future internet

MDPI

Article

Bidirectional Recurrent Neural Network Approach for Arabic Named Entity Recognition

Mohammed N. A. Ali [1], Guanzheng Tan [1,*] and Aamir Hussain [2]

[1] School of Information Science and Engineering, Central South University, Changsha 410083, China; md.alkhatib@csu.edu.cn

[2] Department of Computer Science, Muhammad Nawaz Shareef University of Agriculture, Multan 60000, Pakistan; aamir.hussain@mnsuam.edu.pk

* Correspondence: tgz@csu.edu.cn

Received: 28 October 2018; Accepted: 10 December 2018; Published: 13 December 2018

Abstract: Recurrent neural network (RNN) has achieved remarkable success in sequence labeling tasks with memory requirement. RNN can remember previous information of a sequence and can thus be used to solve natural language processing (NLP) tasks. Named entity recognition (NER) is a common task of NLP and can be considered a classification problem. We propose a bidirectional long short-term memory (LSTM) model for this entity recognition task of the Arabic text. The LSTM network can process sequences and relate to each part of it, which makes it useful for the NER task. Moreover, we use pre-trained word embedding to train the inputs that are fed into the LSTM network. The proposed model is evaluated on a popular dataset called "ANERcorp." Experimental results show that the model with word embedding achieves a high F-score measure of approximately 88.01%.

Keywords: Arabic named entity recognition; bidirectional recurrent neural network; GRU; LSTM; natural language processing; word embedding

1. Introduction

The Named entity recognition (NER) is important in natural language processing (NLP) tasks used to detect named entities (NEs) in texts and classify them into predefined categories, such as location, person, time, date, and organization [1]. NER is a crucial preprocessing phase in various NLP applications to improve the overall performance. It extracts valuable information from raw data and simplifies downstream tasks, such as text clustering, information retrieval, translation, and question answering [2].

In recent years, Arabic NER (ANER) has become a challenging task and is receiving increasing attention from current researchers due to the limited availability of annotated datasets [3]. Arabic is a Semitic and the standard language spoken in the Arab world. The language is used in the Middle East, the Horn of Africa and North Africa, and it is used in the United Nations as one of the five official languages. In the Arab world, around 360 million people speak Arabic in more than 25 countries [4].

The Arabic NLP has drawn attention in recent years. Several NLP tasks are tricky, such as NER [5], particularly language important features, such as high morphological uncertainty of meaning, writing style, doubtfulness to the meaning of common words/proper noun, and absence of capitalization [2].

The ANER systems are based on either one of two methods: one is based on handcrafted rules, particularly the NERA2.0 system [6], and the other relies on statistical learning such as in [7]. Each method, nonetheless, has its pros and cons. NER systems designed on rule-based primarily depend on manually crafted grammatical rules learned from linguists. Therefore, the maintenance of these systems is time-consuming and laborious, especially when the knowledge and background of the linguists are poor. By contrast, systems based on machine learning (ML) obtain patterns

related to the NER task from the training set of samples automatically, thereby not requiring in-depth language-specific knowledge. These ML-based systems are superior to rule-based systems because they are adjustable and easy to update with minimum cost and time, provided a sufficient corpus.

In recent years, neural networks have drawn much attention with various models being proposed. Researchers have combined different semi-supervised learning and deep neural networks (DNNs) to find an optimum solution to the NER task and other chunking tasks [8]. Contrary to ordinary ML methods, deep learning can concurrently learn representations, categorize patterns, and considerably reduce the difficulty in NER tasks. Moreover, current deep learning models generally utilize word embeddings, which allow them to learn similar representations for semantically similar words. However, out-of-vocabulary (OOV) words, which are words that do not have any corresponding representation in the word embedding model, are difficult to handle especially for the Arabic language because of the limitation in resources. These words are also set randomly to a specific value. Therefore, we fully utilize the character representations of a token to label those OOV words. We also introduce the embedding attention layer that works as a gating mechanism that allows the model to dynamically learn what features are important.

ANER is considered a sequence labeling task. Recurrent neural network (RNN) is a natural choice to tackle problems with sequential structure, such as NER, due to their ability to memorize previous values and correlate to other parts of a sequence. RNN surpasses other methods in terms of NER performance and other sequence labeling problems for many languages.

