International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 8, No.3 (May-2019)

http://dx.doi.org/10.12785/ijcds/080310

Automatic Arabic Part-of-Speech Tagging: Deep Learning Neural LSTM Versus Word2Vec

Khwlah Alrajhi[1] and Mohammed A ELAffendi[2]

[1,2] EIAS Data Science & Blockchain Lab, Department of Computer Science, Prince Sultan University, Riyadh, Saudi Arabia

Received 30 Oct. 2018, Revised 31 Dec. 2018, Accepted 28 Apr. 2018, Published 1 May 2019

Abstract: Part-of-speech (POS) tagging is the process of selecting an appropriate POS tag for each word in a natural language sentence. POS tagging is a vital part of most natural language processing (NLP) applications. In comparison to other languages, there is a dearth of studies on NLP applications for the Arabic language. Recently, neural networks (NNs) and deep learning technologies have shown excellent results for some English and Latin NLP applications. However, for Arabic, the practice is still in its infancy, and more work is needed to determine whether neural technologies will lead to convincing results for NLP applications. In this paper, a long short-term memory (LSTM) model has been used to investigate the effectiveness of NNs in Arabic NLP. The model has been specifically applied to identify the POS tags for Arabic words and morphemes taken from the Quranic Arabic Corpus (QAC) data set. QAC is a well-known gold standard dataset prepared by researchers from Leeds University. It is interesting to note that LSTM tagger achieved 99.72% accuracy for tagging morphemes and 99.18% for tagging words, while the Word2Vec tagger achieved 99.55% for tagging morphemes and 97.33% for tagging words.

Keywords: Arabic parts of speech, Deep learning, Long short-term memory (LSTM), Neural network (NN), Recurrent neural network (RNN), Tag, Word embedding, Word2Vec.

1. Introduction

The main focus of NLP is to make a computer capable of interacting with people using natural human languages. One of the fundamental tasks in NLP is part-of-speech (POS) tagging, which is the process of identifying the type (tag) of a given word, such as a noun, verb, pronoun, or adverb, in an input sentence [1]. POS tagging is a prerequisite for many high-level NLP applications, such as information extraction, the process of deriving structured factual information from unstructured text [2]; machine translation, the process of translating a text from a natural language by a computer; parsing, the task of assigning a syntactic structure to a sentence [3]; and many others.

Natural languages are naturally ambiguous, subjective, and complex. Ambiguity appears at different levels of the NLP process [4]. This implies that the development of NLP applications is very challenging compared to that of programming languages, which are highly structured and deterministic [5]. Several studies have been conducted to automate the POS tagging process. Most of the early studies are rule-based and require the development of collections of rules that are relatively complex and not flexible. Lately, probabilistic corpus-based approaches have become popular. In these approaches, tagged corpora are used to train models that identify tags based on the context, which is captured using n-gram probabilities. These models have proven to be more accurate than the rule-based approaches but require the preparation of a large, manually tagged corpus, which is an obvious overhead [6].

Recently, there have been several successful attempts in using neural networks (NNs) and deep learning to develop NLP and POS tagging systems for many languages. Although Arabic is the standard language for more than 20 countries in North Africa and the Middle East, NLP research in the Arabic language is lagging, and very few attempts for using deep learning technologies have been reported in this area.

The aim of this paper is to show how NNs and deep learning can be used to build a highly accurate automated Arabic POS tagger [7]. A long short-term memory (LSTM) recurrent neural networks (RNNs) [8] has been successfully used to build a highly accurate tagger that compares favourably with all reported taggers. The main advantage of LSTM RNNs is their ability to learn long-term dependencies without the need to compute explicit probabilities [8]. In addition, one of the major advantages of the LSTM neural model is that no feature engineering is required. Accordingly, morphological segmentation is not explicitly needed. However, morphological

E-mail: khrajhi@psu.edu.sa, affendi@psu.edu.sa

segmentation for our data is performed using our underling Sliding Window Asymmetric Matching (SWAM) Algorithm [9], and the accuracy of the SWAM algorithm is well above 98% when using the latest version.

2. Literature Review

Arabic POS tagging has been gaining traction recently due to the increasing importance of tagging in building modern NLP and artificial intelligence applications. Some efforts have been made for Classical Arabic (CA) and Modern Standard Arabic (MSA) during the past few years. These can be classified into four categories: rule-based approaches [10], [11], probabilistic corpus-based approaches [9], [12], [13], hybrid methodologies [14], [15], and machine learning approaches [16], [17], [18], [19].

A. *Machine Learning for POS Tagging Arabic language*

Habash and Rambow [16] applied support vector machines (SVMs) to identify Arabic tagsets. In their view, POS tagging is a classification problem that depends on many features. For Arabic, the features are more complex than those of other languages, which means the tagsets for Arabic are much larger than those for Germanic languages, such as English. While tagsets for English consist of 50 elements at most, for Arabic, 2,200 morphological tags have been used for tagging the first 280,000 words in the Penn Arabic Treebank out of 333,000 words completely specified in the morphological analysis. Therefore, Habash and Rambow developed a morphological analyser which is supported by a morphological disambiguation approach for Arabic POS tagging and a machine learning classification component. They reported that morphological tagging and tokenisation were similar operations that involved three steps:

- Collection of all appropriate tags from the morphological analyser for an assigned word.
- Application of machine learning classifiers on the words of a sentence. Through this approach, the value, feature and class of a word can be identified. Habash and Rambow used Yamcha, an SVM that includes Viterbi decoding.
- Application of classifiers to choose the expected morphological tag.

