting (nine cases) than have been previously reported for Arabic word segmentation based on the output of the deep learning model without the use of any type of morphological analyzer, dictionary, or rules. To train our model, we used Arabic Treebank (ATB) data and an ATB clitics segmentation schema. The resulting model achieved high *F1* values for both segmentation without rewriting and for most cases of segmentation with rewriting. The model also achieved a competitive *F1* value when compared with state-of-the-art Arabic word segmentation systems.

This paper is organized as follows: In Section II, we present the related literature. In Section III, we provide some background information about linguistic and neural networks. In Sections IV and V, we explain the proposed method and the dataset used in the experiments. In Sections VII and VIII, we illustrate the experimental results.

II. RELATED WORK

This work is related to two research topics: Arabic word segmentation and using deep learning for word segmentation. Arabic word segmentation is typically considered as a part of the POS tagging problem. Two approaches have been conducted in the field of Arabic POS tagging research to manage Arabic word segmentation: rule-based and statistical. For the rule-based approach, different methods have been used. For instance, Khoja [8], Zribi *et al.* [9], Alqrainy *et al.* [10], Al-Taani and Al-Rub [11], and Hadni *et al.* [12] used the lexicon approach and a predefined set of morphological rules to identify affixes and clitics. Other rule-based methods have been used, such as transformation-based learning [13], regular expressions [14], and the first solution provided by the morphological analyzer [15].

For the statistical approach, two methods have been used for Arabic word segmentation. The first method separates the segmentation process from the POS tagging process. In this method, segmentation is considered as a classification problem and machine learning (ML) methods are applied to train the chosen classifier. Different ML algorithms have been applied, such as SVM ([16]–[18]), k-nearest neighbor algorithm ([19], [20]), conditional random fields (CRF) [21], and maximum likelihood model [22]. For the second method, the process of choosing the correct segmentation is conducted within the process of choosing the correct morphology analysis, which contains the segmentation information and POS tagging decision. For instance, Habash and Rambow [23] and Roth *et al.* [24] used the SVM algorithm to choose the best solution of BAMA, Zalmout and Habash [6] used Bi-LSTM, and Freihat *et al.* [25] used a maximum entropy POS tagger. All the aforementioned studies have either relied on the Buckwalter morphological analyzer or dictionaries with rules to solve the issue of segmentation with rewriting.

Some studies have focused only on Arabic word segmentation, such as those undertaken by Lee *et al.* [26], and Benajiba and Zitouni [4]. In both studies, they used the same approach, in which they built a seed segmenter based on a language model and tables for affixes, clitics, and stems. Both of the seed segmenters were trained on small manually segmented corpora (10K and 572K). To improve the accuracy of both segmenters, the seed segmenters were then used to segment a large corpus containing 155Mwords to enhance the stems table in the seed segmenters by acquiring new stems. Neither of these two studies considered the problem of rewriting. Lee *et al.* [26] reported 97% segmentation accuracy for both affixes and clitics on a test corpus comprising 28,449 words extracted from the LDC ATB: Part 1 v 2.0. Benajiba and Zitouni [4] reported the accuracy for two experiments. In the first experiment, they achieved 98.1% accuracy for affixes and clitics segmentation, and for the second experiment, they achieved 99.4% for clitics segmentation. Both experiments used a 42,591 word corpus extracted from Parts 1–3 of the ATB.

Monroe *et al.* [5] developed an Arabic word segmenter for both modern standard Arabic (MSA) and the Egyptian dialect using the CRF algorithm. For MSA, they used Parts 1–3 of the ATB and the Broadcast News ATB (BN) to train their segmenter to segment all clitics as provided by the ATB. They used five values for segmentation representation and considered only three cases of segmentation rewriting. The reported F-measure for MSA was 98.30% using ATB and 97.17% using BN.

The most recent work that has focused on Arabic word segmentation only is Farasa [7]. Farasa considers the segmentation of clitics as provided by ATB in addition to some affixes, such as the definite article ال "Al" and taa marbuttah ة "p," among others. Farasa was built based on SVM-rank using linear kernels and the following features: likelihood of stems, prefixes, suffixes, and their combination; presence in lexicons containing valid stems and named entities; and underlying stem templates. Farasa was trained on different parts of ATB and tested on a corpus comprising 70 WikiNews articles containing 18,271 words. The error rate was 1.06% (accuracy equals 98.94%). Farasa uses lexicons to solve the problem of segmentation with rewriting.

Several researchers have used DNNs to manage word segmentation. RNNs with different configurations of hidden units and window sizes have been used for standalone Chinese word segmentation [27], and for Chinese word segmentation and POS tagging as a single task [28]. The claimed performance in both studies is competitive with that for state-of-the-art approaches. Long short-term memory (LSTM) networks with different layers have also been used for Chinese word segmentation [29]. The reported experiments show *F1* values above 94% with some network architectures and configurations on all the datasets. Yao and Huang [30] proposed different Bi-LSTM architectures with single, double, and three layers for Chinese word segmentation. They reported that Bi-LSTM with three layers achieved the highest performance on all the test sets in the experiments, attaining an *F1* value above 97%.