To the best of our knowledge, no work has been done to address the ANER problem using RNN techniques, and most existing works are based on feature engineering and statistical methods. In this work, we propose a new method that has not been investigated in much detail for the Arabic language to date. However, the method has been examined and applied widely in other domains and languages, such as English, and has shown outstanding results.

The model computes the harmonic mean F-score measure for tokens in the dataset. The proposed system has improvements that boost the recognition efficiency and accuracy. The main contribution of our work are listed as follows:

- An ANER system based on bidirectional long short-term memory (Bi-LSTM)/gated recurrent units (GRUs) is proposed by considering the NER problem as a classification problem and without using any manual feature engineering.
- Character embedding is used in addition to word embedding to enhance the model prediction.
- An embedding attention layer that combines word and character embeddings enable the model to create a good word representation.

Our work differs from existing ones for the Arabic language because the proposed model shifts from traditional ML algorithms to neural network algorithms. Moreover, the Bi-LSTM unit uses character and word embeddings as input. A well-recognized dataset called "ANERcorp" is used to evaluate the performance of the proposed system in comparison with other common state of the art systems.

The rest of the paper is organized as follows. Section 2 reviews the related work briefly. Section 3 presents the proposed approach for NER in detail. Section 4 describes the experimental settings. Section 5 discusses the results. Section 6 presents the discussion. Finally, Section 7 elaborates the conclusion.

2. Related Work

As per recent systematic review for NER conducted by [9], NER approaches can be categorized into three mains categories rule-based, learning-based and hybrid approaches. Furthermore, another comprehensive survey conducted by W Etaiwi [2], classify the statistical methods of Arabic NER into six main approaches: CRF, NB, HMM, ME, SVM and Neural network. Going more in-depth into deep learning, a survey on ANLP using deep learning techniques by [10]. The author reviewed many kinds

of literature on various ANLP tasks and concluded that yet a considerable gap exists in the Arabic NLP compared to that in English NLP.

Basic ML approaches rely on feature engineering, external gazetteers, and chunks. They can achieve excellent accuracy and performance but require a dedicated expert in the domain of knowledge and are time-consuming. Therefore, scholars have begun to consider the artificial neural network (ANN) and DNN methods. These approaches reduce the dependency on feature engineering and are thus less laborious and time-consuming. These methods have been adopted to many NLP problems successfully such as in [11].

The authors in [12] proposed an RNN model for the Chinese Bio-NER that detects two types of predefined annotations, namely, subject and lesion, which are two main parts of symptom entities. A real-world Chinese clinical dataset that includes 12,498 records was used. A priori word information (POS tags) was added for improving the final performance. The final F-scores for the subject and lesion detection reach 90.36% and 90.48% respectively.

In [13], the authors proposed a hybrid NER system for the Russian language that uses a Bi-LSTM-CRF system for various kinds of DNN models experimented starting from vanilla (Bi-LSTM). The models were further supplemented with CRF along with highway networks and added with word embedding. He evaluated all the proposed systems across three datasets: Gareev's, Person-1000, and FactRuEval 2016. He concluded that the quality of predictions is considerably increased with the extension of Bi-LSTM model with CRF and can be further improved by setting the word input to preprocessing with exterior word embedding.

The authors in [14] proposed an improved NER system using deep learning module for Chinese text. Without using any manual feature engineering, the system can detect the word features automatically. He used word embedding with a Bi-LSTM obtained from the outputs of NER to model the substance within a sentence. Along with additional features to the model, the experimental results show that the model achieves an F-score of 0.9247 when trained on a large corpus on Bi-LSTM with word embedding.

The authors in [15] investigated a deep learning method to recognize NEs from Chinese clinical text using the minimal feature engineering approach. Two DNN models were developed: one is for generating the word embedding, and the other is for the main NER task. They evaluated the system on two Chinese clinical datasets: one is an annotated corpus that contains 400 randomly selected admission notes, and the other is unlabeled admission notes that include 36,828 notes, both of which were collected from the EHR database of Peking Union Medical College Hospital in China. The model results indicate that the proposed approach with DNN outperforms the CRF and achieves a high F1-score of 0.9280.

In addition to NER for other languages, such as English, Russian, Chinese, and Hindi, The ANER is an attractive and challenging task and requires a recognition task due to the peculiar and unique characteristics of the Arabic language.