As a result, Habash and Rambow obtained accuracy rates on all tasks in the 90% range. They found enough information on affixes and clitics for good tokenisation performance, and they performed POS tagging, disambiguation, and morphological tokenisation in a single round using this approach. It was observed in this experiment that the classifiers used may be partially disambiguated, which creates some errors in choice matching. So, increasing the efficiency of classifiers opens new directions in the research field. For future researches, in order to address the Arabs community needs, a fully automated POS tagging system is required for the Arabic language.

A similar improved SVM approach was used by Diab, Hacioglu, and Jurafsky [17] that performs tokenisation, POS tagging, and base phrase chunking of Arabic text automatically. Actually, they selected the most efficient tools used for POS tagging in English and applied them to Arabic texts. In POS tagging, they modelled this as a 1-of-24 classification task in which the class labels are POS tags. However, if a token did not occur in the training data, it was assigned a Noun tag by default; However, 50% of the errors resulted from confusing nouns, with adjectives or vice versa. To prepare the data, they converted the Arabic Treebank into Latin-based abbreviated from American Standard Code for Information Interchange (ASCII) characters using the Buckwalter transliteration scheme. Not only did they report that the SVM-POS tagger achieved 95.49% accuracy, but they also promised to further improve the performance of the system using additional features, a wider context, and more data.

Ben Ali and Jariri [18] developed a different Arabic POS tagger using genetic algorithms that assigned POS tags to the input text. The findings of their research study suggest that the genetic algorithm approach is more robust than other statistical systems for tagging natural language texts, achieving approximately 94% accuracy. Unfortunately, they did not perform segmentation, which is an important operation for Arabic morphology.

Much work has been achieved in Arabic POS tagging using the sequence of clitics in a word simultaneously. Different approaches have been used for this purpose. Darwish, Mubarak, Abdelali, and Eldesouki [19] evaluated the comparative performance of two different methods for POS tagging. They compared applications of SVM-based ranking and bidirectional long-short-term memory (bi-LSTM) NN-based sequence labelling in building Arabic POS tagging system. They also found that adding explicit features to the bi-LSTM NN and employing word embeddings separately improved POS tagging results. The accuracy result was 95.50%, which, though promising, was not as much so as rule-based taggers. However, this work is useful for applying NN to Arabic POS tagging.

B. *Deep learning for POS Tagging other languages*

Wang, Qian, Soong, He, and Zhao [20] introduced a novel approach called the bidirectional long short-term memory recurrent neural network (Bi-LSTM-RNN), which the authors used in word embedding for POS tagging without the employment of morphological features. With the Bi-LSTM-RNN, they implemented six types of word embedding for comparison and found that all instances exhibited high accuracy. In demonstrating that competitive tagging accuracy (around 97%) can be achieved without the benefit of morphological features, the authors illustrated the advantage of the Bi-LSTM-RNN as a method of tagging a language that lacks necessary morphological knowledge.

Plank, Søgaard, and Goldberg [21] explored automating POS tagging based on an LSTM network and proposed a novel multi-task bi-LSTM model with an auxiliary loss function that improves the accuracy with which specific words are tagged. Under varying conditions and multiple parameters, including data size and label noise, the authors assessed token and sub-token-level representations of NN-based POS tagging across 22 languages. They indicated that sub-token representations are necessary to obtain a state-of-the-art POS tagger and that character embedding is particularly helpful for non-Indo-European and Slavic languages. Finally, the authors confirmed that their bi-LSTM tagger exhibits competitive performance on data from the Wall Street Journal, for which POS accuracy reached 97.3%.

3. Proposed Methodology

As stated earlier, the main purpose of this paper is to investigate the possibility of building a highly accurate automated Arabic POS tagger using NNs and deep learning technologies. To achieve this goal, a two-level research methodology was used. At the upper (macro) level, the methodology is a combination of exploration, experimentation, recommendations, and design. At the lower level, a typical machine learning process has been applied to train and test the models recommended by the upper (macro) process. Typical phases of a machine learning process include data collection, preprocessing, transformation to the fit input/output requirements of an underlying model, model configuration, training, testing, and analysis.

A. *The Upper (Macro) Level Process*

- Exploration Phase: The main purpose of the exploration phase is to conduct a detailed survey of potential NN models and identify the most appropriate Arabic tagset to be used in the experiment. In addition to selecting the most relevant neural models, the exploration phase surveys the types of algorithms and approaches used in training and testing NN models. The exploration phase also covers the types of machine learning development platforms and utilities used in training and testing NNs. This phase ends with recommendations regarding the above three issues: type of model to be used, appropriate tagset, and recommended platform.
- Initial Experimentation Phase: The aim of this phase is to perform initial experimentation and comparison of the most relevant NN models selected in the previous step. At the same time, an evaluation of the appropriate word embeddings and text encoding techniques is conducted.
- Design: The results obtained from the experimentation performed in Step 2 are thoroughly analysed, and concrete architectures for the models to be used are developed.
- Implementation: The designed models are implemented, trained, tested, and evaluated.

