CharNER: Character-Level Named Entity Recognition

Onur Kuru
Koç University, Turkey
`okuru13@ku.edu.tr`

Ozan Arkan Can
Koç University, Turkey
`ocan13@ku.edu.tr`

Deniz Yuret
Koç University, Turkey
`dyuret@ku.edu.tr`

Abstract

We describe and evaluate a character-level tagger for language-independent Named Entity Recognition (NER). Instead of words, a sentence is represented as a sequence of characters. The model consists of stacked bidirectional LSTMs which inputs characters and outputs tag probabilities for each character. These probabilities are then converted to consistent word level named entity tags using a Viterbi decoder. We are able to achieve close to state-of-the-art NER performance in seven languages with the same basic model using only labeled NER data and no hand-engineered features or other external resources like syntactic taggers or Gazetteers.

1 Introduction

Named Entity Recognition is commonly formulated as a word-level tagging problem where each word in the sentence is mapped to a named entity tag. A typical approach is to slide a window over each word position to extract features for a classifier that produces tags. Moreover, one can allow the classifications at adjacent positions to interact by chaining local classifiers together and perform joint inference. To achieve good performance, one has to overcome the data sparsity problem in the labeled training data. This is achieved by handcrafting good word-level features by exploiting affix, capitalization, or punctuation (Zhang and Johnson, 2003), using the output of syntactic analyzers (part-of-speech taggers, chunkers) and external resources such as gazetteers, word embeddings, word cluster ids to improve performance further (Turian et al., 2010; Ratinov and Roth, 2009). These solutions require significant engineering effort or language specific resources that are not readily available in all languages of interest.

It is even harder to design a multilingual model with handcrafted features considering each language offers different challenges. A distinguishing feature for one language may not be informative for another. For example, capitalization (a typical feature for English NER) is not a distinguishing orthographic feature for the Arabic script. Models for languages with agglutinative or inflectional morphologies may need to utilize the output of a language specific morphological analyzer. In some morphologically rich languages, production of hundreds of words from a given root is common, which makes models even more susceptible to the data sparsity problem.

This work is motivated by the desire to eliminate the tedious work of feature engineering, language specific syntactic and morphological analyzers, and language specific lexical or named-entity resources which are considered necessary to accomplish good performance on Named Entity Recognition. We accomplish this by combining the following ideas: First, instead of considering entire words as the basic input features, we take the characters as the primary representation as in (Klein et al., 2003; Gillick et al., 2016). Second, we use a stacked bidirectional Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) which is able to operate on sequential data of arbitrary length and encode observed patterns in its memory at different scales. Finally, we use a Viterbi decoder to convert the character level tag probabilities produced by the LSTM into consistent word level tags.

Considering characters as the primary representation proves fruitful in several ways. Characters provide sub-word-level syntactic, morphological, and orthographic information that can be directly exploited by our model, whereas word-based models have to incorporate this information using feature engineering. Furthermore, useful sub-word features may vary from language to language, which our model can automatically learn but word-based models would have to incorporate using language-specific resources and features. Finally, character-based models reduce the size of the input vocabulary compared to word-level models, using fewer parameters, increasing computational and statistical efficiency.

We report results similar to or better than the state-of-the-art in resource-free Named Entity Recognition in seven languages. Moreover, our proposed model is not limited to Named Entity Recognition in particular and can be applied to other tagging tasks such as part-of-speech tagging.

In the rest of the paper we discuss related work in Section 2, detail our model and describe the input/output representation, stacked bidirectional LSTMs and inference details in Section 3. In section 4, we describe evaluation datasets in detail and present our experiments and results. Section 5 summarizes our contributions.

2 Related Work

In this section, we first outline the previous work on NER with word-level inputs then move onto character-based NER models. Next, we summarize the applications of character-based models in NLP in general.

2.1 Word based NER

Early successful studies on NER use hand-crafted features and language specific name lists with word-level classifiers. Both of the first place submissions in CoNLL-2002 (Spanish & Dutch), CoNLL-2003 (English & German) NER shared tasks (Carreras et al., 2002; Florian et al., 2003) use a rich set of handcrafted features along with gazetteers to achieve top performance. Subsequently, semi-supervised approaches (Ando and Zhang, 2005; Suzuki and Isozaki, 2008; Turian et al., 2010) have reported better results by utilizing large unlabeled corpora. Demir and Ozgur (2014) employs a semi-supervised learning approach to achieve best result for Czech. Darwish (2013) exploits cross-lingual features and knowledge bases from English data sources to achieve the top performance on Arabic. The current state-of-the-art system for Turkish (Seker and Eryigit, 2012) is based on Conditional Random Field (CRF) and utilizes language dependent features along with gazetteers.