The authors in [16] adopted a new attempt for ANER using ANNs. They used ANN techniques and evaluated the performance of their model with decision trees. Their system consists of three main phases: preprocessing of the data, conversion of Arabic letters to Roman alphabets, and classification of the collected data using a neural network. The relationship between the accuracy of the system and the corpus size was also assessed. The results showed that high accuracy on the same dataset ("ANERCorp") is achieved when an ANN method is adapted to the NER system than when complemented with the decision trees. The experimental results also showed that the accuracy of the system increases proportionally with the enlargement in the size of the corpus.

3. Approach

A prototype of bidirectional RNN (B-RNN) is explicitly built on long short-term memory (LSTM)/GRUs. For the entire work, we start with a brief overview of RNN, LSTM, and GRU. Then, we present the bidirectional architecture for the ANER task.

3.1. Recurrent Neural Network (RNN)

RNN is a type of ANN in which the connections between units produce a directed graph along a sequence. This architecture enables the network to demonstrate dynamic temporal behavior for a time sequence. Contrary to feedforward neural networks (FFNNs), RNNs can process sequences of inputs by using their internal state (memory), which makes them appropriate to many NLP tasks [17–19]. LSTM is a type of RNN architecture that is the best among all other existing architectures [20]. RNNs are deep learning systems that stem from the modifications of FFNN with recurrent connections. In a typical neural network, the output of a neuron at time t is calculated as

$$y_i^t = \sigma(W_i x_t + b_i), \tag{1}$$

where W_i is the weight matrix, and b_i is a bias term. In RNN, the calculation of the activation function is modified because the output of the neuron at the time '$t-1$' is fed back into the neuron. Then, the new activation function can be computed as

$$y_i^t = \sigma\Big(W_i x_t + U_i y_i^{t-1} + b_i\Big). \tag{2}$$

RNNs can remember previous information of a sequence by using the output of the earlier node as recurrent connections while presenting output depending on the former states. This characteristic makes the network useful for the sequence labeling task. The backpropagating error can regularly "strike up" or blast, which yields infeasible convergence. It can also vanish to render the capability of the network to learn long-term dependencies, which are difficult to learn through gradient descent.

3.2. Long Short-Term Memory (LSTM)

LSTM networks are a class of RNNs that are designed efficiently to study long-term dependencies and avert the fading gradient issue. LSTM prevents back-propagating errors from vanishing or exploding. To realize this task, LSTM holds an inner condition that symbolizes the memory cell of the LSTM neuron. The inner condition state is usually augmented by recurrent gates that control the movement of the information over the cell state. These gates are updated and calculated as follows:

$$i_t = \tanh(W_{xi} x_t + W_{hi} h_{t-1}), \tag{3}$$

$$f_t = \sigma\Big(W_{xf} x_t + W_{hf} h_{t-1}\Big), \tag{4}$$

$$o_t = \sigma(W_{xo} x_t + W_{ho} h_{t-1}), \tag{5}$$

where i_t, f_t, and o_t represent input, forget, and output gates, respectively. The first two gates decide the input of the last output and the present input in the new cell condition c_t. The last gate takes charge of the quantity of the cell state c_t, which is exposed as the output. The new c_t and h_t can be computed as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c), \tag{6}$$

$$h_t = o_t \odot \tanh(c_t). \tag{7}$$

The cell state keeps relevant information from the last time steps. The cell state can be updated via the input and forget gates in an additive manner only. This procedure can be regarded as permitting the error to stream back over the cell state unchecked until it gets propagated back to the time step in which the related information is added. This mechanism enables LSTM to study long-term reliance.

3.3. Gated Recurrent Units (GRUs)

GRU, which was first presented in 2014, is a mechanism that uses gates in RNNs. Although GRU is modern, it can be presented as a simplified version of the LSTM. GRU uses different mechanisms

for generating the update and updating the cell state. The activation function of the candidates h_t is calculated on the basis of the present input and the last cell state as follows:

$$\hbar_t = \sigma(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1})), \tag{8}$$

where r_t is the reset gate that has the same functional form as that of an update gate, and all of its weights are the same size. Reset gate is multiplied by the previous hidden state value and controls the usage of last cell state while computing the activation input. It can reset the hidden value. The computation of the reset gate is based on the last activation cell h_{t-1} and the present candidate activation.

