Character-Aware Neural Networks for Arabic Named Entity Recognition for Social Media

Mourad Gridach
High Institute of Technology
Ibn Zohr University - Agadir
m.gridach@uiz.ac.ma

Abstract

Named Entity Recognition (NER) is the task of classifying or labelling atomic elements in the text into categories such as Person, Location or Organisation. For Arabic language, recognizing named entities is a challenging task because of the complexity and the unique characteristics of this language. In addition, most of the previous work focuses on Modern Standard Arabic (MSA), however, recognizing named entities in social media is becoming more interesting these days. Dialectal Arabic (DA) and MSA are both used in social media, which is deemed as another challenging task. Most state-of-the-art Arabic NER systems count heavily on hand-crafted engineering features and lexicons which is time consuming. In this paper, we introduce a novel neural network architecture which benefits both from character- and word-level representations automatically, by using combination of bidirectional Long Short-Term Memory (LSTM) and Conditional Random Field (CRF), eliminating the need for most feature engineering. Moreover, our model relies on unsupervised word representations learned from unannotated corpora. Experimental results demonstrate that our model achieves state-of-the-art performance on publicly available benchmark for Arabic NER for social media and surpassing the previous system by a large margin.

1 Introduction

Named Entity Recognition (NER) is the task of tagging, labeling or identifying atomic items in the text with predefined set of named entity categories such as Person, Location, Organization, etc. from large corpora (Nadeau and Sekine, 2007) . Recently, named entity recognition has gained an important role in Natural Language Processing (NLP) because it can have an impact on other NLP applications. In Question Answering (QA), Ferrndez (2007) showed that using NER system in their QA model improves its performance and questions contain 85% Named Entities. Toda (2005) showed that adding NER system in their Text Clustering system enhanced its performance and allowed them to outperform the existing state-of-the-art system. Babych (2003) clarified that using named entity recognition in Machine Translation (MT) can help the system to improve the translation task. Thompson (1997) prompted that using NER improve Information Retrieval (IR) performance. In addition, NER could be used in various NLP systems to improve their performance such as semantic parsers, part of speech taggers, document and news searching.

Current state-of-the-art systems perform very well on recognizing Arabic named entities such as Person, Location, or Organization (Shaalan and Oudah, 2014) for MSA texts. However, there is relatively less interest on recognizing named entities in social media like Twitter, movies, TV shows. In this paper, we focus on recognizing Arabic named entities in Twitter. Lately, Arabic language was the fastest growing language on Twitter, and in 2012, it was the 6th most used language on Twitter (Semiocast, 2012). This rapid increase in online social media has encouraged researchers in many fields to analyze its content for many purposes such as opinion mining, event detection, and others. Since NER plays a vital role in many NLP applications, any of these applications focused on dealing with Twitter content,

This work is licensed under a Creative Commons Attribution 4.0 International Licence. Licence details: http://creativecommons.org/licenses/by/4.0/

Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing,
pages 23–32, Osaka, Japan, December 11-17 2016.

needs to use NER system to improve its performance and deals with Twitter specific challenges. As consequences, Arabic is very complex language compared to European languages. On the one hand, it has both complex and rich morphology as well as ambiguity. On the other hand, there are no capital letters and Arabic texts are written without diacritics. Therefore, building Arabic NLP applications and especially NER is very intriguing.

Most of Arabic NER systems use three approaches: rule-based, machine learning and hybrid approaches. In this paper, we use Deep Neural Networks (DNN) because they are extremely powerful machine learning models that have attained great success in vast applications such as image classification (He et al., 2015) and speech recognition (Hinton et al., 2012). Furthermore, DNNs can achieve state-of-the-art in many NLP applications for instance Machine Translation (Cho et al., 2014; Sutskever et al., 2014) and sentiment analysis (Socher et al., 2013; dosSantos and Gatti, 2014). These powerful models can use backpropagation algorithm for training (Rumelhart et al., 1986).

In order to process variable length input, recurrent neural networks (RNNs) are the best solution (Goller and Kuchler, 1996). In recent years, RNNs are widely used and achieved state-of-the-art in several NLP tasks such as language modeling (Mikolov et al., 2011), machine translation (Cho et al., 2014) and speech recognition (Graves, 2013). We use Long Short-Term Memory (LSTM), which is one kind of RNNs with complex cells (Hochreiter and Schmidhuber, 1997). With its forget gate, LSTM allows highly non-trivial long-distance dependencies to be easily learned.