III. BACKGROUND

A. ARABIC MORPHOLOGY

Generally, there are two groups of morphemes that Arabic word segmentation can manage: affixes and clitics. Affixes are certain morphemes that can be added to the beginning (prefixes), middle (infixes), or end (suffixes) of the word stem to form a new word or word form. For example, the masculine singular noun طالب "TAlb" (student) can be changed to the feminine singular noun by attaching the suffix ة "p" to become طالبة "TAlbp" (student), and can be changed to the dual masculine singular noun by adding the suffix ان "An" at the end to become طالبان "TAlbAn" (students). The past tense verb كتب "kataba" (wrote) can be changed to present tense by adding ا "A," ن "n," ي "y," or ت "t" to the beginning to become, for example, يكتب "yaktub" (writes).

Clitics, like affixes, are morphemes that are attached to the beginning (proclitics) or end (enclitics) of a word. Unlike affixes, clitics do not change or affect the word form or meaning, or create a new word. Furthermore, Arabic clitics have POS, whereas affixes do not have this property. Clitics, if needed, are attached to a word after the affixation process. Thus, word segmentation can work on two levels: the first is the separation of clitics from the word, and hence word identification, and the second is the separation of affixes from the word, and hence the identification of the word stem. In this paper, we consider the first level of word segmentation, that is, the identification and separation of clitics.

Because of the rules of the Arabic writing system, when clitics are attached to a word, the letters attached to the clitics may be omitted or changed. For example, the word للجامعة "lljAmEp" (for the university) was originally ل+الجامعة "l+AljAmEp," but the letter ا "A" was omitted because of the attachment of the clitic ل "l" to الجامعة "AljAmEp." The letter ت "t" before the clitic ه "h" in the word سيارته "syArth" (his car), for example, needs to be re-written to be ة "p" after segmentation: سيارة+ه "syArp+h."

Attaching clitics to Arabic words can form two, three, or four words, or even a complete sentence. For example, فسيكفيكهم "fsykfykhm" (So will suffice you against them) is a complete sentence containing four clitics ف، س، ك، هم "f, s, k, hm" and one word يكفي "ykfy" (suffices). This case is rare, there are much simpler forms for the concatenation of clitics in Arabic writing system, where certain single morphemes can be concatenated at the beginning of a word, such as و "w," ف "f," and ك "k," or at the end of a word, such as ه "h," هم "hm," and نا "na." It is also possible that single or multiple morphemes can be concatenated at the beginning and end of a word, such as و+ب "w+b" at the beginning and هما "hma" at the end.

Clitics can be a single letter, such as و "w," ب "b," and ه "h;" two letters, such as وا "wA," ها "hA," and هم "hm;" or three letters, such as هما "hmA," تما "tmA," and كما "kmA." Word POS specifies the clitics that can be concatenated at the beginning and/or end. For instance, singular common nouns can accept و "w," ب "b," and ف "f" at the beginning and ها "hA," هما "hmA," and كما "kmA" at the end. Clitics also belong to certain POS tags in the Arabic language. Arabic proclitics can be grouped under particles and enclitics under pronouns.

B. DEEP LSTM NETWORKS

A DNN is a network of multiple layers of nonlinear transformations of data. Given a set of instances of the form $\{(\boldsymbol{x}, \boldsymbol{y})\}_{t=1}^{n}$, where $\boldsymbol{x}_t$ is a feature vector and $\boldsymbol{y}_t$ is a vector representing an instance label, the DNN aims at approximating a function that transforms $\boldsymbol{x}_t$ into $\boldsymbol{y}_t$ with the least amount of error. In our case, that is, word segmentation, the feature vector is a set of symbols from some alphabet and the label is a scalar, that is, a one-component vector, that indicates the segmentation type. There are different types of network architecture. One type that is suitable for processing sequences, and thus natural language, is RNN. In this architecture, the context of an instance is considered during its processing. This is achieved by feeding the result of the mapping from each time step to the succeeding step, as shown in Fig. 1. h_t is a composition of nonlinear transformations and $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ are linear transformation functions.

There are different types of networks based on which the hidden layers in $\boldsymbol{h}$ are structured. One type that is commonly used is LSTM ([31], [32]). This type is known for its ability to capture long-distance contexts, thus making it possible to label the current data instance based on a large window of preceding and succeeding instances. To achieve this, an LSTM state consists of the following functions (Fig. 2 shows how

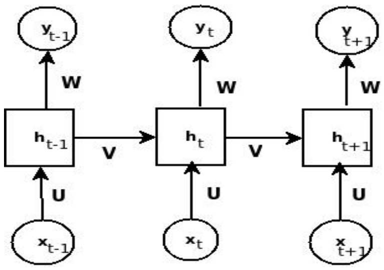

FIGURE 1. Recurrent neural network.

