T.C.

GEBZE TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ II

DERİN ÖĞRENME İLE ADLANDIRILMIŞ ÖĞE TANIMA

NAMED ENTITY RECOGNITION (NER) WITH DEEP LEARNING

Mehmed MUSTAFA

Proje Danışmanı
Dr. Öğr. Üyesi Burcu YILMAZ

Haziran, 2018
Gebze, KOCAELİ

Bu çalışma 06/06/2018 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Bölümü'nde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

Danışman Adı	Dr. Burcu YILMAZ	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

Jüri Adı	Prof. Dr. Yusuf Sinan AKGÜL	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

Contents

FIGURE LIST

ABSTRACT

Pek çok kelime okuyucuda belirli izlenimler oluşturur. Bu izlenimler sayesinde okuyucu bir metinde geçen kelimeler hakkında bir fikir edinir. NLP'de (Doğal Dil İşleme), bu sürece Named Entity Recognition (NER) denir. Bir cümledeki kelimelere karşılık gelen konum, kişi, kuruluş, zaman, para veya tarih gibi birçok öğe etiketi tanımlanabilir. Bu operasyonlardaki en büyük problemlerden biri, kelimenin bir konum, bir kişinin ismi veya bir kurumun ismi olup olmaması veya bir sayının zaman, para ya da bir tarih anlamına gelip gelmediğini anlamakta oluşan karışıklıklardır. Bu projenin amacı, Bi-LSTM, CNN ve CRF kombinasyonlarını kullanarak, hem kelime, hem de karakter düzeyinde sinir ağı mimarilerini uygulamak ve test etmek. Mimariler herhangi bir özellik mühendisliği (feature engineering) veya veri ön işlem (data preprocessing) gerektirmez. Böylelikle bu mimariler bir çok çeşitli dizi etiketleme görevlerinde uygulanabilirler. İngilizce ve Türkçe için modeller hem kelime, hem de karakter düzeyinde eğitilmektedir. Modeller, dizi etiketleme görevleri için iki veri kümesinde değerlendirilmiştir - İngilizce için CoNLL2003 ve İngilizce (EWNERTC) ve Türkçe (TWNERTC) için Vikipedi dökümü veri kümeleri. İngilizce için en iyi sonuçlar CoNLL2003 veri kümesinde elde edilmiştir. Kelime düzeyinde %96.41 doğruluk, 83.80 F1 puanı ve karakter düzeyinde ise %98.22 doğruluk, 89.87 F1 puanı. Türkçe için kelime ve karakter düzeyinde elde edilen en iyi sonuçlar sırasıyla %90,31 doğruluk, 63,85 F1 puanı ve %90,98 doğruluk, sırasıyla 67.33 F1 puanı ve TWNERTC veri kümesinde elde edilmiştir.

Anahtar kelimeler:

Named Entity Recognition,

Sequence Labeling,

LSTM

Haziran, 2018 **Mehmed MUSTAFA**

ABSTRACT

Many words create certain impressions on the reader. With the help of these impressions the reader gains an insight about the words in a text via some entities. In NLP (Natural Language Processing), this process is called Named Entity Recognition (NER). A lot of named entities in the sentence like location, person, organization, time, money or date can be identified. One of the major problems in these operations are confusions about whether the word denotes the name of a location, a person or an organization, or whether an number stands for time, money or a date. This project implements neural network architectures both for word-level and character-level representations by using combination of Bi-LSTM, CNN and CRF. The architectures require no feature engineering or data preprocessing, thus making it applicable to a wide range of sequence labeling tasks. Models for English and Turkish are trained both in word and character level. The models are evaluated on two data sets for sequence labeling tasks – CoNLL2003 for English and Wikipedia dump datasets for English (EWNERTC) and Turkish (TWNERTC). The best obtained results for English for word-level and char-level are 96.41% accuracy, 83.80 F1 and 98.22% accuracy, 89.87 F1 respectively on CoNLL2003 dataset. The best obtained results for Turkish for word-level and char-level are 90.31% accuracy, 63.85 F1 and 90.98% accuracy, 67.33 F1 respectively on TWNERTC dataset.