For sequential labeling tasks, it has been shown that using a bi-directional LSTM model is preferred to LSTM model because it can capture infinite amount of context on both sides for a sentence by eliminating the main problem of limited context in feed-forward neural networks (Graves et al., 2013). In fact, to build a system for Arabic NER for social media, we propose a model based on bidirectional LSTM networks with Conditional Random Field (CRF) layer on the top of the networks.

Most of the existing Arabic NER systems rely on handcrafted engineering features which is time consuming and the use of large gazetteers to improve the accuracy. In this paper, we investigate the impact of using character-level and word embedding features as inputs for bidirectional LSTM network with CRF on the top of the networks as contextual feature on Arabic NER performance for social media. Experimental results show that we are able to obtain state-of-the-art performance on Twitter dataset without using any large gazetteers and lots of handcrafted engineering features.

The main contributions of this paper are the following:

- Study the impact of bidirectional LSTM on sequence tagging like NER on Arabic Twitter texts;
- The effectiveness of using character-level for morphologically rich languages (Arabic as an example) and also show that using word representations improve the system performance;
- Investigate the use of CRF on the top of bidirectional LSTM to capture contextual features in Arabic Twitter texts;
- We get state-of-the-art results and outperform the existing systems on publicly available dataset.

2 Models

In this section, we provide a brief description of the models used in this paper. We begin by presenting the main neural network used: LSTMs and bidirectional LSTMs. Without further ado, CRF will be presented. Finally, we investigate the combination of bidirectional LSTMs and CRF.

2.1 LSTM Networks

Recurrent neural networks (RNNs) are an extension of a conventional feed-forward neural network. They are remarkably general models for sequence processing tasks. RNNs can handle the variable-length sequence using a recurrent hidden unit state whose activation at each time step is dependent on that of the previous one. However, standard RNNs suffer from the problem of vanishing gradients when it comes to long sequences. More recently, (Hochreiter and Schmidhuber, 1997) propose "Long Short-Term Memory" (LSTM) networks to solve this problem. LSTM can learn to bridge minimal time lags for

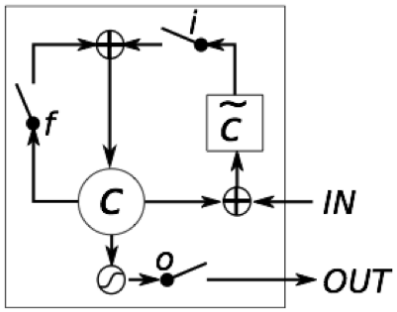

Figure 1: Long Short-Term Memory unit

more than 1000 discrete time steps (Hochreiter and Schmidhuber, 1997). Since then, many researchers in the field came with some minor changes to the original LSTM unit. In this paper, we follow the implementation of LSTM as used in (Graves, 2013). LSTM unit is different from RNNs unit, which simply computes a weighted sum of the input signal and applies a nonlinear activation function. Each LSTM unit maintains a memory cell c_t^j at each time t. The output h_t^j, or the activation, of the j-th LSTM unit is then computed as follow:

$$h_t^j = o_t^j \tanh(c_t^j) \tag{1}$$

where o_t^j is an *output gate* that modulates the amount of memory content exposure. The output gate o_t^j is then computed using the following equation:

$$o_t^j = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t)^j \tag{2}$$

where σ is a logistic sigmoid function. V_o is a diagonal matrix. c_t^j is the memory cell updated by partially forgetting the existing memory and adding a new memory content $\widetilde{c}_t^j$:

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \widetilde{c}_t^j \tag{3}$$

This new memory cell is computed using the following equation:

$$\widetilde{c}_t^j = \tanh(W_c x_t + U_c h_{t-1})^j \tag{4}$$

f_t^j is called the *forget gate*, when its output value is close to zero, the network will effectively forget whatever value it was remembering. i_t^j is called the *input gate*, when its output value is close to zero, this value will be blocked from entering into the next layer. These two gates are computed using the following equations:

$$f_t^j = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})^j \tag{5}$$

$$i_t^j = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})^j \tag{6}$$

It should be noted that V_f and V_i are diagonal matrices. Figure 1 illustrates the graphical representation of LSTM unit.