B. *Lower (Micro) Level*

At the lower (micro) level, Step 4 of the macro level (implementation) is conducted again using typical machine learning cycles, which involve

- data preparation and preprocessing;
- data wrangling or munging;
- word embedding and encoding;
- training the model using appropriate algorithms;
- evaluating the model; and
- repeating the above steps until the performance of the model is satisfactory.

The same steps, with slight modifications, are used in the testing and deployment phases.

4. Proposed Model

This section covers the type of NN model, the Arabic tagset, and the platform that were used. It also describes the method of preparing data, the encoding approach and how the approach works.

A. *LSTM*

Based on the adopted methodology and experimentation with a selection of neural models, the LSTM recurrent neural model [19] has been identified as the strongest candidate for the purpose of our work. It has been shown that this model is suitable for modelling long-range dependencies in several POS tagging studies [20], [21]. Based on the results proposed by [19], we concluded that LSTM is an appropriate NN to use for the Arabic POS tagging task. LSTM architectures designed to deal with the vanishing gradient problem have been proven to learn long-term dependencies more efficiently through internal units [22]. In a repeated LSTM module, there are four interacting layers, unlike a single RNN layer that contains a simple sigmoid or tanh layer. LSTM uses the hidden state from the previous time step and the current input with these four layers, which interact in a particular way to implement recurrence. Fig. 1 represents the LSTM cell at time step t. The internal memory of the LTSM unit is represented by cell state c, which is indicated by the horizontal line at the top of the figure. The horizontal line on the bottom represents the hidden state. The LSTM cell is able to remember important things from the present, and forget unimportant things from the past, by using LSTM gates. The *i*, *f*, *o* and *g* gates let LSTM solve the vanishing gradient problem.

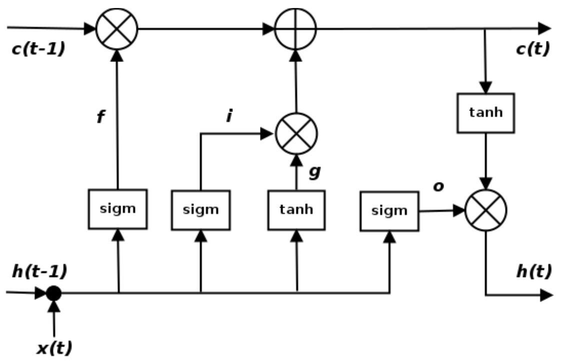


Figure 1. LSTM Cell [22].

LSTM gates can be represented by equations that explain how to calculate the hidden state, h_t, at time t from the hidden state at the previous time step, h_t-1. Values for short or long times are stored by cell C_t. In the first step, the LSTM tagger determines the importance of data to be stored in a cell with the help of a forget gate layer, also known as a sigmoid layer. The principle is based on computation of data from h_t-1 and x_t by generating a number between 0 (completely discard set) and 1 (save entire set) for every number in cell state C_t-1.

$$f_t = \sigma (Wf.[h_{t-1},x_t] + b_f) \quad (1)$$

The flow of the new value in the memory is controlled by an input gate, which also determines the value updating. The vector of the new value, $\tilde{C}_t$, is created by the tanh layer of the state, which is then combined to create an update.

$$i_t = \sigma (W_i.[h_{t-1},x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = \tanh (W_c.[h_{t-1},x_t] + b_c) \quad (3)$$

The decisions for remaining values in the cell are controlled by the forget gate, which update the new state of cell $\tilde{C}_t$ from $\tilde{C}_{t-1}$. The old state is multiplied by *ft* and added to $i_t * \tilde{C}_t$. The decisions for this new candidate value scale are then updated.

$$\tilde{C}t = ft * Ct-1 + it * \tilde{C}t \quad (4)$$

The filtered version of the cell state is based on this output. The output of the cell state is determined by running the sigmoid layer. Through the tanh layer, values between -1 and 1 are multiplied by the sigmoid gate output.

$$Ot = \sigma (Wo[ht-1,bo] + bo) \quad (5)$$

$$h_t = O_t \tanh(C_t) \quad (6)$$

B. *Dataset*

Based on a survey of available datasets, the Quranic Arabic Corpus (QAC) was the most suitable candidate for training and testing our model. The QAC is a gold standard that has been manually created and verified [23]. The QAC dataset consists of a sequence of Quranic verses, which consists of a sequence of words. Each word is composed of a stem, prefix, and suffix. Fig. 2 provides an example of word segments. Each segment of a given word is referred to as a morpheme.

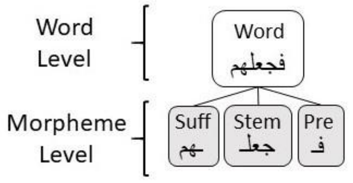


Figure 2. Word Morphemes.

In detail, our dataset consisted of a sequence of words that are characterised by morphemes (prefixes, stems, suffixes) and their corresponding tag sequences (prefix, stem, suffix tags). In other words, each morpheme is tagged with nouns, verbs, prepositions and so on. An example of this is shown in Fig. 3. These detailed morpheme segmentations for words can improve the accuracy of the results derived through the proposed tagger. We therefore expanded our experiments by using words and their corresponding tags.