Keywords:

Named Entity Recognition,

Sequence Labeling,

LSTM

June, 2018 **Mehmed MUSTAFA**

1. INTRODUCTION

Named Entity Recognition (NER) is a state-of-the-art intelligence system that works with nearly the efficiency of a human brain. The system is structured in a such way that it is capable of finding entity elements from raw data and can determine the category in which the element belongs. Generally, it is to identify mentions of entities (persons, locations, organizations, etc.) within unstructured text. However, the diverse and noisy nature of user-generated content as well as the emerging entities with novel surface forms make NER in social media messages more challenging.

Figure 1: An example of NER application on an example text

Most traditional high performance sequence labeling models are linear statistical models, including Hidden Markov Models (HMM) and Conditional Random Fields (CRF). Orthographic features and external resources such as gazetteers are widely used in NER. However, such task-specific knowledge is costly to develop (Ma and Xia, 2014)[1], making sequence labeling models difficult to adapt to new tasks or new domains.

In the past few years, non-linear neural networks with as input distributed word representations, also known as word embeddings, have been broadly applied to NLP problems with great success. Collobert et al. (2011)[2] proposed a simple but effective feed-forward neutral network that independently classifies labels for each word by using contexts within a window with fixed size. Recently, recurrent neural networks (RNN) (Goller and Kuchler, 1996)[3], together with its variants such as long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997[4]; Gers et al., 2000[5]).

In this work, the proposed neural network architecture for sequence labeling (Xuezhe Ma and Eduard Hovy[6]) is implemented and tested with ConLL2003 and Wikipedia Dump Datasets[7] for English and Turkish. It is a truly end-to-end model requiring no task-specific resources, feature engineering, or data pre-processing beyond pre-trained word embeddings on unlabeled corpora. Thus, the tested model can be easily applied to a wide range of sequence

labeling tasks on different languages and domains. The convolutional neural networks (CNNs) (LeCun et al., 1989[8]) is used to encode character-level information of a word into its character-level representation. Then the character- and word-level representations are combined and fed into bi-directional LSTM (BLSTM) to model context information of each word. On top of BLSTM, a sequential CRF is used to jointly decode labels for the whole sentence.

2. RELATED WORK

Recently, many different neural network architectures have been proposed and successfully applied to sequence labeling such as NER. In this section, related works for word and character based NER's will be outlined and discussed.

2.1 Word Based NER

A rich set of language specific name lists and hand-crafted features were used in the recent successful studies on sequence labeling. For example the winning submissions in CoNLL-2002 (for Spanish & Dutch language) and CoNLL-2003 (for English & German language) use a rich set of hand-crafted features along with gazetteers to achieve best results.

Afterwards, semi-supervised approaches have achieved better results by utilizing large unlabeled corpora. Demir and Ozgur (2014) proposes a semi-supervised learning approach to achieve top performance for Czech language. Darwish (2013) exploits cross-lingual features to achieve the top performance on Arabic.

2.2 Character based NER

A Hidden Markov Model (HMM) with character-level inputs are proposed by Klein et al. (2003) to mitigate the data sparsity problem in word-level inputs. Their proposed method reduces the errors over HMM with word-level inputs by 30%. However, the character-level HMM drawback is in the unrealistic independence assumptions and it is not able to compete with their best system where they utilize a maximum-entropy conditional Markov model using richer set of features.

The best published results for English model use pretrained word embeddings along with character-level representation of a word by using a combination fo Convolutional Neural

Network (CNN), bidirectional LSTM and CRF (Ma and Hovy 2016). This project implements the model of Ma and Hovy.

The best announced results for German and Spanish model utilize characters to build word representations with LSTM-CRF and relies on unsupervised word representations (Lample et al. 2016). Gillick et al. (2016) adapt the sequence-to-sequence model used for machine translation (Sutskever et al., 2014) to part-of-speech tagging and NER. Their model inputs the text as sequence of bytes and outputs span annotations of the form: phrase start byte, length of the phrase in bytes, type of the phrase.

3. MODEL

This section gives a detailed view of the components/layers of the tested neural network architecture. The neural layers of the neural network are introduced one-by-one from bottom to top.

3.1 Convolution Neural Network (CNN) for Character-level Representation

CNN is an effective way to get morphological information (prefix or suffix of a word) from characters of words and encode it into neural representations (Santos and Zadrozny, 2014[9]; Chiu and Nichols, 2015[10]).