It would rather be noted that there is a remarkable difference between standard recurrent unit and LSTM unit where the first unit overwrites its content at each time-step, while the second unit has the ability to decide whether to keep the existing memory by using these gates. As a result, LSTM unit can learn important features from input sequence by easily carrying this feature over long distance.

2.2 Bidirectional LSTM Networks

One shortcoming of standard LSTMs is that they are only able to make use of previous context. So, in sequence tagging like NER task, for a given time step, we can have access to both past features (using forward states) and future features (using backward states). To solve this problem, we use bidirectional

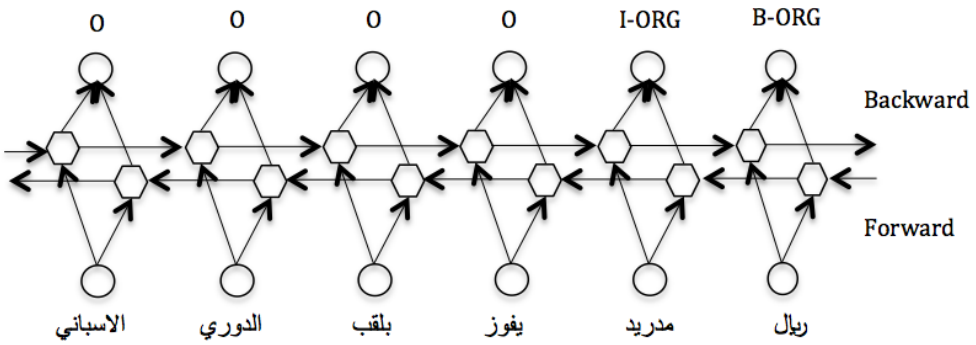

Figure 2: Bidirectional LSTM neural network

LSTMs. These models can process data in both directions where the output layer receives results from the two separate hidden layers. Figure 2 shows a graphical illustration of bidirectional LSTMs with the Arabic sentence "ريال مدريد يفوز بلقب الدوري الاسباني" which means "Real Madrid won the Spanish league title". For a given sentence $(x_1, x_2,...,x_n)$ in Twitter dataset containing n words, we compute two representations: the left context of the sentence at every word t denoted by $\overrightarrow{h}_t$ and the right context of the sentence denoted by $\overleftarrow{h}_t$ by using a second LSTM reading the same sentence in the opposite direction (we note that Arabic texts are written from right to left on the contrary to European languages). Every LSTM network has its own parameters. By using this model, we represent every word in a sentence by concatenating its right and left context representations $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$. These LSTMs networks are trained using backpropagation through time (BPTT) (Boden, 2002).

2.3 Bidirectional LSTM with CRF

We describe our final model, which combines bidirectional LSTM and CRF model. A state transition matrix is used for CRF layer as parameters. To predict the current tag, we use this transition matrix, which represents the past and future tags. We denote this transition matrix by $M_{i,j}$ representing the transition score from the i-th tag to the j-th tag. Given a sentence X = $(x_1, x_2,..., x_n)$, we denote $N([S]_1^T)_{i,t}$ to be the matrix of scores output by the bidirectional LSTM network for the sentence $[S]_1^T$ and the i-th tag at the t-th word. The score for a sentence $[S]_1^T$ along with a sequence of tags $[i]_1^T$ is given by the sum of the transition scores and the scores from the bidirectional LSTM network:

$$s([S]_1^T, [i]_1^T) = \sum_{t=1}^{T} (M_{[i]_{t-1},[i]_t} + N([S]_1^T)_{[i]_t,t}) \tag{7}$$