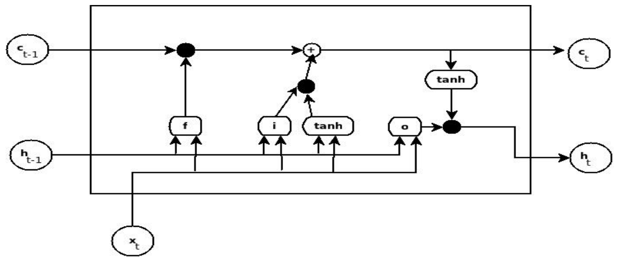

FIGURE 2. LSTM cell.

these functions interact inside the LSTM cell):

$$f_t = sigmoid\left(V_t.h_{t-1} + U_f x_t + b_f\right) \quad (1)$$

$$i_t = sigmoid\left(V_t.h_{t-1} + U_i x_t + b_i\right) \quad (2)$$

$$o_t = sigmoid\left(V_o.h_{t-1} + U_o x_t + b_o\right) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(V_c.h_{t-1} + U_c x_t + b_c) \quad (4)$$

$$h_t = o_t \odot \tanh(c_t) \quad (5)$$

where ***f*** is the forget gate, which is trained to adjust how the information that comes from the previous states is memorized and passed on. ***i*** and ***o*** are the input and output gates, respectively. $\boldsymbol{c}_t$ and $\boldsymbol{h}_t$ are the state and output of cell t, respectively. The black circles in Fig. 2 represent element-wise multiplication. A deep RNN may contain multiple hidden layers of this form. It is obvious from Fig. 2 that the memorization direction is forward. This allows the consideration of the previous data points in the sequence. However, in many language processing tasks, including the one that this work considers, we want to consider the enclosing context from both directions. Therefore, a Bi-LSTM has been proposed in the literature [33]. In this architecture, each hidden layer of the forward LSTM sequence is stacked with a sequence of backward LSTM cells, as shown in Fig. 3. Recent development in deep learning suggests that multiple layers of LSTM cells for both directions are used to model multiple non-linear transformations with context capturing. Therefore, deep LSTM networks have recently been used extensively in natural language processing tasks [27]–[30].

IV. METHOD

Our proposed method is based on a Bi-LSTM neural network with three layers: an input layer, hidden layer, and output layer. During the training phase, and starting from a random state, the hidden layer attempts to learn to map the input to the output using the error backpropagation algorithm. The input layer accepts time series data. Each time series example represents a single sentence. The sentence is fed into the network in several time steps, one character representation at a time. Character data are encoded as vectors to store context information.

The input vector includes a representation of the character itself and its neighbor characters. For example, in our chosen representation, in a window of size five, there are representations for the current character and four other characters, that is, two from each side of the current character.

We use a lookup table to represent characters. In a dictionary of size 50 characters, there are 50 binary features for each character. These characters include Arabic letters, numbers, and punctuation. The first character in the dictionary is encoded to have the value one for the first feature and zero for the remaining features. Similarly, the second character has the value one only for the second feature. The all-zero feature representation is reserved for a special character that indicates the beginning and end of the sentence. Using these settings, for a window size of five, the input vector has 250 binary features (50 features for each of the five characters). The input vector can be further extended with features for word embedding information, as explained in Section IV. Table 1 shows examples from the lookup table.

The output is the label of the current character. We have two label schemes: a basic scheme and an extended scheme. For the basic scheme, the label is either split or no-split. For the split label, we segment the word before the current character. The extended scheme adds eight rewrite labels, as described in Section VI.

A. NETWORK CONFIGURATION

We attempted several settings for the network and we tested it using a development dataset. The best result was achieved

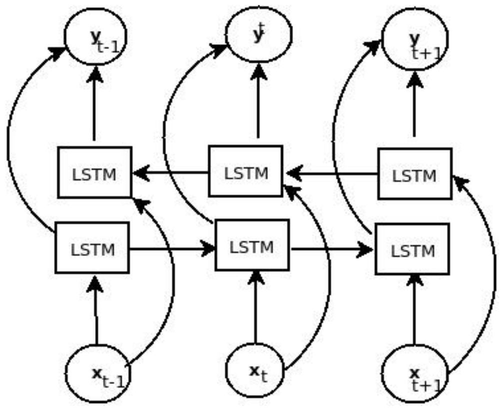

FIGURE 3. Bi-directional LSTM network.

TABLE 1. Lookup table entry example for four characters.

Char	Encoding
ذ	00100000000000000000000000000000000000000000000000
ض	00010000000000000000000000000000000000000000000000
ص	00001000000000000000000000000000000000000000000000
ث	00000100000000000000000000000000000000000000000000

using a Bi-LSTM neural network with three hidden layers. Each hidden layer had 100 nodes. We generally trained the network for three full epochs, with the learning rate value set to 0.1. We used the Deep Learning for Java (DL4J) library[2] for network implementation, training, development, and evaluation.