Figure 2 shows the CNN used in the model (Xuezhe Ma and Eduard Hovy[6]) to extract character-level representation of a given word. This CNN is only different by Chiu and Nichols (2015)[10] by that only character embeddings are used as the inputs to CNN, without features like character type. A dropout layer similar to (Srivastava et al., 2014)[11] is applied before character embeddings are input to CNN.

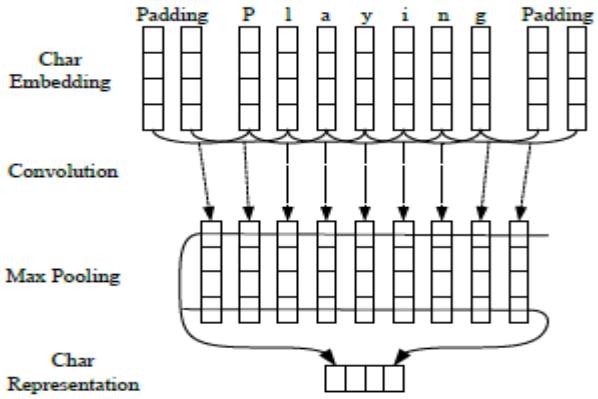

Figure 2: The convolution neural network for extracting character-level representations of words. Dashed arrows indicate a dropout layer applied before character embeddings are input to CNN.[6]

3.2 Bi-directional Long Short-Term Memory (BLSTM)

Recurrent neural networks (RNNs) are a powerful family of connectionist models that capture time dynamics via cycles in the graph. Though, in theory, RNNs are capable to capturing long-distance dependencies, in practice, they fail due to the gradient vanishing/exploding problems (Bengio et al.,1994[12]).

To cope with the gradient vanishing problems LSTMs (Hochreiter and Schmidhuber, 1997[4]) were designed. They are a kind of RNNs. Basically, a LSTM unit is composed of three multiplicative gates which control the proportions of information to forget and to pass on to the next time step. Figure 3 gives the basic structure of an LSTM unit.

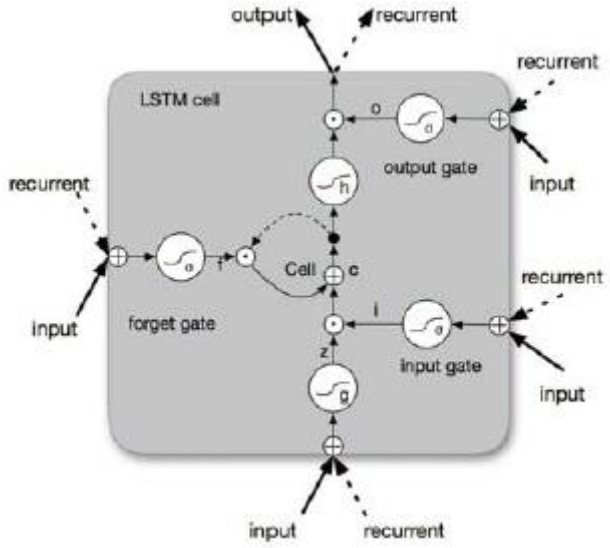

Figure 3: Schematic of LSTM unit.[6]

Formally, the formulas to update an LSTM unit at time t are:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\boldsymbol{W}_i\mathbf{h}_{t-1} + \boldsymbol{U}_i\mathbf{x}_t + \boldsymbol{b}_i) \\
\mathbf{f}_t &= \sigma(\boldsymbol{W}_f\mathbf{h}_{t-1} + \boldsymbol{U}_f\mathbf{x}_t + \boldsymbol{b}_f) \\
\tilde{\mathbf{c}}_t &= \tanh(\boldsymbol{W}_c\mathbf{h}_{t-1} + \boldsymbol{U}_c\mathbf{x}_t + \boldsymbol{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{o}_t &= \sigma(\boldsymbol{W}_o\mathbf{h}_{t-1} + \boldsymbol{U}_o\mathbf{x}_t + \boldsymbol{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}
$$

Where σ is the element-wise sigmoid function and $\odot$ is the element-wise product. xt is the input vector (e.g. word embedding) at time t, and ht is the hidden state (also called output) vector storing all the useful information at (and before) time t. Ui;Uf ;Uc;Uo denote the weight matrices of different gates for input xt, and Wi;Wf ;Wc;Wo are the weight matrices for hidden state ht. bi; bf ; bc; bo denote the bias vectors.