We use dynamic programming to compute $M_{i,j}$ and optimal tag sequences for inference (Lafferty et al., 2001). Finally, we use a softmax function over all possible tag sequences to get probabilities for the sequence $[i]_1^T$:

$$p(y|[S]_1^T) = \frac{e^{s([S]_1^T,[i]_1^T))}}{\sum_{\tilde{c}\in I_s} e^{s([S]_1^T,\tilde{c})}} \tag{8}$$

where I_s represents all possible tag sequences for a given sentence $[S]_1^T$. We note that in this paper, we use the IOB format (Inside, Outside, Beginning), which was the standard representation in the Twitter dataset (Darwish, 2013). During training, we maximize the log-probability $\log(p(y|[S]_1^T)))$ of the correct tag sequence:

$$\begin{aligned} \log(p(y|[S]_1^T)) &= \log(\frac{e^{s([S]_1^T,[i]_1^T))}}{\sum_{\tilde{c}\in I_s} e^{s([S]_1^T,\tilde{c})}}) \\ &= s([S]_1^T, [i]_1^T) - \log(\sum_{\tilde{c}\in I_s} e^{s([S]_1^T,\tilde{c})}) \end{aligned} \tag{9}$$

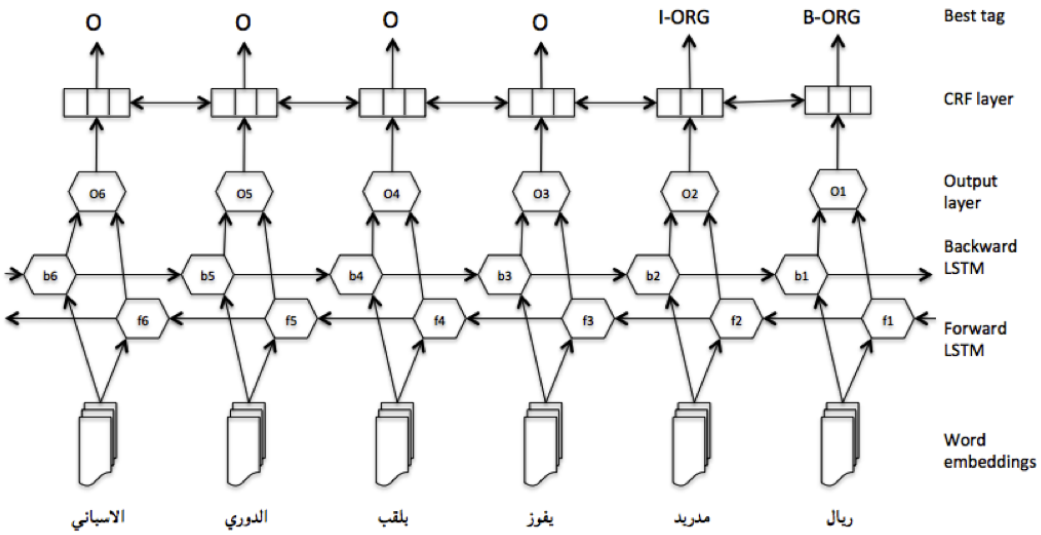

Figure 3: Word embeddings are fed to a bidirectional LSTM where f_j, b_j respectively represent the forward and backward of the word j. O_j represents the concatenation of the previous two vectors resulting in a representation of word j in its context.

3 Training Procedure

The architecture of our model is shown in Figure 3. For each sentence, we initialised our word vectors with pretrained word embeddings (Zahran et al., 2015) (see section 4.2 for more details). The input to the bidirectional LSTM network is the sequence of word embeddings for a given sentence S. Hence, every word in a sentence gets its left and right representation from the bidirectional LSTM. Then, we concatenate these two representations and linearly projected onto another layer whose size is equal to the number of distinct NER tags. As explained before, CRF layer is used on the top of the bidirectional LSTM in order to capture contextual features in the form of neighboring named entity recognition tags. As a result, we get predictions for each word in a sentence.

From equation 7, the model parameters are the parameters of the bidirectional LSTM obtained from the matrix $N([S]_1^T)_{i,t}$, the parameters obtained from the transition matrix of bigram scores $M_{i,j}$ and word embeddings. To train our models, we use Stochastic Gradient Descent (SGD). In each epoch, we divide the training dataset into mini-batches and process one batch at a time. Each mini-batch contains a number of sentences in the training data which is defined by the parameters of mini-batch size. More details will be discussed in the experimental results section.