B. WORD EMBEDDING

We also used the DL4J library to build a word2vec model. The model was built using a data dump from Wikipedia for 2013 for Arabic content. The corpus contained over 48M tokens, with approximately 311K distinct tokens. The corpus was used in its original form without segmentation. We built the word2vec model using different window sizes, with a vector size of 100 features and a minimum frequency of five occurrences. Based on our experiments, we found that a window size of three tokens (one before the word and one after the word) was sufficient for the present task. The size of the final model was approximately 490MB. Table 2 lists some words with their closest words using this word2vec model.

TABLE 2. Example of the closest words using the word2vec model.

Word	Closest Words
محمد	أحمد، إبراهيم، مباركي، الحركان، الخوجة
والحصان	واللاما، والثور، والوعل، والنمس، المرقط
اصفر	مصفر، داكن، شاحب، للصفرة، محمر
بيته	داره، يعودونه، مسجده، أهله، دكانه
حسابك	الزبون، المخترق، حاسبك، السيرفر، جهازك
بالرياض	بجدة، بالدمام، بالظهران، بجامعه، بالقصيم
أراضيهم	أرضهم، وأراضيهم، اراضيهم، ممتلكاتهم، ثرواتهم
للاجئين	مخيم، مخيمات، اللاجئين، الأنروا، الدهيشة
هويته	ماضيه، امتناعه، رأيه، أعذار، ويكشف

The word2vec data were added to the character information in the input vector. For each character, we added the word2vec data of the word to which the character belonged. Thus, with an input vector of size 250 features of context data of five characters, we added 100 features. This increased the size of the input vector to 350 features.

V. DATASET

The dataset used in our experiments was ATB (Parts 1–3)[3] [34]. It contains approximately 22K sentences that include approximately 620K words. For the experiments, we split the data into three datasets: training (60%), development (20%), and evaluation (20%).

Word segmentation occurred in 98,597 words (approximately 16%) in the dataset. Approximately 94% of these words had a single segmentation (e.g., و+محمد "w+muHam∼ad" "and Mohammad"). Most of the remaining words had two segmentations (e.g., ب+بيت+ك "b+byt+k" (in your house), and there were only a few occurrences of words with three segmentations (e.g., و+ل+أطفال+هم "w+l+>TfAl+hm" "and for their children"). Table 3 shows all the clitics in the corpus and their counts. Because of the ambiguity of some clitics and the fact that we used simple text patterns for counting, the count may not be very precise. However, these numbers provide an excellent estimation of the distribution of all the clitics in the corpus. As an example of clitic ambiguity, the word كم "kam" in و+كم "w+kam" is counted as an instance of the clitic +كم "km+" as in ثياب+كم "vyAb+km" (your clothes).

TABLE 3. Counts of all the clitics in the corpus with example words.

Clitic	Count	%	Example
و+	43,967	41.09	و+طرحت
ل+، +ل+	16,169	15.11	ل+شركة
ب+، +ب+	13,080	12.22	ب+شكل
+ه	12,814	11.98	سعر+ه
+ها	10,811	10.10	جهود+ها
+هم	3,136	2.93	أراضي+هم
ف+	2,228	2.08	ف+مثلا
+نا	1,724	1.61	عصر+نا
ك+، +ك+، +ك	742	0.69	ك+المعتاد
+ما	675	0.63	عن+ما
+هما	636	0.59	مناطق+هما
+ي	504	0.47	عين+ي
+ني	203	0.19	يسعد+ني
+كم	194	0.18	كتاب+كم
+ا	30	0.03	عن+ا
+هن	24	0.02	طاقات+هن
+كن	20	0.02	بيوت+كن
+كما	20	0.02	بين+كما
أ+	18	0.02	أ+ليس
Total	**106,995**		

Table 3 shows that the coordinator و+ "w+" was the most frequent clitic in the corpus. It represented approximately 41% of the clitics. Four other frequent clitics were the prepositions ل+ "l+" and ب+ "b+," and the singular pronouns +ه "+h" and +ها "+hA." These five frequent clitics represented approximately 90% of the clitics in the corpus. The remaining 14 clitics covered less than 10% of the clitics in the dataset. These statistics suggest two main conclusions. First, any word segmenter should precisely manage the five frequent clitics for it to be a successful segmenter. Second, it would be difficult for a machine learning-based segmenter to learn a sufficient amount about the less frequent clitics that, most of the time, have very few examples in the dataset.

A. DATA FORMAT

We extracted the segmentation data from the ATB and stored it in text files, sentence by sentence. Each file contained information for a single sentence. Each sentence was considered as a time series instance. The file contained a number of lines equal to the number of characters in the sentence. In each line, we stored the input vector data and output label.

[2]The DL4J library can be downloaded from https://deeplearning4j.org .