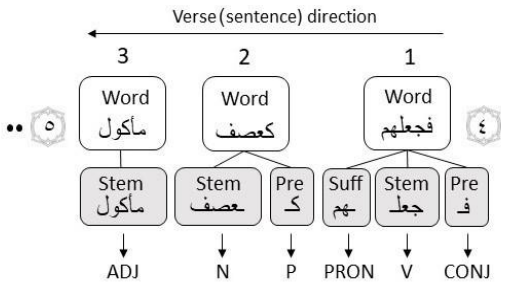


Figure 3. POS Tagging Sequence of Morphemes.

We used stem tags in a tag sequence to label words given that these are the tags available for the meantime. An example of word POS tagging is shown in Fig. 4.

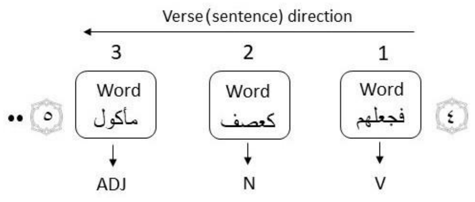


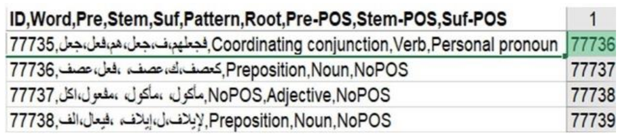
Figure 4. POS Tagging Sequence Of Words.

C. *Tagset*

The proposed tagset for this system has been derived from the QAC [24]. The advantages of this tagset are that it has a good size, which can reveal information about the text; it is intended to be used for the Arabic language by analyzing Quran grammar; and it is effective in manual annotation by crowdsourcing. The original POS tagset contained 44 tags. To tag morphemes, an expanded tagset comprising 87 tags was used because some morphemes were labelled with the composite tags of the original labels. To tag words, a collapsed tagset consisting of 37 tags was used.

D. *Data Preprocessing*

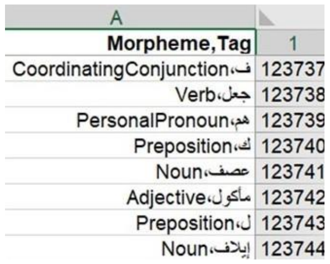
To prepare the data for easy processing by the proposed taggers, some pre-processing steps were taken on the given data file, which was stored as a comma-separated values (CSV) file. The content of the original file consisted of 77,915 words with 14,901 unique words. Each line in data file comprised the ID, word, prefix, stem, suffix, pattern, root, prefix tag, stem tag, and suffix tag. Fig. 5 presents how the data were stored in the original file.



ID,Word,Pre,Stem,Suf,Pattern,Root,Pre-POS,Stem-POS,Suf-POS	1
77735,فجعلهم،ف،جعل،هم،فعل،جعل,Coordinating conjunction,Verb,Personal pronoun	77736
77736,كعصف،ك،عصف، ،فعل،عصف,Preposition,Noun,NoPOS	77737
77737,مأكول، ،مأكول، ،مفعول،اكل,NoPOS,Adjective,NoPOS	77738
77738,لإيلاف،ل،إيلاف، ،فيعال،الف,Preposition,Noun,NoPOS	77739

Figure 5. Quran Data File.

Each line of data consisted of three segments and corresponding tags. We separated each segment (morpheme) with its corresponding tag into a new line. These updates enabled the LSTM tagger to access the file and tag morphemes easily. The final file consisted of 124,002 items with 7,508 unique morphemes. Fig. 6 presents how the data were stored in the updated file.



A	
Morpheme,Tag	1
CoordinatingConjunction،ف	123737
Verb،جعل	123738
PersonalPronoun،هم	123739
Preposition،ك	123740
Noun،عصف	123741
Adjective،مأكول	123742
Preposition،ل	123743
Noun،إيلاف	123744

Figure 6. Updated Quran File

E. *Encoding*

At first, input data for taggers were planned to be without word embeddings. And because NNs cannot understand letters, we needed to find a way to convert words or characters into numbers. Therefore, our two datasets, comprised of morphemes with corresponding tags or words with corresponding tags, had to go through an encoding process. First, we planned to use one-hot encoding (OHE), which is an easy and popular encoding method for a specialised learning algorithm for dealing with numerical data. In the OHE method, every word in the vocabulary is represented by a binary vector with a size equal to the vocabulary size. This method maps each word to a vector with a length equal to the number of unique morphemes, which, in case of converting morphemes, vector length will equal 7,508, and the nth digit is an indicator of the presence of a particular word [25]. While, in case of converting words, vector length will equal 14,901. The vector length indicates the number of neurons needed for the first neural layer. This method performs well with a small data sample, but when working with a large data sample, as in our case, the neural model training time grows exponentially. Therefore, we chose another method, the reversible integer transformation (RIT) algorithm, which was introduced by Affendi [26]. This method converts textual training words or morphemes into binary numerical form. The most important feature is maximum number of neurons needed in the input layer is 64. Because each character in a word is represented using only six bits.