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1}) \tag{9}$$

The current cell state or activation is a linear combination of the previous cell and candidate activations.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h_t, \tag{10}$$

where Z_t is the update gate that balances the combined amount of the previous and new candidate hidden values to obtain the new hidden value.

$$Z_t = (W_{hz}h_{t-1} + W_{xz}x_t) \tag{11}$$

3.4. Proposed Model

In this section, we discuss our bidirectional model for recognizing NEs. Instead of a feedforward network, Bi-LSTM/GRU network is adapted to obtain bidirectional word arrangement for excellent foresight.

Problem Definition: Given a document D containing a sequence of sentences (s_1, s_2, . . . , s_m) for the input sentence X = (x_1, x_2, · · ·, x_n), Z is identified as the output matrix n × k, where k is the number of labels that represent entities. Z_{ij} refers to the score when the ith token (x_i) in the sentence tag_j. Y = (y_1, y_2, y_3, · · ·, y_n) is the predicted sequence.

First, the word and character embeddings are obtained for the input sentence. Then, we adopt an embedding attention layer that combines the two features to obtain the best word representation. The embedded feature representation is fed into the encoder layer for processing. Finally, the result is obtained from the output layer. Xi refers to the input word, whereas Yi is the predicted tag for the i'th token in the sentence, where $1 \leq i \leq t$. The embedding layer consists of the word vector mapping from tokens to dense n-dimensional vector representations. The model is explained in detail in the following sections. Figure 1 shows the B-RNN setup with the LSTM/GRU network, and the main components are described below.

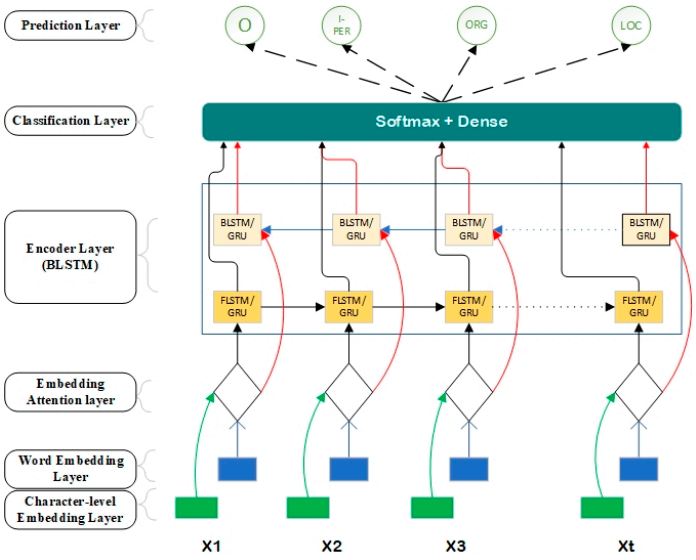

Figure 1. Main architecture of the network. Word embedding is provided to a Bi-LSTM/GRU. The forward unit represents the word X_t and its left context, whereas the backward unit represents the word X_t and its right context. Concatenating the two vectors yields a representation of the word X_t and its context, and this embedded representation is fed to the classification layer.

A. Embedding Layer

The work introduced by [21] for word distributed representation has substituted the traditional bag-of-words encoding technique and accomplishes excellent results on many NLP tasks. In distributed embeddings, the model is generalizable because each word gets a map to space. As a result, semantically approximated words can have similar vector representations. However, using word embedding alone as the smallest feature representation unit can result in losing some accurate information. For languages with rich morphology, such as Arabic, we need to capture all morphological and orthographic information. As word embedding encodes semantic and syntactic word relationships, character embedding carries important morphological and shape information. Inspired by this integration as in [22], we acquire the sequence representations from character and word levels.

- Character Embedding Layer

Character sequence representations are helpful for morphologically rich languages and for dealing with the OOV problem for tasks, such as POS tagging and language modeling [23] or dependency parsing [24]. The authors in [25] proposed CharCNN, which is a character-aware neural language model that learns character word representations using CNNs. We follow the same technique for generating the character embedding representation. Details about the implementation can be found in [25].

- Word Embedding

Word embedding refers to the representation of words as vectors in a continuous space, thereby capturing many syntactic and semantic relations among them. We treat the embeddings as fixed constants as this consideration performs better than treating them as learnable model parameters [26]. We adopt pre-trained word embedding, AraVec 2.0 [27], to obtain the fixed word embedding of each token.