[3]The Penn ATB parts have several versions. In this paper, we used Part 1 version 3.0, Part 2 version 2.0, and Part 3 version 2.0.

The output label is the tag of the current character, which can be either 1 or 0 for the spilt/no-split scheme, or a value from 0 to 9 for the extended rewrite scheme.

B. REWRITE INFORMATION

Character rewrite information is not provided in the ATB; the information in the ATB only marks the split positions. We used MADAMIR (Version 2.1) to collect the rewrite information and added it the corpus. MADAMIR can process segmentations that require rewriting with good accuracy. We identified all the rewrite cases in the corpus and used MARAMIRA to provide solutions, and reviewed and corrected these solutions manually.

TABLE 4. Labels for the extended scheme with counts and examples.

Label	Description	Example	Count
0	no-split	بيت	3,232,458
1	split (no-rewrite)	بيته ⇦ بيت+ه	92,948
2	the لل case	للبيت ⇦ ل+البيت	6,162
3	the ي case	إليه ⇦ إلى+ه	1,437
4	the ت case	سيارته ⇦ سيارة+ه	3,065
5	the ئ/ؤ case	أدائه ⇦ أداء+ه / حلفاؤه ⇦ حلفاء+ه	416
6	the ا case	مداه ⇦ مدى+ه	304
7	the و case	تلقوها ⇦ تلقوا+ها	87
8	the آ case	رآه ⇦ رأى+ه	6
9	the للّ case	للاجئين ⇦ ل+اللاجئين	89

Table 4 shows examples and counts for all the segmentation and rewrite cases in the corpus. Approximately 89% of the segmentation cases required no rewrite. The most frequent rewrite case in the corpus was the preposition ل "l" when added to a noun with the definite article ال "Al" (approximately 6%). The other two frequent rewrite cases were the letters ي "y" and ت "t." "ت." The remaining cases had a low frequency in the dataset, particularly the letter آ "|,"which only had six examples. As mentioned previously, the low frequency of some rewriting cases made it difficult for the model to predict it.

VI. EXPERIMENTS

We used the development data to select the best settings for the proposed method. Table 5 shows the main configuration of the best settings that were used for all the experiments in this paper. Using this configuration, we conducted two main experiments. In the first experiment, we investigated the performance of word segmentation without rewriting. In the second experiment, we investigated the performance of word segmentation with 10 cases of rewriting.

A. SEGMENTATION WITHOUT REWRITING

In this experiment, we built models for the basic scheme. Table 6 shows the scores for two models. The basic configuration achieved a very good result ($F1 = 97.65\%$). Adding the word embedding information to the model slightly improved the result by 0.38 points to reach 98.03%.

TABLE 5. Basic network configuration for the proposed method.

Setting	Value
Network Type	Bi-LSTM
Number of Hidden Layers	3
Number of Hidden Nodes	100
Learning Rate	0.1
Batch Size	10
Epochs	(3, 10)
Dictionary Size	50
Window Size	5
word2vec vector size	100
word2vec window Size	3

TABLE 6. Results for the basic model with and without word embedding.

Settings	F1 (%)
Basic Configuration	97.65
+ Word Embedding	98.03

TABLE 7. Results for the rewrite model using two datasets.

Label	Rewrite Model	+ More Data	Score Change
	F1 (%)	F1 (%)	
0	99.93	99.93	0.00
1	97.44	97.55	+0.11
2	99.61	99.09	-0.52
3	99.84	99.36	-0.48
4	89.55	93.62	+4.07
5	95.30	96.60	+1.30
6	78.67	75.16	-3.51
7	55.17	11.76	-43.41
8	0.00	0.00	0.00
9	30.00	76.47	+46.47
Total			+4.03

B. SEGMENTATION WITH REWRITING

The results for the rewrite model are shown in Table 7. The scores are shown for each label in the extended scheme. The second column shows the results for the first rewrite model. This model was trained for 10 epochs instead of three. Using the development data, we noticed that we needed more epochs to enhance the accuracy for low-frequency labels in the training dataset. With the exception of low-frequency labels, the model achieved excellent scores. Five labels scored above 95% on the *F1* scale. The model failed to learn any knowledge about the least frequent label in the dataset, *Label 8*, which only had six examples in the corpus.

As an attempt to improve the score for the low-frequency labels, we attempted to train the model using more data. Thus, we combined the training data and development data in one dataset and used it for training. The results for this model using the evaluation dataset are shown in the third column of Table 7. These results show an overall improvement of 4 points. Most results for the individual labels showed slight changes or remained unchanged. The results for the less frequent labels were mixed. *Label 9* had an excellent boost of approximately 46 points. By contrast, the score for *Label 7* declined by almost the same amount: 43 points. Even though these outcomes may not lead to clear conclusions, the results for less frequent labels are most likely to improve with substantially more training data.