F. *Word Embeddings Technique*

Word embeddings represent a new rising form of distributed semantics representations, where each word in a given corpus or text is represented by a unique numeric vector based on the contexts within which it occurs [27]. Currently, the most popular approach is the Word to Vector (Word2Vec) approach [27], in which the vector representations for words are computed using a shallow MLP model using backpropagation. In the Continuous-Bag-of-Words (CBOW) version of the model a sliding window approach is used to pass the context of the word (for example two words before, and two words after) to the neural network and use any target as the output (for example the word itself). The hidden layer resulting from the training is the semantic vector representation of the word. In this paper, a variation of the CBOW model has been used to create word vectors for Quranic Text Corpus. Word vectors were created by using Gensim [28], a Python library. The motivation is to compare the results obtained of Word2Vec approach to those obtained using LSTM. Fig. 7 explains how to create Word2Vec vectors for morphemes.

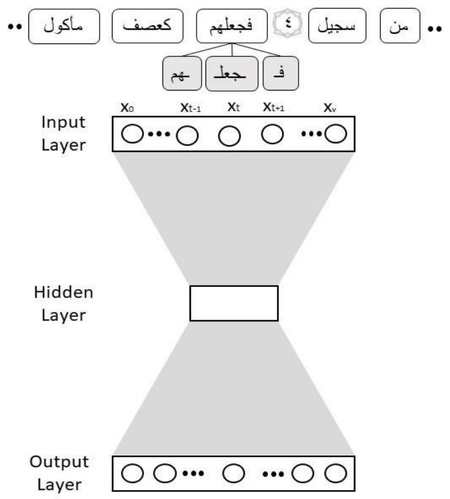


Figure 7. Word2Vec Process.

The first line shows the sequence verses of the Quran, and second line shows the separated morphemes, which are our input for first case. In first case, the input layer size was equal to the number of different morphemes in the vocabulary for training (7,510 words). The hidden layer size was set to the dimensionality of the resulting word vectors (i.e. 300).

G. *Platform*

Several open-source libraries and frameworks are available for advanced deep learning. To develop our NNs, we chose Keras, an NN library written in Python. Keras is a high-level application programming interface (API) that can be run on top of TensorFlow and Theano. We chose it because it has a user-friendly interface and allows for easy and fast prototyping and testing of NNs with short lines of code.

5. Results

The data were divided into a training component (70%) and a testing component (30%). The former was used to train the model, and the latter was used to compute the model's accuracy. When tagging morphemes, the data sample was comprised of 124,002 pairs of morphemes and tags. The training component was comprised of 86,801 pairs, and the testing component included 37,201 pairs. When tagging words, the data sample included 77,190 pairs of words and tags. The training component was comprised of 54,540 pairs, and the testing component consisted of 23,375 pairs.

The training set was fed into the input layer to train the LSTM model. After the model training was completed, the retained data were used to test the model accuracy on datasets that have not been previously examined.

We conducted four experiments. First, we fed the morphemes into LSTM model with using RIT encoding algorithm. The performance of the proposed LSTM POS tagger during testing and training is depicted in Fig. 8, which shows that the model achieved 99.72% accuracy.

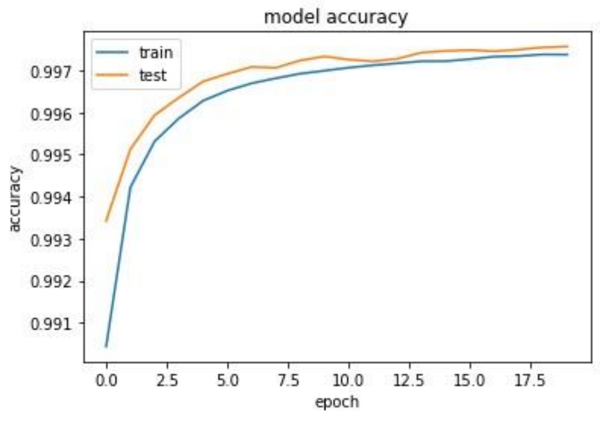


Figure 8. Accuracy of Tagging Morphemes Using the LSTM Model.

As Fig. 8 illustrates, accuracy increased for the training and testing data after the second epoch. Beyond the 10th epoch, no notable variation was observed in either the train or test lines.

Second, we fed words into the LSTM model using only RIT encoding algorithm. The performance of the proposed LSTM POS tagger during testing and training words is shown in Fig. 9. The model achieved 99.18% accuracy.

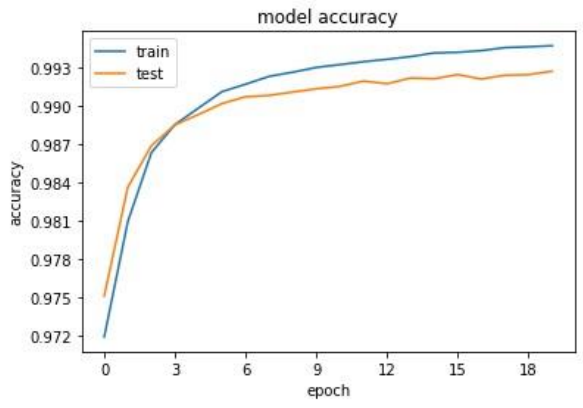


Figure 9. Accuracy of Tagging Words Using the LSTM Model.