B. Embedding Attention Layer

Word embedding treats words as the smallest unit and disregards any morphological resemblances between various words, thereby leading to the OOV problem. By contrast, character embedding can operate over individual characters in each word and can, therefore, be useful for

handling OOV. However, research on character embedding is still in the initial phase, and systems that work solely on characters are not superior to those based on words on most tasks [28]. Therefore, character and word embeddings can be integrated to fully utilize their advantages. However, we adopt an embedding attention layer that works as a gate mechanism, can learn similar representations, and allows the model to determine how to consolidate the information for each word. After getting the character feature of each word, we calculate the attention matrix through a two-layer perceptron and combine the two levels of features by a weighted sum as follows:

$$z = \sigma(U_a \tanh(V_a x + W_a m)), \tag{12}$$

$$\widetilde{x} = z \cdot x + (1 - z) \cdot m \tag{13}$$

where U_a, V_a, and W_a are the weight matrices for computing the attention matrix z and $\sigma()$ is the sigmoid logistic function with values between 0 and 1. x and m are sequence representations of word and character embeddings, respectively. The dimensions of vector z are the same as those of x or m and act as the weight between the two vectors. Accordingly, the model can dynamically determine the amount of information to use from each of the embeddings (character or word).

C. Bidirectional Recurrent Neural Networks (B-RNNs)

Despite its simplicity, B-RNN is a powerful way to improve the neural network ability to learn, especially for NLP problems. The need for B-RNN helps us consider it in the context of NLP tasks, especially NER in which we try to extract NEs, such as people, location, organization, date, and time.

We consider the three sentences below:

سبأ مملكة عربية يمنية قديمة "Saba is an ancient Yemeni kingdom."

سبأ المتحدة للصناعات الكيماوية المحدودة "Saba United Chemical Industries Ltd."

سبأ بنت مبخوت الشهراني زعيم أكبر القبائل العربية "Saba, the daughter of Mbkhout Shahrani leader of the largest Arab tribes."

The word **"- سبأSaba"** in the three sentences will be tagged with something else in each sentence. This word represents a location in the first sentence, a company name in the second sentence, and a proper noun in the third sentence. The problem is that the word **"سبأ"** appears at the beginning of each sentence. This condition indicates that the RNN network will not detect any other word in the sentence before it has the opportunity to make a prediction on the word **"سبأ"**, resulting in a possible incorrect prediction. B-RNN solves this problem by traversing the sequence in both directions. The backward RNN will calculate $\overleftarrow{h_T}$ in reverse direction, starting from h_T and then going backward until h_1 for ease of prediction of the right label for the given NE. The idea of going backward helps accurately mark the NEs for the word **"سبأ"** that appears at the beginning of the sentence.

The bidirectional unit contains two LSTM/GRU series: one is propagating in the forward direction, and the other is propagating toward the back direction. We concatenate the outcome from the two series to establish a joint representation of the word and its context.

$$h_t = [\overrightarrow{h_T}, \overleftarrow{h_1}], \tag{14}$$

where $\overrightarrow{h_T}$ is the last word in the forward chain and $\overleftarrow{h_1}$ is the last word in the backward chain.

D. Prediction Layer

The joined vectors from the B-LSTM/GRU network are fed into a classification layer with a feedforward neuron. Furthermore, a SoftMax function is used to normalize the output, and it is given by the following equation:

$$output = softmax\ (h_t). \tag{15}$$

For each tag type j, the probability of similar outputs can be calculated as follows:

$$P(l_t = j|u_t) = \frac{\exp(u_t W_j)}{\sum_{k=1}^{K} \exp(u_t W_k)}, \quad (16)$$

where l_t and u_t are the tags or labels and the concatenated vector for each time step t. The highest possible tag at each word position is chosen. The entire network is trained via backpropagation. The embedding vectors are updated on the basis of back-propagating errors as well.

4. Experiment

A series of extensive experimentation was conducted to validate the methodology. The datasets used and the experimental setup were explained thoroughly.