VII. COMPARISON WITH STATE-OF-THE-ART ARABIC WORD SEGMENTERS

In this section, we compare the proposed method with the main and state-of-the-art methods in the literature. Most of these methods, including in this study, were trained and/or tested using ATB. However, the results may not be directly comparable for several reasons, such as using a different split or different version of the corpus. A fair comparison requires using new testing data that were never processed by any of the methods. Therefore, we selected two texts, one from MSA and the other from classical Arabic, and preformed the required word segmentations manually and used them for comparison.

We compared our proposed method with the following state-of-the-art methods:

1) Farasa[4] [7]
2) MADAMIRA [18] (version 2.1)
3) Stanford Arabic Segmenter [5] (version 3.3.1)
4) IBM Arabic Word Segmenter [26]

A. MSA TEST

We randomly selected an article from the *Al Riyadh* newspaper to prepare the first test. The article was short to allow for a rough comparison. The article had 646 tokens with 129 segmentation cases. We only considered common segmentation cases that were shared among all the methods. For example, we did not count ال "Al" segmentation case because it was not considered by all the methods.

All the tested methods achieved excellent results in this test. Out of the 129 segmentation cases, there were only 20 segmentation cases that were somehow challenging. Table 8 shows the results for this test. Our proposed model achieved the best results, with only two segmentation errors. Farasa and MADAMIRA were ranked next, with three and four errors, respectively. Both, IBM and Stanford, generated nine errors. However, most of the IBM errors were rewrite errors. The IBM method only performs segmentation without rewriting.

B. CLASSICAL ARABIC TEST

The results of the comparison for the previous test may not be sufficient because of the small size of the test. Therefore, for this experiment, we chose another text and made it as difficult as possible.

[4]An online demo of Farasa can be accessed at http://qatsdemo.cloudapp.net/farasa/demo.html

TABLE 8. Comparison of five segmentation methods using a newspaper article. Segmentation errors are shown for each method.

No.	Correct Segment.	Proposed Method	Farasa	MADAMIRA	Stanford	IBM
1	ب+مؤلفات+هما				ب+مؤلفاة+هما	
2	ف+رغم				فرغم	
3	و+قد		وقد			وقد
4	ل+الحج					ل+لحج
5	ل+الصعوبات					ل+لصعوبات
6	ب+عدة				بعدة	
7	ل+الحجاج					ل+لحجاج
8	ف+رغم				فرغم	
9	ل+يستمر				ليستمر	
10	تطويرية			تطويري+ه		
11	ب+حكمة+ه		ب+حكم+ت+ه			ب+حكمت+ه
12	ك+فتح	كفتح			كفتح	
13	ل+الشركات					ل+لشركات
14	ل+كيان					لكيان
15	ل+مزدلفة			لمزدلفة		
16	و+جدة				وجدة	
17	ب+نقلات			بنقلات		
18	أبناء+ه		أبناؤ+ه		ابناؤ+ه	أبناؤ+ه
19	تتبع+نا				تتبعنا	
20	ل+وجد+نا	ل+وجدنا		ل+وجدنا		ل+وجدنا
Number of Errors		**2**	**3**	**4**	**9**	**9**

The new text was a famous classical Arabic poem by Imru' al-Qais from the sixth century. As mentioned previously, the test was difficult because it contained different vocabulary and syntax patterns that may never have been seen in the training data. Despite this, classical and modern Arabic still share many common features that make it possible to make the required connections.

The classical poem contains 791 tokens with 294 segmentation cases. We ran the test using the five methods, and the results showed lower performance than was expected. Out of the 294 cases, there were 156 segmentation cases with errors based on one or more methods. Farasa achieved the best accuracy, with only 39 segmentation errors. MADAMIRA ranked second with 51 errors. Our proposed method, IBM, and Stanford generated 63, 80, and 91 errors, respectively.

Table 9 shows the results of this experiment. The table shows the unique segmentation errors for each method. These words were correctly segmented by all the methods, except for the given method. Most of MADAMIRA unique errors were for rare classical Arabic words. MADAMIRA worked better with common words. Farasa had only two unique errors, which means it was good for both rare and common (non-rare) words.

Our proposed method achieved the best result among the three non-dictionary-based ML methods. Most of the unique errors for these three methods were for common words, which means that rare and common words were considered in the same manner by these methods.

TABLE 9. Comparison of five segmentation methods using a classical Arabic poem. Unique segmentation errors are shown for each method.