It is beneficial to have access to both past (left) and future (right) contexts for many sequence labeling tasks. However, the LSTM's hidden state ht knows nothing about the future and takes information only from past. An elegant solution whose effectiveness has been proven by previous work (Dyer et al., 2015[13]) is bi-directional LSTM (BLSTM). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are concatenated to form the final output.

3.3 Conditional Random Field (CRF)

For a given input sentence it is useful to consider the interrelationship between labels in neighborhoods and jointly decode the best chain of labels while performing sequence labeling task. For example, in NER with standard BIO2 annotation I-PER cannot follow I-ORG (Tjong Kim Sang and Veenstra, 1999[14]). Accordingly, instead of decoding each label independently, the label sequence is modeled jointly using a conditional random field (CRF) (Lafferty et al., 2001[15]). For a sequence CRF model (only pair of successive label interactions are considered), training and decoding can be solved efficiently by adopting the Viterbi algorithm.

3.4 Final Model Formulation (BLSTM+CNNs+CRF)

Finally, by feeding the output vectors of BLSTM into a CRF layer the final neural network model is obtained. Figure 4 illustrates the architecture of the network in detail. The CNN in Figure 2 computes the character-level representation of each word by the character embeddings passed as inputs. Then the word embedding vector is concatenated with the character-level representation vector to feed into the BLSTM network. Finally, the output vectors of BLSTM are fed to the CRF layer to jointly decode the best label sequence. On both the input and output vectors of BLSTM dropout layers are applied as shown in Figure 4.

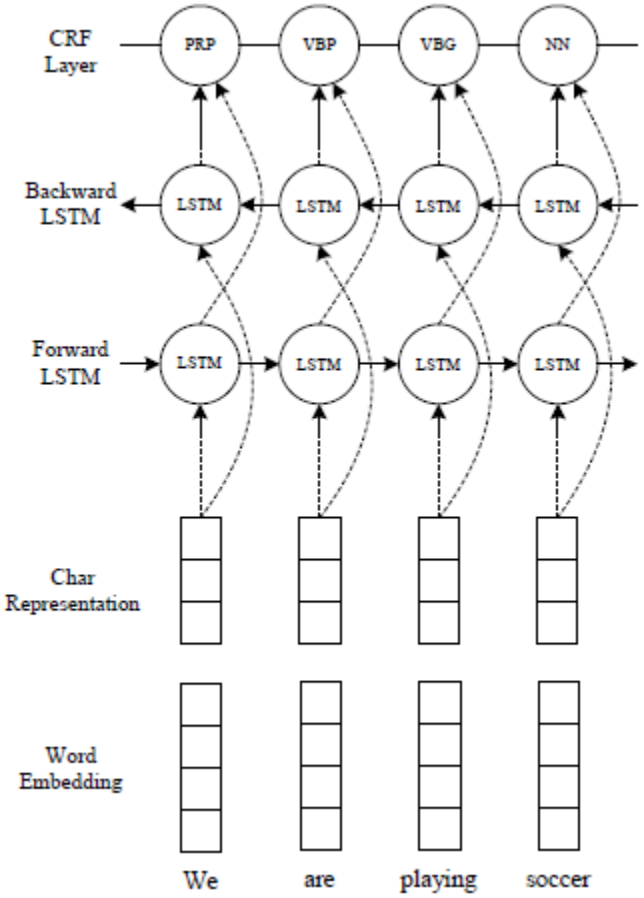

Figure 4: The main architecture of the neural network. The character representation for each word is computed by the CNN in Figure 2. Then the character representation vector is concatenated with the word embedding before feeding into the BLSTM network. Dashed arrows indicate dropout layers applied on both the input and output vectors of BLSTM.[6]

4. EXPERIMENTS

4.1 Data Sets

The different models are trained and tested with two main datasets: ConLL2003 English and Wikipedia Dump Data (EWNERTC and TWNERTC)[7] for English and Turkish languages. These data sets contain four different types of named entities: PERSON, LOCATION, ORGANIZATION and MISC.

Example sentence from the datasets:

Sentence: The Gran Premio d'Italia is a Listed flat horse race in Italy open to three year old thoroughbreds.