Third, we created word vectors for morphemes using Word2vec model, after which we passed the result vectors onto the backpropagation POS tagger to evaluate the impact of using word vectors for predicting POS tags. The performance of the proposed Word2Vec POS tagger during testing and training with the use of word vectors is represented in Fig. 10. The model achieved 99.55% accuracy.

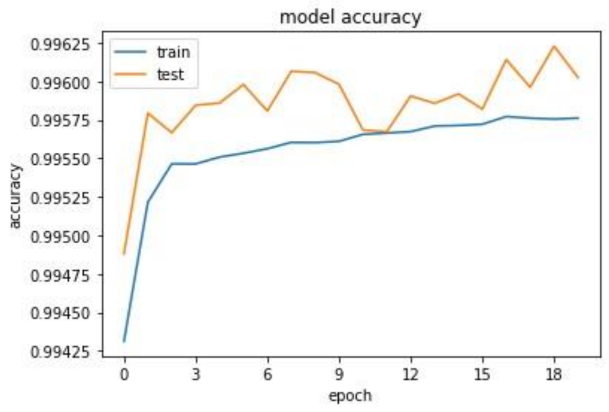


Figure 10. Accuracy of Tagging Morphemes Using the Word2Vec Model

Fourth, we also used Word2Vec for creating words vectors and then incorporated vectors as inputs into Word2Vec POS tagger. The performance of the Word2Vec POS tagger during the testing and training is shown in Fig. 11. As can be seen, the model performed at an accuracy of 97.33%.

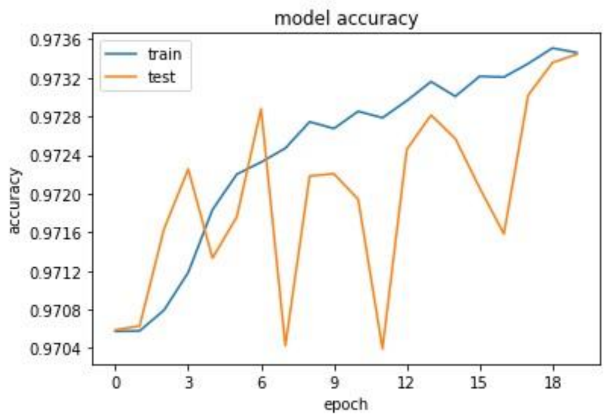


Figure 11. Accuracy of Tagging Words Using the Word2Vec Model.

In the above-mentioned experiments, POS tagging was carried out using the LSTM model for morphemes and words. Also, POS tagging was carried out using the Word2Vec model for morphemes and words with using word vectors to compare word2Vec accuracy with the results obtained using LSTM. Table I compares obtained accuracy results for POS tagging Quran text.

Table I. POS Tagging Accuracy Results For Quran Text

Tagger	Dataset	Accuracy
LSTM	Words	99.18%
LSTM	Morphemes	99.72%
Word2Vec	Words	97.33%
Word2Vec	Morphemes	99.55%

6. Analysis

This section details the comparison of our models results and the comparison of these findings with those of six related Arabic POS taggers. The proposed Word2Vec POS tagger tagged morphemes at an accuracy of 99.55% and words at an accuracy of 97.33% with using word vectors. The proposed LSTM POS tagger tagged morphemes at an accuracy of 99.76% and tagged words at an accuracy of 99.18% without using word vectors. These high results of LSTM POS model confirmed that feature engineering is not required as the power of LSTM-RNNs lies in their ability to learn long-term dependencies [8] and determine context even without embedding. However, the comparison of morpheme and word tagging showed that accuracy for the former is higher by both taggers. This result is ascribed to the fact that decomposing words into parts can provide precise information.

To compare our results with other research, we found two NN experiments on Arabic POS tagging. The first is Abu-Malloh's POS tagger, which was based on a classical NN. This tagger was trained by a standard backpropagation algorithm and designed with a three-layer network with eight hidden neurons [29]. To implement the tagger, Abu-Malloh used a tagset with a size of 18 tags and a total of 16,672 distinct words written in MSA. These words were divided into two distinct sets: a training set and a testing set consisting of 13,337 and 3,335 Arabic words, respectively. The overall accuracy of the developed tagger reached 87.02%.

In the second experiment, Muaidi used the Levenberg Marquardt (LM) algorithm and a tagset with a size of 189 tags [30]. Muaidi's tagger was trained and tested on an Arabic corpus consisting of 24,810 Arabic words with their associated tags, and it was divided into a distinct training set (19,848 Arabic words) and testing set (4,962 Arabic words). The developed Artificial neural networks (ANN) for Muaidi's tagger was successful, reaching an accuracy of 90.21% for the testing dataset.

We also found a study by Abdelkareem, which compared the performance of some POS tagging techniques for Arabic text using the QAC [31], which is the same corpus used in our experiments. These techniques included n-gram, Brill, Hidden Markov Model (HMM), and Trigrams'n'Tags (TnT) taggers. The comparison experiments were conducted on diacritised and undiacritised CA which means CA with and without diacritical marks. The framework was the Natural Language Toolkit (NLTK), which consists of open-source Python modules and enables documentation for research and development in NLP and text analytics. A total of 77,430 words was divided into a training set of 74,859 words and a testing set of 2,571 words. Abdelkareem's experiments were applied on CA for both diacritised and undiacritised data. For the two forms, they studied a 33-tag tagset, and they simplified a new form of a 9-tag tagset. This produced four experimental cases.