4.1. Datasets

To train and test our ANER system, we evaluated the system with "ANERCorp," which is a dataset created by Benajiba from several online resources [29]. The ANERCorp dataset is a manually annotated corpus that is freely available for research purposes. Two corpora were used: training and testing. One person annotated the corpus to guarantee the coherence of the annotation, and it had 4901 sentences with 150,286 tokens. Each line contained a single token for easy parsing. Each word in this dataset was tagged with one of the following: person, location, company, and others. ANERcorp was annotated into eight classes: B-PERS, beginning of the person's name; I-PERS, inside of the person's name; B-LOC, beginning of the location's name; B-ORG, beginning of the organization's name; I-ORG, inside of the organization's name; B-MISC, beginning of the miscellaneous word; I-MISC, inside of the miscellaneous word; O, the word that is not an NE but refers to other NEs. The dataset distribution was as follows: 39% for person, 30.4% for location, 20.6% for organization, and the remaining 10% for miscellaneous.

4.2. Baseline

Many approaches tackle the ANER problem. We selected some of the works that have been carried out previously and compared them with our work by using the same dataset and evaluation metrics. The following works were selected as baselines:

- The work using a minimally supervised approach for ANER proposed in [30]
- The approach using CRFs by Yassine Benajiba in [31]
- The ANN approach using a neural network technique by Naji and Nazlia in [16]

4.3. Setting

An NVIDIA GeForce GTX1080Ti (12 GB and Intel i7-6800K 3.4GHZx12 Processor with 32 GB RAM) was used to train the model. It was built on Ubuntu and implemented in the Keras environment. For each token, the model was trained to predict either one of the eight appropriate labels described in Section 4.1. The embedding dimension was fixed to 100, and the size of the hidden state was aligned to 200. The combination of forward and backward LSTM provided a dimension of 400 Tanh, which was used as the hidden activation function. Its output was fed into a Softmax output layer to produce probabilities for each of the eight tags. Categorical cross-entropy was used as the objective function, and L2-regularization component was added to the cost function for output tuning. For the over-fitting problem, 50% dropout was used as an additional measure to control the inputs to the LSTM network and the Softmax layer. AdaGrad was used to optimize the network cost. The batch sizes were set to 128. A total of 30 epochs were used to train each network. We set the maximum sequence length to 100 to ensure the same length of all the sequences. Sequence greater than this length would be truncated and sequence less than this length would be padded with zeros to obtain the same length.

4.4. Evaluation

The efficiency of the system was assessed by the general measures, namely, precision (P), recall (R), and F-measure score (F), because they are standard for NER.

$$P = \frac{X}{Y}, \tag{17}$$

$$R = \frac{X}{Z}, \tag{18}$$

$$F - measure(F) = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{19}$$

where X is the total number of correctly extracted entities, Y is the entire number of recognized entities, and Z is the total number of correct entities.

5. Results

We run our experiments on the ANERCorp dataset described above. To evaluate the performance of our Bi-LSTM/GRU model, we run each model 5 times and take the average of each score value. The experimental results are summarized in Table 1. RNN can handle the sequence labeling problem efficiently without the need for additional information, such as Chunks or Gazetteers, which are essential especially for the NER task.

Table 1. Results obtained by our model with different architectures.

Model	*P*	*R*	*F1*
LSTM	80.49	79.71	80.07
GRU	75.46	76.86	76.13
BLSTM	86.85	86.01	86.42
BGRU	85.70	83.96	84.81
BLSTM + char	88.49	87.57	88.01
BGRU + char	87.73	86.51	87.12

The experimental results show that our proposed model with embedding attention layer performs considerably better than the other neural NE recognition and previous baseline systems. The best F-scores of our model are 88.01% and 87.12% for BLSTM and BGRU, respectively.

6. Discussion

6.1. Comparison of Different Architectures

To evaluate the performance of our Bi-LSTM/GRU model and show the effect of different architectures (i.e., the bidirectionality and the impact of character embedding), we first run our experiment on the unidirectional LSTM/GRU. Then, we run the Bi-LSTM/GRU. Finally, we add the character embedding. As observed from the results above, the model achieves 80.07% and 76.13% for LSTM and GRU, respectively. The model obtains slight improvement in bidirectionality and achieves the best results of 88.01% and 87.12% for BLSTM and BGRU, respectively, when the character embedding is added to it. Notably, the performance of B-LSTM and B-GRU is nearly similar and slightly better than that of LSTM. Moreover, RNN with word embedding can perform effectively in the ANER task without the need for manual feature engineering.