4 Inputs for the Model

In this section, we explain the inputs to our model. We begin by presenting the character-based model of words used in this paper and how it could be useful especially for morphological rich languages like Arabic. Thus, we present word embeddings used to initialise our word vectors.

4.1 Character-based models of words

It has been shown that using character-level can help to improve the performance of models in many NLP applications. It is viewed as new paradigm in NLP applications using deep neural networks. It is widely used in Neural Machine Translation (NMT) and recent work show that adding character-level features improve the translation results for many languages (Luong and Manning, 2016; Ling et al., 2015; Chung et al., 2016). Hence, character-based approaches have also been applied to other tasks in natural language processing such as document classification (Xiao and Cho, 2016), language modeling (Kim et al., 2015; Ling et al., 2015) and parsing (Ballesteros et al., 2015). For NER, character-level was not extremely used to build models that deal with sequence tagging task. It was used in the context of text classification for

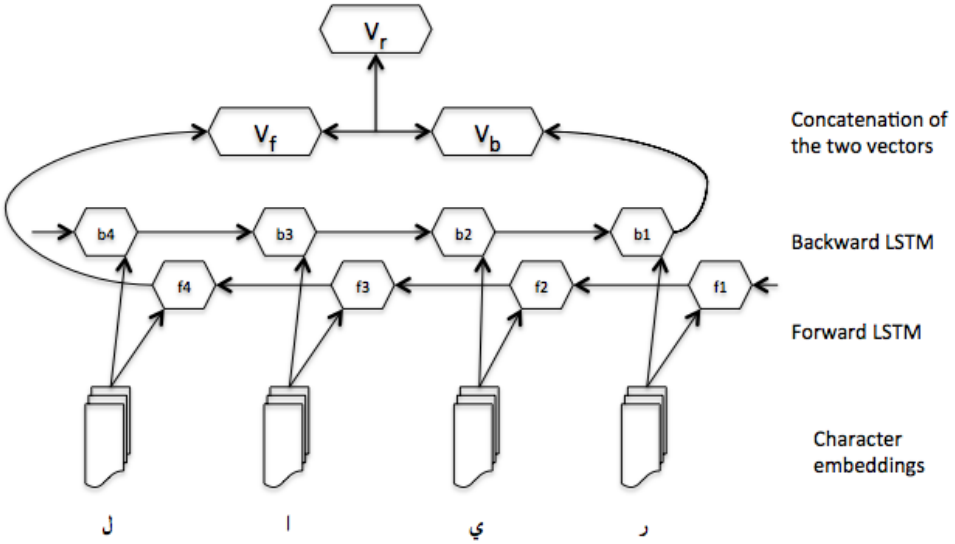

Figure 4: The character embeddings of the Arabic word "ريال" (real) using a bidirectional LSTMs. The final vector V_r is the result of concatenating the vector embedding V_f which represents the forward pass and the vector embedding V_b which represents the backward pass

learning representations of words from their characters (Zhang et al., 2015). As far as we know, we are the first to use character-level representations to build a system for Arabic NER for social media.

Arabic language belongs to the category of languages that has rich morphology. It is also an agglutinative language where some words could mean an entire sentence in English. Most of the Arabic words are constructed using: prefix (es) + stem + suffix (es), thus, it exhibits large vocabulary sizes and relatively high out-of-vocabulary (OOV) rates on the word level. Regardless, word-level embeddings generalize poorly to rarely seen or unseen words and therefore, can significantly impair the performance for high OOV rates. (Luong et al., 2013) proposed another approach based on morphemes as the sub-word unit to improve generalization. Compared to morphemes, characters have the advantage of being directly available from the original text and do not need complex pre-processing steps which makes character-based approach more robust to use in order to improve system performance.

Therefore, to add character-level features in our model, we modify the architecture of our system presented in Figure 3. A graphical illustration of the new architecture is depicted in Figure 4. We use bidirectional LSTMs to compute character-based vector embeddings of Arabic words. Hence, each character is represented with an LSTM cell. We read words character by character from right to left to compute the first vector embedding (V_f). We employ the same process to compute the second vector embeddings (V_b) where we start from the last character. Finally, we concatenate the first and second vectors to get the final representation of the word based on its characters (V_r). This representation is then concatenated with a word-level representation from a word lookup-table. This lookup-table was initialized using word2vec (see the next section).