The Brill tagger performed the best among all other taggers for undiacritised Arabic for both the 33 tagset (80.9%) and the 9 tagset (83.2%). For diacritised Arabic, the Bigram tagger performed well for both the 33 tagset (80.1%) and the 9 tagset (82.0%). It is noteworthy that each tagger gave better accuracy in tagging undiacritised Arabic than in tagging diacritised Arabic because diacritics are considered additional characters. The comparison of the proposed model with other Arabic POS taggers is a complicated task in which accuracy depends on different factors, such as tagset size and dataset size [30]. A relative comparison of the accuracy results from our proposed tagger with other experiments is shown in Table II.

The results show that our proposed NN models for tagging the QAC produced better results than other statistical techniques used for this purpose. We also noticed that our proposed taggers are better and more effective than the traditional feedforward multilayer perceptron models for tagging MSA words.

TABLE II. COMPARISON OF ACCURACY RESULTS FOR MOST RELATED ARABIC POS TAGGERS. IN THE NOTATIONS, THE SUBSCRIPT D REPRESENTS DIACRITISED ARABIC, U REPRESENTS UNDIACRITISED ARABIC, CA REPRESENTS CLASSICAL ARABIC, MDA REPRESENTS MODERN ARABIC, W REPRESENTS WORD, M REPRESENTS MORPHEME AND WE REPRESENTS WORD EMBEDDING.

Author	Data Type	Corpus size	Tag size	Approach	Accuracy
Our study	CA - U - M	124,002	87	LSTM	99.72%
Our study	CA - U - W	77,190	34	LSTM	99.18%
Our study	CA - U - M - WE	124,002	87	Word2Vec	99.55%
Our study	CA - U - W - WE	77,190	34	Word2Vec	97.33%
Abu Malloh	MDA - U	16,672	18	BPNN	87.02%
Muaidi	MDA - U	24,810	189	LM	90.21%
Abdelkareem	CA - U	77,430	33	Brill	80.90%
Abdelkareem	CA - U	77,430	9	Brill	83.20%
Abdelkareem	CA - D	77,430	33	Bigram	80.10%
Abdelkareem	CA - D	77,430	9	Bigram	82.00%

7. CONCLUSION

A. *Findings*

In this study, we aimed to apply NN technologies for Arabic POS tagging. Lately, NNs have become a well-known and valuable tool for building many NLP applications, such as text classification. Tagging Arabic words is a complex task, and more Arabic POS tagging studies are needed. In this paper, we examined the application of backpropagation and LTSM to identify POS tags of Arabic words. An LSTM-RNN model was used to predict the POS tags of Arabic words and morphemes after being trained using a gold-standard pre-tagged set. Word tagging was carried out at accuracy levels of 99.18% while morphemes tagging achieved 99.72%.

Also, a CBOW model has been applied to predict POS tags for Quranic Text Corpus. The reason for applying word embedding was to compare word embedding accuracy with the results obtained using LSTM.

Tagging by using word embedding was accomplished at accuracy levels of 97.33% and 99.55% for words and morphemes. Decomposing words into parts (morphemes) can provide precise information for POS tagging task. This fact was proved, when POS tagging morphemes was higher than tagging words in both taggers.

We also proved that LSTM POS tagging takes into consideration the context of a word (i.e. neighboring words) in a manner similar to word embeddings. This method showed higher accuracy than POS tagging of words. In general, the high accuracy levels indicated that LSTM is a powerful classifier that may be used to build highly accurate Arabic POS taggers.

B. *Contributions*

- The first attempt to apply LSTM for CA POS tagging using the QAC.
- The use of RIT algorithm that converts textual training words into a binary representation, which reduces the time it takes to train the network.
- Comparing word embedding accuracies with LSTM accuracies for POS tagging QAC words and morphemes.
- Elimination of the need for probabilistic approaches and complex rules to capture the context and dependency, since LSTM models naturally take dependency into consideration and capture the context.
- A highly accurate POS tagger for the Arabic language that compares favourably with all known types of taggers.

C. *Future Work*

Many aspects of this research can be improved further, such as the following:

- Training and testing of different Arabic data samples are needed for NN models.
- Other NN models for NLP, such as convolutional NNs, can be explored.
- Additional experiments for designing NNs can be conducted by tuning NN hyperparameters and trying different numbers of layers and neurons.

References

[1] D. Jurafsky and J. H. Martin, Speech and language processing. Pearson, London, 2014, vol. 3.

[2] A.M. Popescu and O. Etzioni, “Information extraction from unstructured web text,” 2007, vol. 68, no. 02.

[3] R. P. Van Gompel and M. J. Pickering, “Syntactic parsing,” The Oxford Handbook of Psycholinguistics, 2007, pp. 289–307.

[4] D. Khurana, A. Koli, K. Khatter, S. Singh, "Natural language processing: State of the art, current trends and challenges," arXiv preprint arXiv:1708.05148. 2017.

[5] M. M. Lopez and J. Kalita, "Deep learning applied to NLP," arXiv preprint arXiv:1703.03091, 2017.