Tags: O B-MISC I-MISC I-MISC O O O O O O O B-LOCATION O O O O O O O

4.2 Used configuration and parameters for the models

Same configuration and parameters are used for all the trained models.

Epochs: The total number of the epochs is set to **100**.

Batch Size: 200

Early Stopping: The "best" parameters appear at around **50** epochs, according to the experimets.

Dropout Training: To mitigate overfitting, the dropout method (Srivastava et al.2014[16]) is applied to regularize the model. Best results are obtained by dropout rate at **0.3**

Learning Method: Adam

Learning Decay: Momentum 0.9

Decay rate: 0.005

Clipping: To reduce the effects of "gradient exploding", gradient clipping of **5.0** is used.

LSTM on chars: 100

LSTM on words: 300

Embedding dimension for chars: 100

Embedding dimension for words: 300

4.3 Results for English

- **Word level model:**
  - **EWNERTC Dataset:**

    Training Size: 1,274M tokens

    Evaluating Size: 260K tokens

    Testing Size: 260K tokens

    Training: Acc: 90.25%, Pre: 59.59%, Rec: 46.76%, F1: 52.40

    Testing: Acc: 89.76%, Pre: 57.33%, Rec: 39.74%, F1: 46.94

  - **ConLL2003 English Dataset:**
    - **Training: Acc: 96.41%, Pre: 87.61%, Rec: 80.29%, F1: 83.80**
    - **Testing: Acc: 94.32%, Pre: 77.58%, Rec: 69.97%, F1: 73.58**

- **Character level model:**
  - **ConLL2003 English Dataset:**
    - **Training: Acc: 98.22%, Pre: 90.26%, Rec: 89.48%, F1: 89.87**
    - **Testing: Acc: 96.67%, Pre: 82.76%, Rec: 82.88%, F1: 82.82**

As expected the character level results are slightly better than the word level results. The difference in results between EWNERTC and ConLL2003 datasets is more than expected.

The reason for this is might be the corrupted data in the EWNERTC dataset. There are some data cases in which the tag labels of the words are misplaced.

4.4 Results for Turkish

TWNERTC Dataset is used for the Turkish training and testing.

Training Size: 6,786M tokens

Evaluating Size: 1,359M tokens

Testing Size: 1,359M tokens

- **Word level model:**
  - **Training: Acc: 90.31%, Pre: 64.34%, Rec: 63.37%, F1: 63.85**
  - **Testing: Acc: 88.23%, Pre: 58.17%, Rec: 57.41%, F1: 57.79**

- **Character level model:**
  - **Training: Acc: 90.98%, Pre: 70.77%, Rec: 64.21%, F1: 67.33**
  - **Testing: Acc: 89.46%, Pre: 65.62%, Rec: 59.38%, F1: 62.34**

5. CONCLUSION

In this work a proposed neural network architecture for sequence labeling[6] was implemented and tested with two different datasets on char and word level. The network is a truly end-to-end model relying on no task-specific resources, feature engineering or data pre-processing.

The results have shown that the proposed neural network is usable for different datasets, but the f1 score is lower than the conLL2003 dataset. The proposed network was originally tested on conLL2003 dataset in the work which designed this architecture[6].

6. REFERENCES

1) Xuezhe Ma and Fei Xia, Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In Proceedings of ACL-2014, pages 1337-1348, Baltimore, Maryland, June.

2) Ronan Collobert, Jason Weston, L´eon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.
3) Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In Neural Networks, 1996., IEEE International Conference on, volume 1, pages 347–352. IEEE.
4) Sepp Hochreiter and J¨urgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.
5) Felix A Gers, J¨urgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. Neural computation, 12(10):2451–2471.
6) Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF
7) https://data.mendeley.com/datasets/cdcztymf4k/1 last visited on 30 May 2018
8) Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551.
9) Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In Proceedings of ICML-2014, pages 1818–1826.
10) Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. arXiv preprint arXiv:1511.08308.
11) Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958.
12) Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on, 5(2):157–166.
13) Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In Proceedings of ACL-2015 (Volume 1: Long Papers), pages 334–343, Beijing, China, July.
14) Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In Proceedings of EACL'99, pages 173–179. Bergen, Norway.

15) John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of ICML-2001, volume 951, pages 282–289.
16) Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958.