4.2 Pretrained Word Embeddings

Word representations derived from unlabeled text have proven useful for many NLP tasks, such as part-of-speech (POS) tagging (Huang et al., 2014), named entity recognition (Collobert et al., 2011), chunking (Turian et al., 2010) and parsing (Bansal et al., 2014). In large corpora, names appear in regular contexts which will be fruitful for most of the sequence tagging tasks: like NER. So that, we initialize our word vectors with pretrained word embeddings. (Soricut and Och, 2015) show that using word embeddings can encode morphological information and may provide additional information to the character-based word embeddings.

Purposefully, to test the performance of pretrained word embeddings, we performed two experiments

Embedding	Dimension	F1 Score
Random Initialisation	100	75.22
Glove	100	83.25
Word2vec	100	**85.71**

Table 1: Results with different choice of word embeddings.

	Tokens	PER	LOC	ORG
Twitter-Train	55k	788	713	449
Twitter-Test	26k	464	587	316

Table 2: Twitter Evaluation data statistics.

with different sets of publicly available word embeddings and compare the results with a random sampling method to initialize our model. Table 1 shows the results obtained using the two different word embeddings as well as the randomly sample one. According to the results in Table 1, we got a significant improvements using pretrained word embeddings contrasted to the one using random embeddings. This is consistent with results reported by previous work (Collobert et al., 2011; Chiu and Nichols, 2015; Huang et al., 2015). We state that Arabic pretrained word embeddings for both word2vec and Glove models used in the experiments are publicly available and developed by (Zahran et al., 2015). From the two different embeddings, Word2vec achieves the best results, about 2.46 points in F1 score better than Glove embeddings and 10.49 points in F1 score better than random initialization. In the rest of the paper, we initiliaze our word vectors with pretrained word embeddings using word2vec model.

5 Evaluation

Evaluation was performed on the Twitter dataset developed by (Darwish, 2013). Table 2 gives an overview of this dataset. Before training starts, we split the dataset into sentences by replacing "." and "," by spaces. Every digit in the dataset is replaced by zero.

5.1 Twitter Dataset

We use the training and test datasets developed by (Darwish, 2013) in order to test our model. This dataset was tagged with three types of named entities: location, person and organisation. The training dataset contains tweets randomly selected from the period of May 3-12, 2012. The testing data contains tweets that were randomly selected between November 23, 2011 and November 27, 2011. We mention that these two datasets were annotated using the Linguistics Data Consortium ACE tagging guidelines. As we will see in the experimental results, this dataset was used in (Darwish and Gao, 2014) and (Zirikly and Diab, 2015) for testing.

5.2 Hyperparameters Details

We train our model using backpropagation through time (BBTT) algorithm to update parameters. We use mini-batch stochastic gradient descent (SGD) with a fixed learning rate. We explored more sophisticated optimization algorithms such as momentum, RMSProp (Hinton et al., 2012), Adam (Kingma and Ba, 14) and Adadelta (Zeiler, 2012). Even if some of these methods are considerably used in computer vision

Model	Precision	Recall	F1
B-LSTM	80.95	57.63	67.33
B-LSTM + char	74.07	67.80	70.80
B-LSTM + char + word emb	89.58	72.88	80.37
B-LSTM + char + word emb + CRF	90.57	81.36	**85.71**

Table 3: Twitter dataset results with our models

Model	Precision	Recall	F1
Zirikly and Diab	81.7	46.9	59.59
Darwish and Gao	76.8	56.6	65.20
Our system	**90.57**	**81.36**	**85.71**

Table 4: Comparison of our system with two other models on Twitter dataset

and show better results, our experiments demonstrate that these methods converge very fast than SGD, but none of them perform better than SGD. We use an embedding dimension of 100. For the hidden dimension of our character bidirectional LSTMs, we use 25 for each one. The final dimension of our character-based representation of words is 50.