[6] E. Brill, "A simple rule-based part of speech tagger," Proceedings of the Third Conference on Applied Natural Language Processing. Association for Computational Linguistics, 1992.

[7] X. Li, T. Qin, J. Yang, X. Hu, and T. Liu, "LightRNN: memory and computation-efficient recurrent neural networks," in Advances in Neural Information Processing Systems, 2016, pp. 4385–4393.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, 1997, vol. 9, no. 8, pp. 1735–1780.

[9] M. A. ELAffendi and M. ALTayeb, "The SWAM Arabic morphological tagger: multi-level tagging and diacritization using lexicon driven morphotactics and viterbi," inProceedings on the International Conference on Artificial Intelligence (ICAI), p. 1, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

[10] M. H. Btoush, A. Alarabeyyat, and I. Olab, "Rule based approach for Arabic part of speech tagging and name entity recognition," International Journal of Advanced Computer Science and Applications, 2016, vol. 7, no. 6.

[11] S. Alqrainy, H. M. AlSerhan, and A. Ayesh, "Pattern-based algorithm for part-of-speech tagging Arabic text," in Proceedings of the 2008 International Conference on Computer Engineering & Systems (ICCES). IEEE, 2008, pp. 119–124.

[12] M. Albared, N. Omar, and M. J. Ab Aziz, "Improving Arabic part-of-speech tagging through morphological analysis," in Asian Conference on Intelligent Information and Database Systems. Springer, 2011, pp. 317–326.

[13] A. H. Aliwy, "Combining POS taggers in master-slaves technique for highly inflected languages as Arabic," in 2015 Cognitive Computing and Information Processing (CCIP), 2015 International Conference on. IEEE, 2015, pp. 1-5.

[14] Y. Tlili-Guiassa, "Hybrid method for tagging Arabic text," Journal of Computer Science, 2006, vol. 2, no. 3, pp. 245–248.

[15] S. Khoja, "APT: Arabic part-of-speech tagger," in Proceedings of the Student Workshop at NAACL, 2001, pp. 20–25.

[16] N. Habash and O. Rambow, "Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop," in Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2005, pp. 573– 580.

[17] M. Diab, K. Hacioglu, and D. Jurafsky, "Automatic tagging of Arabic text: from raw text to base phrase chunks," in Proceedings of HLTNAACL 2004: Short Papers. Association for Computational Linguistics, 2004, pp. 149–152.

[18] B. B. Ali and F. Jarray, "Genetic approach for Arabic part of speech tagging," arXiv preprint arXiv:1307.3489, 2013.

[19] K. Darwish, H. Mubarak, A. Abdelali, and M. Eldesouki, "Arabic POS tagging: Don't abandon feature engineering just yet," in Proceedings of the Third Arabic Natural Language Processing Workshop, 2017, pp. 130–137.

[20] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "Part-of-speech tagging with bidirectional long short-term memory recurrent neural network," arXiv preprint arXiv:1510.06168, 2015.

[21] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," arXiv preprint arXiv:1604.05529, 2016.

[22] A. Gulli and S. Pal, Deep learning with Keras. Packt Publishing Ltd; 2017.

[23] K. Dukes, "Statistical parsing by machine learning from a classical Arabic treebank," arXiv preprint arXiv:1510.07193, 2015.

[24] K. Dukes and N. Habash, "Morphological annotation of Quranic Arabic," in LREC, 2010.

[25] D. Harris, and S. Harris. "Digital design and computer architecture," Morgan Kaufmann, 2010.

[26] M. A. ELAffendi and K. S. Alrajhi, "Text encoding for deep learning neural networks: a reversible base 64 (Tetrasexagesimal) integer transformation (RIT64) alternative to one hot encoding with applications to Arabic morphology," in Proceedings of the 6th International Conference on Digital Information, Networking and Wireless Communication (DINWC2018,) Lebanon, 2018.

[27] Y. Goldberg and O. Levy, "Word2Vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method," arXiv preprint arXiv:1402.3722, 2014.

[28] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 2010, pp. 45–50.

[29] R. Abu-Malloh, "Arabic part-of-speech tagger: an approach based on neural network modeling," Master's thesis, AlBalqa Applied University, 2010.

[30] H. Muaidi, "Levenberg-Marquardt learning neural network for part-of-speech tagging of Arabic sentences,' WSEAS Transactions on Computers, 2014, vol. 13, pp. 300–09.

[31] A. M. Alashqar, "A comparative study on Arabic POS tagging using Quran corpus," in Proceedings of the 8th International Conference on Informatics and Systems (INFOS), IEEE, 2012, pp. NLP-29.

Khwlah Alrajhi is a research assistant in the Computer Science department of the College of Computer & Information Sciences, Prince Sultan University, Riyadh. She is also a member of the EIAS Data Science and Blockchain Lab. She received her M.Sc. in Software Engineering from Prince Sultan University, Riyadh, in 2018 and her B.Sc. in Information Technology from King Saud University, Riyadh, in 2013.

Mohammed A ELAffendi is a professor of computer science, in the Department of Computer Science, Prince Sultan University, AIDE to the Rector, Director of CCIS Research, Director of CCIS Graduate Programs, Founder and Director of EIAS Data Science and Blockchain Lab. Current research interests includes Data Science, Intelligent and Cognitive Systems and Machine Learning.