Method	Unique Segmentation Errors	Total Errors
Farasa	فأدبرن، وجر+ة	39
MADAMIRA	ب+طن، بالذبال، بالكديد، بالمتنزل، ببيسان، بعدما+ما، بماسل، بمنجرد، بمنسل، جواحرها، حلف+ه، كالجديل، كالسجنجل، وارخاء، والت، وتعطو، ومجول	51
Proposed Method	امر+أ، بضاف، رى+هب، ف+ألحقنا، ك+تان، كجلمود، كقنو، كموج، ل+بسة، للك، نب+ك، و+ب+يضة	63
IBM	أمره، اليمان+ي، ب+نحره، بصلبه، بماء، ت+ك، زمامه، سدوله، صوبه، غل+ي، قبلها، ل+لعذار+ي، له، مساوي+ك، منهما، نجومه، و+دونه، و+ل+جامه، وأردف	80
Stanford	بدارة، ثيابك، ثيابها، حبي، رسمها، عنك، ف+الهية+ها، فاضحى، فراشها، فظل، كهداب، منك، مني، نسجة+ها، و+حش، و+س+اقا، و+قوفا، وراءنا	91

VIII. CONCLUSIONS

In this paper, we addressed the problem of Arabic word segmentation, both as a standalone approach and with a rewrite. Our approach was based solely on learning from data. We used a Bi-LSTM neural network. Our proposed method obtained a competitive result compared with state-of-the-art systems. In our experiments, the proposed method attained the lowest number of segmentation errors for an MSA test. On a text from classical Arabic, a poem, the proposed method attained the lowest number of errors among the systems that do not rely on language resources, but it was third following Farasa and MADAMIRA, which use language resources. Based on our observations, a robust Arabic word segmentation system can be learned from data if the training set covers a large vocabulary of the intended language and most of the segmentation rules.

REFERENCES

[1] N. Zalmout and N. Habash, "Optimizing tokenization choice for machine translation across multiple target languages," *Prague Bull. Math. Linguistics*, vol. 108, no. 1, pp. 257–269, 2017.

[2] A. Al-Thubaity and A. Al-Subaie, "Effect of word segmentation on Arabic text classification," in *Proc. IALP*, Suzhou, China, Oct. 2015, pp. 127–131.

[3] S. Green and C. D. Manning, "Better Arabic parsing: Baselines, evaluations, and analysis," in *Proc. 23rd Int. Conf. Comput. Linguistics*, Beijing, China, Aug. 2010, pp. 394–402.

[4] Y. Benajiba and I. Zitouni, "Arabic word segmentation for better unit of analysis," in *Proc. LREC*, Valletta, Malta, May 2010, pp. 1352–1364.

[5] W. Monroe, S. Green, and C. D. Manning, "Word segmentation of informal Arabic with domain adaptation," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA, vol. 2, 2014, pp. 206–211.

[6] N. Zalmout and N. Habash, "Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Copenhagen, Denmark, 2017, pp. 704–713.

[7] K. Darwish and H. Mubarak, "Farasa: A new fast and accurate Arabic word segmenter," in *Proc. LREC*, Paris, France, 2016, pp. 1070–1074.

[8] S. Khoja, "APT: Arabic part-of-speech tagger," Ph.D. dissertation, Comput. Dept., Lancaster Univ., Lancaster, U.K., 2003.

[9] C. B. O. Zribi, A. Torjmen, and M. B. Ahmed, "A multi-agent system for POS-tagging vocalized Arabic texts," *Int. Arab J. Inf. Technol*, vol. 4, no. 4, pp. 322–329, 2007.

[10] S. Alqrainy, H. M. AlSerhan, and A. Ayesh, "Pattern-based algorithm for part-of-speech tagging Arabic text," in *Proc. Int. Conf. Comput. Eng. Syst.*, Nov. 2008, pp. 119–124.

[11] A. Al-Taani and S. A. Al-Rub, "A rule-based approach for tagging non-vocalized Arabic words," *Int. Arab J. Inf. Technol.*, vol. 6, no. 3, pp. 302–328, 2009.

[12] M. Hadni, S. A. Ouatik, A. Lachkar, and M. Meknassi, "Hybrid part-of-speech tagger for non-vocalized Arabic text," *Int. J. Natural Lang. Comput.*, vol. 2, no. 6, pp. 1–15, 2013.

[13] S. AlGahtani, W. Black, and J. McNaught, "Arabic part-of-speech tagging using transformation-based learning," in *Proc. 2nd Int. Conf. Arabic Lang. Resour. Tools*, Cairo, Egypt, Apr. 2009, pp. 66–70.

[14] S. Kulick, "Exploiting separation of closed-class categories for Arabic tokenization and part-of-speech tagging," *ACM Trans. Asian Lang. Inf. Process.*, vol. 10, no. 1, 2011, Art. no. 4.

[15] F. Al Shamsi and A. Guessoum, "A hidden Markov model-based POS tagger for Arabic," in *Proc. 8th Int. Conf. Stat. Anal. Textual Data*, Besançon, France, 2006, pp. 31–42.

[16] M. Diab, K. Hacioglu, and D. Jurafsky, "Automatic tagging of Arabic text: From raw text to base phrase chunks," in *Proc. HLT-NAACL*, Boston, MA, USA, 2004, pp. 149–152.

[17] M. Diab, "Second generation AMIRA tools for Arabic processing: Fast and robust tokenization, POS tagging, and base phrase chunking," in *Proc. 2nd Int. Conf. Arabic Lang. Resour. Tools*, Cairo, Egypt, 2009, pp. 285–288.