6 Experimental Results and Discussions

We run many experiments representing the combination of different models and architectures to understand their influence on Arabic NER system for social media. We explored the impact of using CRF as contextual features, pretrained word embeddings and character-level embeddings. Table 3 shows the different results. Experiments show that the marvelous improvement in the overall system performance was observed with the use of pretrained word embeddings (indicated by "word emb" in Table 3) which gives us an improvement by 9.57 points in F1 score. Adding CRF layer provides us an improvement of 5.34 points in F1 score. Using character-level embeddings (indicated by "char" in Table 3) improve our system performance by 3.47 points in F1 score.

We compare our system with two other models. The best score reported on this task was obtained by (Darwish and Gao, 2014). Their system uses large gazetteers, and a semi-supervised method. They got 65.2 points in F1 score. The same Twitter dataset was used by (Zirikly and Diab, 2015) to test their model, which used a lot of handcrafted engineering features and gazetteers. They obtained F1 score of 59.59. Our model outperforms these two models without using any large gazetteers, and with the use of minimal features combined with bidirectional LSTMs. Table 4 shows our results on Arabic NER for social media in comparison with these two systems.

On the one hand, as far as we know, we are the first to explore the impact of character-level embeddings, pretrained word embeddings and contextual features (CRF) to develop a system for Arabic NER for social media. Using character-level embeddings allow our model to learn interesting morphological and orthographic features instead of hand-engineering them. On the other hand, we are the first to use Arabic pretrained word embeddings developed by (Zahran et al., 2015) to initialize our word vectors for Arabic NER for social media and explore their impact on our system performance.

7 Conclusion

In this paper, we have shown that our neural networks model, which uses bidirectional LSTMs, character-level embeddings, pretrained word embeddings, CRF on the top of the neural networks achieves state-of-the-art results in building an Arabic named entity recognition system for social media and surpassing the previous state-of-the-art system by a large margin without the use of any large gazetteer and lots of hand-engineering features.

Given the inflectional and derivational aspect of Arabic language which leads to a language with complex morphological rules, the intuition behind these results lies in incorporating both pretrained word and character-level embeddings, which allow our system to learn interesting morphological features without hand-engineering them. In addition, using CRF as contextual features was another key success for our system.

References

Bogdan Babych and Anthony Hartley. 2003. *Improving Machine Translation Quality with Automatic Named Entity Recognition*, *Proceedings of EACL*,

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. *Improved transition-based dependency parsing by modeling characters instead of words with LSTMs*, *Proceedings of EMNLP*, Lisbon. Association for Computational Linguistics.

Mohit Bansal Kevin Gimpel, and Karen Livescu. 2014. *Tailoring continuous word representations for dependency parsing*, *Proceedings of ACL*, Baltimore. Association for Computational Linguistics.

Mikael Boden. 2002. A guide to recurrent neural networks and backpropagation, *Technical report*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns, *arXiv preprint arXiv:1511.08308*

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. *Learning phrase representations using RNN encoder-decoder for statistical machine translation*, *Proceedings of EMNLP*, Doha. Association for Computational Linguistics.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. *A Character-level Decoder without Explicit Segmentation for Neural Machine Translation*, *Proceedings of ACL*, Berlin. Association for Computational Linguistics

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koran Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch, *Journal of Machine Learning Research*,12(1):2493- 2537.

Kareem Darwish. 2013. *Named entity recognition using cross-lingual resources: Arabic as an example*, Proceedings of ACL, Sofia, Bulgaria. Association for Computational Linguistics.

Kareem Darwish and Wei Gao. 2014. *Simple effective microblog named entity recognition: Arabic as an example*, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, Reykjavik, Iceland.

Cìcero dos Santos and Maìra Gatti. 2014. *Deep convolutional neural networks for sentiment analysis of short texts*, Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics. Technical Papers, Dublin, Ireland Association for Computational Linguistics.

Sergio Ferrndez, Antonio Toral, scar Ferrndez, Antonio Ferrndez, and Rafael Muoz. 2007. *Applying wikipedias multilingual knowledge to crosslingual question answering*, Zoubida Kedad, Nadira Lammari, Elisabeth Mtais, Farid Meziane, and Yacine Rezgui, editors, Reykjavik, Iceland. Springer.

Christoh Goller and Andreas Kuchler. 1996. *Learning task-dependent distributed representations by backpropagation through structure*, Proceedings of the International Conference on Neural Networks,

Alex Graves and Jürgen Schmidhuber. 2005. *Framewise phoneme classification with bidirectional LSTM networks*, Proceedings of IJCNN.