6.2. Comparison with Other Models

The performance of the proposed model is compared with those of other existing state of the arts. Table 2 shows the comparison results obtained by our model against those of other systems. Our proposed model, B-LSTM/GRU, achieves the best result regarding the metrics used for evaluation

on the "ANERCorp" dataset. This performance emphasizes that using RNN leads to excellent performance and improves the labeling task. Specifically, the efficiency results obtained on the NER task are comparable to those of previous methods.

Table 2. Comparison of F-score performance measure of the proposed models concerning the baseline systems on ANERCorp.

Model	*P*%	*R*%	*F*%
Maha Althobaiti [30]	84.94	52.68	63.22
Benajiba (ANERsys 2.0) [31]	65.33	58.60	61.73
Nazlia Omar [16]	65.03	62.33	57.47
F Dahan (HMM) [7]	77	73	75.2
OUR B-LSTM	88.49	87.57	**88.01**
OUR B-GRU	87.73	86.51	**87.12**

6.3. Effect of Character Embedding

Apart from the effect of word embedding, which is a powerful tool to learn word representation and can perform excellently on various NLP tasks, character embedding enhances the performance of the model by handling the OOV problem and providing an accurate representation for words that do not have corresponding word embeddings.

7. Conclusions

Arabic is a challenging language because of its high morphological ambiguity, writing style, and absence of capitalization. Thus, any NLP task for this language requires a large number of feature engineering and preprocessing steps. In this study, we experiment with a B-RNN with LSTM/GRU for the ANER task. Without using any feature engineering or additional preprocessing, we tackle the problem of NER for the Arabic text. We find that deep learning-based approaches, especially LSTM network, are useful for identifying Arabic NEs and can efficiently outperform many other methods based on manually engineered features or rule systems. The incorporation of pre-trained word embedding enables the system to obtain considerable improvements in the recognition task and achieve excellent results in F-score measures, where we achieved a high F-score measure of approximately 88.01% and 87.12% for Bi-LSTM and Bi-GRU respectively.

Author Contributions: M.N.A.A. wrote the paper with support from A.H. A.H carried out the experiments with the help of M.N.A.A. G.T. supervised the whole project. All the authors revised the manuscript.

Funding: This work was partially funded by the National Natural Science Foundation of China (Project No. 61403422).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shaalan, K.; Oudah, M. A hybrid approach to Arabic named entity recognition. *J. Inf. Sci.* **2014**, *40*, 67–87. [CrossRef]
2. Etaiwi, W.; Awajan, A.; Suleiman, D. Statistical Arabic Name Entity Recognition Approaches: A Survey. *Procedia Comput. Sci.* **2017**, *113*, 57–64. [CrossRef]
3. Zirikly, A.; Diab, M. Named entity recognition for arabic social media. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, Berlin, Germany, 12 August 2016; pp. 176–185.
4. Nydell, M.K. *Understanding Arabs: A Guide for Modern Times*; Intercultural Press: Boston, MA, USA, 2018.
5. Shaalan, K.; Raza, H. NERA: Named entity recognition for Arabic. *J. Assoc. Inf. Sci. Technol.* **2009**, *60*, 1652–1663. [CrossRef]
6. Oudah, M.; Shaalan, K. NERA 2.0: Improving coverage and performance of rule-based named entity recognition for Arabic. *Nat. Lang. Eng.* **2017**, *23*, 441–472. [CrossRef]