2.2 Character based NER

Klein et al. (2003) introduce a Hidden Markov Model (HMM) with character-level inputs to alleviate the data sparsity problem inherent in word-level inputs. Their character-level HMM achieves a 30% error reduction over an HMM with word-level inputs. However, the character-level HMM suffers from 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 (word, all substrings of the word, part-of-speech and chunk tags). Ma and Hovy (2016) utilize pretrained word embeddings and character-level representation of a word by using a combination of Convolutional Neural Network (CNN), bidirectional LSTM and CRF. They report the best published result for English. Lample et al. (2016) also utilize characters to construct word representations with LSTM-CRF model and relies on unsupervised word representations extracted from unannotated corpora. They announce the best results for German and Spanish. 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. The 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). Our model CharNER has a similar input/output representation with the character-based HMM of (Klein et al., 2003) and employs a much more compact network than (Gillick et al., 2016).

2.3 Other character-based models

Character-based models have been used successfully for NLP tasks other than NER as well. Depending on the nature of the task, characters are utilized in two different ways. One line of work uses characters to form a word representation for each token in a sentence (Kim et al., 2015; Ling et al., 2015a; Ballesteros et al., 2015). Alternatively character representations are used without mapping to words first (Klein et al., 2003; Zhang and LeCun, 2015; Ling et al., 2015b; Dhingra et al., 2016).

Ling et al. (2015a) construct vector representations of words by composing characters using bidirectional LSTMs and Kim et al. (2015) employs a convolutional neural network (CNN) over characters to form word representations. Ling et al. (2015a) achieve state-of-the-art results in language modeling and part-of-speech tagging by utilizing these word representations. Kim et al. (2015) use word representations constructed by CNN with recurrent neural network for language modeling. They show that taking characters as the primary representation is sufficient to encode both semantic and orthographic information and their model is on par with the existing state-of-the-art despite having significantly fewer parameters. Ballesteros et al. (2015) employs the same strategy with (Ling et al., 2015a) to represent each token for a continuous-state dependency parsing model. They show that the parsing model benefits from incorporating the character-based encodings of words for morphologically rich languages.

Zhang and LeCun (2015) demonstrate that convolutional neural networks are successful at mapping characters directly to ontology/sentiment classes and text categories. Ling et al. (2015b) introduce a neural machine translation model that views the input and output sentences as sequences of characters rather than words. They show that the model is capable of interpreting and generating unseen word forms. They achieve translation results that are on par with conventional word-based models. Dhingra et al. (2016) proposes a model which finds vector space representations of whole tweets by utilizing character sequences rather than words. Their model performs significantly better compared to word-based counterpart when the input contains many out-of-vocabulary words or unusual character sequences.

3 Model

In this section we outline the architecture of our model, CharNER. We first define the input/output representation. Next we provide preliminary background on LSTMs and bidirectional LSTMs. Then we explain the deep stacked bidirectional LSTM network used in CharNER. Finally, we detail the decoding process.

3.1 Input/Output Representation

Since Named Entity Recognition is proposed as a word-level tagging problem, all of the proposed data sets use word-level tags to denote named entity phrases. A named entity (NE) phrase may span multiple words, hence a NE tag is composed of concatenation of a position indicator (B- Beginning, I- Inside) and a NE type (PER, ORG, LOC, ...). In addition, an O tag indicates that a token is not inside a NE phrase. A named entity phrase starts with a B- tag and if it consists of multiple words, the following word tags are prefixed with I-.

Our model, however, examines a sentence as a sequence of characters and outputs a tag distribution for each character. Therefore, we convert word-level tags to character-level tags. We abandon the position indicator prefixes (B-, I-) and use phrase types (PER, ORG, ...) directly as character tags. If a character subsequence in a sentence constitutes a NE phrase, all of the characters in that subsequence (including spaces) receive the same NE phrase tag. Otherwise, characters get the outside tag (O). Figure 1 shows word-level and character-level tags for an example sentence.