Alex Graves. 2013. Generating sequences with recurrent neural networks, *arXiv preprint*

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. *Coursera, Lecture 6e: rmsprop: divide the gradient by a running average of its recent magnitude*, Neural Networks for Machine Learning,

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory, *Neural Computation*, 9(8):1735–1780.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition, *arXiv preprint*

Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. 2014. Learning representations for weakly supervised natural language processing tasks, *Computational Linguistics*, 40(1).

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging, *arXiv preprint arXiv:1508.01991*

Jun'ichi Kazama and Kentaro Torisawa. 2007. *Exploiting Wikipedia as external knowledge for named entity recognition*, *Proceedings of EMNLP - CoNLL*, Prague, Czech Republic. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models, *CoRR, abs/1508.06615*

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, *Proceedings of ICML*.

Wang Ling, Tiago Luıs, Luıs Marujo, Ramon F. Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. *Finding function in form: Compositional character models for open vocabulary word representation*, Proceedings of EMNLP - CoNLL, Lisbon, Portugal. Association for Computational Linguistics.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. *Better Word Representations with Recursive Neural Networks for Morphology*, *Proceedings of CoNLL*, Sofia, Bulgaria. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D. Manning. 2016. *Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models*, *Proceedings of ACL*, Berlin, Germany. Association for Computational Linguistics.

Tomàš Mikolov, Stefan Kombrink, Lucàš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. *Extensions of recurrent neural network language model*, Proceedings of ICASSP.

Tomàš Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. *Distributed representations of words and phrases and their compositionality*, Proceedings of NIPS.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification, *Journal of Linguisticae Investigationes*, 30(1).

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams 1986. *Learning Internal Representations by Error Propagation*, Symposium on Parallel and Distributed Processing.

Semiocast. 2012. *Geolocation analysis of Twitter accounts and tweets by Semiocast*.

Khaled Shaalan and Mai Oudah. 2014. A hybrid approach to Arabic named entity recognition, *Journal of Information Science*, 40(1):67–87.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. *Recursive deep models for semantic compositionality over a sentiment treebank*, Proceedings of EMNLP, Seattle, USA. Association for Computational Linguistics.

Radu Soricut and Franz Och. 2015. *Unsupervised Morphology Induction Using Word Embeddings*, Proceedings of the NAACL-HLT, Denver, Colorado. Association for Computational Linguistics.

Ilya Sutskever, Orion Vinyals, and Quoc V. Le. 2014. *Sequence to sequence learning with neural networks*, Proceedings of NIPS, Montreal, Canada.

Paul Thompson and Christopher C. Dozier. 1997. *Name searching and information retrieval*, Proceedings of EMNLP, Association for Computational Linguistics.

Hiroyuki Toda and Ryoji Kataoka. 2005. *A Search Result Clustering Method using Informatively Named Entities*, Proceedings of the 7th annual ACM international workshop on Web information and data management (WIDM), ACM Press.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio 2010. *Word representations: A simple and general method for semi-supervised learning*, Proceedings of ACL, Uppsala, Sweden. Association for Computational Linguistics.

Yijun Xiao and Kyunghyun Cho 2016. Efficient character-level document classification by combining convolution and recurrent layers, *arXiv preprint arXiv:1602.00367*

Mohamed A. Zahran, Ahmed Magooda, Ashraf Y. Mahgoub, Hazem Raafat, Mohsen Rashwan, and Amir Atyia. 2015. *Word Representations in Vector Space and their Applications for Arabic*, the Proceedings of CICLing, Cairo, Egypt. Springer.

Matthew D. Zeiler 2012. ADADELTA: an adaptive learning rate method, *CoRR, abs/1212.5701*

Xiang Zhang, Junbo Zhao, and Yann LeCun 2015. *Character-level convolutional networks for text classification*, the Proceedings of NIPS, Montreal, Canada.

Ayah Zirikly and Mona Diab. 2015. *Named Entity Recognition for Arabic Social Media*, Proceedings of the NAACL-HLT, Denver, Colorado. Association for Computational Linguistics.