[18] A. Pasha *et al.*, "MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic," in *Proc. LREC*, Reykjavik, Iceland, 2014, pp. 1094–1101.

[19] A. van den Bosch, E. Marsi, and A. Soudi, "Memory-based morphological analysis and part-of-speech tagging of Arabic," in *Arabic Computational Morphology*. Dordrecht, The Netherlands: Springer, 2007, pp. 201–217.

[20] M. Abdul-Mageed, M. Diab, and S. Kübler, "ASMA: A system for automatic segmentation and morpho-syntactic disambiguation of Modern Standard Arabic," in *Proc. Recent Adv. Natural Lang. Process.*, Hissar, Bulgaria, 2013, pp. 1–8.

[21] K. Darwish, A. Abdelali, and H. Mubarak, "Using stem-templates to improve Arabic POS and gender/number tagging," in *Proc. LREC*, Reykjavik, Iceland, 2014, pp. 2926–2931.

[22] S. Khalifa, N. Zalmout, and N. Habash, "Yamama: Yet another multi-dialect Arabic morphological analyzer," in *Proc. 26th Int. Conf. Comput. Linguistics, Syst. Demonstrations (COLING)*, Osaka, Japan, 2016, pp. 223–227.

[23] N. Habash and O. Rambow, "Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, Lansing, MI, USA, 2005, pp. 573–580.

[24] R. Roth, O. Rambow, N. Habash, M. Diab, and C. Rudin, "Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking," in *Proc. 46th Annu. Meeting Assoc. Comput. Linguistics Hum. Lang. Technol.*, Jun. 2008, pp. 117–120.

[25] A. A. Freihat, G. Bella, H. Mubarak, and F. Giunchiglia, "A single-model approach for Arabic segmentation, POS tagging, and named entity recognition," in *Proc. 2nd Int. Conf. Natural Lang. Speech Process.*, Algiers, Algeria, Apr. 2018, pp. 1–8.

[26] Y. S. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan, "Language model based Arabic word segmentation," in *Proc. 41st Annu. Meeting Assoc. Comput. Linguistics*, Sapporo, Japan, vol. 1, Jul. 2003, pp. 399–406.

[27] X. Chen, X. Qiu, Z. Zhu, and X. Huang, "Gated recursive neural network for Chinese word segmentation," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, 2015, pp. 1744–1753.

[28] X. Zheng, H. Chen, and T. Xu, "Deep learning for Chinese word segmentation and POS tagging," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Olympia, WA, USA, 2013, pp. 647–657.

[29] X. Chen, X. Qiu, C. Zhu, P. Liu, and X. Huang, "Long short-term memory neural networks for Chinese word segmentation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1197–1206.

[30] Y. Yao and Z. Huang, "Bi-directional LSTM recurrent neural network for Chinese word segmentation," in *Proc. 23rd Int. Conf. Neural Inf. Process.*, Kyoto, Japan, 2016, pp. 345–353.

[31] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[33] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal. Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[34] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki, "The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus," in *Proc. NEMLAR Conf. Arabic Lang. Resour. Tools*, vol. 27, Sep. 2004, pp. 466–467.

ABDULRAHMAN ALMUHAREB received the B.Sc. degree in information systems from King Saud University, Riyadh, Saudi Arabia, in 1993, the M.Sc. degree in computer science from Ball State University, IN, USA, in 1998, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2006. He is currently a Research Assistant Professor of computer science with the National Center for Artificial Intelligence and Big Data Technology, King Abdulaziz City for Science and Technology, Riyadh. His research interests include Arabic language processing, text data mining, and search engines.

WALEED ALSANIE received the B.Sc. and M.Sc. degrees from King Saud University, Riyadh, Saudi Arabia, in 2002 and 2006, respectively, and the Ph.D. degree from the University of York, U.K., in 2013. He has been active in research and development, since 2002, as a Software Engineer with the Research and Development Department, Advanced Electronics Company, as a System Engineer with Saudi Telecom Company, and a Senior Researcher with King Abdulaziz City for Science and Technology (KACST). He is currently a Research Assistant Professor with the National Center for Artificial Intelligence and Big Data Technology, KACST. His research interests include machine learning, natural language processing, and probabilistic logic.

ABDULMOHSEN AL-THUBAITY received the B.Sc. degree in electrical and computer engineering from Umm Al-Qura University, Makkah, Saudi Arabia, in 1990, the M.Sc. degree in computer engineering from King Abdulaziz University, Jeddah, Saudi Arabia, in 1995, and the Ph.D. degree in computer science from the University of Surrey, Surrey, U.K., in 2004. He is currently a Research Associate Professor of computer science with the National Center for Artificial Intelligence and Big Data Technology, King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia. His research interests include Arabic computational linguistics, Arabic language processing, Arabic text mining, and Arabic language resources.