7. Dahan, F.; Touir, A.; Mathkour, H. First Order Hidden Markov Model for Automatic Arabic Name Entity Recognition. *Int. J. Comput. Appl.* **2015**, *123*, 37–40. [CrossRef]
8. Tomas, M. Statistical Language Models Based on Neural Networks. Available online: http://www.fit.vutbr.cz/~{}imikolov/rnnlm/google.pdf (accessed on 9 December 2018).
9. Goyal, A.; Gupta, V.; Kumar, M. Recent Named Entity Recognition and Classification techniques: A systematic review. *Comput. Sci. Rev.* **2018**, *29*, 21–43. [CrossRef]
10. Al-Ayyoub, M.; Nuseir, A.; Alsmearat, K.; Jararweh, Y.; Gupta, B. Deep learning for Arabic NLP: A survey. *J. Comput. Sci.* **2018**, *26*, 522–531. [CrossRef]
11. Awad, D.; Sabty, C.; Elmahdy, M.; Abdennadher, S. Arabic Name Entity Recognition Using Deep Learning. In Proceedings of the International Conference on Statistical Language and Speech Processing, Mons, Belgium, 15–16 October 2018; pp. 105–116.
12. Li, J.; Zhao, S.; Yang, J.; Huang, Z.; Liu, B.; Chen, S.H.; Pan, H.; Wang, Q. WCP-RNN: A novel RNN-based approach for Bio-NER in Chinese EMRs: Paper ID: FC_17_25. *J. Supercomput.* **2018**. [CrossRef]
13. Le, T.A.; Arkhipov, M.Y.; Burtsev, M.S. Application of a hybrid Bi-LSTM-CRF Model to the task of Russian named entity recognition. *Commun. Comput. Inf. Sci.* **2018**, *789*, 91–103.
14. Ouyang, L.; Tian, Y.; Tang, H.; Zhang, B. Chinese Named Entity Recognition Based on B-LSTM Neural Network with Additional Features. In Proceedings of the International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, Guangzhou, China, 12–15 December 2017; pp. 269–279.
15. Wu, Y.; Jiang, M.; Lei, J.; Xu, H. Named Entity Recognition in Chinese Clinical Text Using Deep Neural Network. *Stud. Health Technol. Inform.* **2015**, *216*, 624–628. [PubMed]
16. Mohammed, N.F.; Omar, N. Arabic named entity recognition using artificial neural network. *J. Comput. Sci.* **2012**, *8*, 1285–1293.
17. Yousfi, S.; Berrani, S.-A.; Garcia, C. Contribution of recurrent connectionist language models in improving LSTM-based Arabic text recognition in videos. *Pattern Recognit.* **2017**, *64*, 245–254. [CrossRef]
18. Baly, R.; Hajj, H.; Habash, N.; Shaban, K.B.; El-Hajj, W. A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in arabic. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2017**, *16*, 23. [CrossRef]
19. Chherawala, Y.; Roy, P.P.; Cheriet, M. Feature set evaluation for offline handwriting recognition systems: Application to the recurrent neural network model. *IEEE Trans. Cybern.* **2016**, *46*, 2825–2836. [CrossRef] [PubMed]
20. Jozefowicz, R.; Zaremba, W.; Sutskever, I. An empirical exploration of recurrent network architectures. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2342–2350.
21. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
22. Li, F.; Zhang, M.; Fu, G.; Ji, D. A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinform.* **2017**, *18*, 198. [CrossRef] [PubMed]
23. Ling, W.; Luís, T.; Marujo, L.; Astudillo, R.F.; Amir, S.; Dyer, C.; Black, A.W.; Trancoso, I. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv* **2015**, arXiv:1508.02096.
24. Ballesteros, M.; Dyer, C.; Smith, N.A. Improved transition-based parsing by modeling characters instead of words with LSTMs. *arXiv* **2015**, arXiv:1508.00657.
25. Kim, Y.; Jernite, Y.; Sontag, D.; Rush, A.M. Character-Aware Neural Language Models. In Proceedings of the AAAI, Phoenix, AZ, USA, 12–17 February 2016; pp. 2741–2749.
26. Cocos, A.; Fiks, A.G.; Masino, A.J. Deep learning for pharmacovigilance: Recurrent neural network architectures for labeling adverse drug reactions in Twitter posts. *J. Am. Med. Inform. Assoc.* **2017**, *24*, 813–821. [CrossRef] [PubMed]
27. Soliman, A.B.; Eissa, K.; El-Beltagy, S.R. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Comput. Sci.* **2017**, *117*, 256–265. [CrossRef]
28. Rei, M.; Crichton, G.K.O.; Pyysalo, S. Attending to characters in neural sequence labeling models. *arXiv* **2016**, arXiv:1611.04361.

29. Benajiba, Y.; Rosso, P.; Benedíruiz, J.M. Anersys: An arabic named entity recognition system based on maximum entropy. In Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, Mexico, 18–24 February 2007.
30. Kruschwitz, U.; Poesio, M. Combining minimally-supervised methods for arabic named entity recognition. *Trans. Assoc. Comput. Linguist.* **2015**, *3*, 243–255.
31. Benajiba, Y.; Rosso, P. Arabic Named Entity Recognition using Conditional Random Fields. *Proc. Work. HLT NLP Arab. World LREC* **2008**, *8*, 143–153.

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).