John	works	for	Globex	Corp.	.
B-PER	O	O	B-ORG	I-ORG	O

J	o	h	n		w	o	r	k	s		f	o	r		G	l	o	b	e	x		C	o	r	p	.		.
P	P	P	P	O	O	O	O	O	O	O	O	O	O	O	G	G	G	G	G	G	G	G	G	G	G	G	O	O

Figure 1: An example sentence with word level and character level NER tags.

3.2 Long Short-Term Memory

Recurrent Neural Networks (RNN) have recently achieved state of the art results in natural language processing tasks such as language modeling (Mikolov et al., 2010), parsing (Dyer et al., 2015), and machine translation (Sutskever et al., 2014). One major problem with simple RNNs is that they are difficult to train for long term dependencies due to the vanishing and the exploding gradient problems (Bengio et al., 1994). Hochreiter and Schmidhuber (1997) developed Long Short-Term Memory (LSTM) to overcome the long term dependency problem. They introduced a special memory cell which is controlled by input, output and forget gates. The input gate controls how much new information should be added to current cell, the forget gate controls what old information should be deleted. The output gate controls the information flow from the cell to the output. Many variants of the LSTM have been developed, in this study we use the LSTM architecture with peephole connections which was proposed by Gers et al. (2000). The LSTM memory cell is defined by the following equations[1]:

$$
\begin{aligned}
f_t &= \sigma(\boldsymbol{W_{fx}} x_t + \boldsymbol{W_{fh}} h_{t-1} + w_{fc} * c_{t-1} + b_f) \\
i_t &= \sigma(\boldsymbol{W_{ix}} x_t + \boldsymbol{W_{ih}} h_{t-1} + w_{ic} * c_{t-1} + b_i) \\
c_t &= f_t * c_{t-1} + i_t * \tanh(\boldsymbol{W_{cx}} x_t + \boldsymbol{W_{ch}} h_{t-1} + b_c) \\
o_t &= \sigma(\boldsymbol{W_{ox}} x_t + \boldsymbol{W_{oh}} h_{t-1} + w_{oc} * c_t + b_o) \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

where σ is the logistic sigmoid function, $\tanh$ is the hyperbolic tangent function and $*$ is element wise multiplication. f, i and o are the forget, input and output gates respectively, c denotes the cell vector, and h is the hidden state vector. All gate vectors and the cell vector have the same dimensionality as the hidden state vector. Bold upper case letters stand for matrices, lowercase variables are vectors, and subscripts indicate the connection (e.g. $\boldsymbol{W_{fx}}$: input to forget gate weight matrix).

3.3 Bidirectional LSTMs

It is a common approach to use both preceding and following tokens to derive features for the current token in natural language processing tasks. If we look at the LSTM equations, the current output depends only on previous inputs, the initial cell value and hidden state. Graves and Schmidhuber (2005) proposed bidirectional LSTM (BLSTM) to gain information from future inputs. In a BLSTM, two LSTM components are present, namely the forward LSTM and the backward LSTM. The forward LSTM traverses the sequence in the forward direction and the backward LSTM traverses same sequence in the reverse order using h_{t+1} and c_{t+1} are used instead of h_{t-1} and c_{t-1} for the gate calculations. For example, input gate at time t is calculated using the following:

$$i_t = \sigma(\boldsymbol{W_{ix}} x_t + \boldsymbol{W_{ih}} h_{t+1} + w_{ic} * c_{t+1} + b_i)$$

In a bidirectional model the output at time t depends on both the forward hidden state $\overrightarrow{h_t}$ and the backward hidden state $\overleftarrow{h_t}$.

3.4 Deep BLSTMs

The CharNER model uses bidirectional stacked LSTMs (Graves et al., 2013) to map character sequences to tag sequences. Figure 2 demonstrates the model overview. The network takes characters as the input sequence and each character is fed into the first forward and backward LSTM layers as one-hot vectors. The output of the first forward and backward layers are concatenated and fed into the next layer. The same process is carried on for the additional BLSTM layers. To obtain the distribution over the tag at position t, an affine transformation followed by a softmax is applied to the hidden representation of the final BLSTM.

[1] We denote matrices with bold upper case letters and vectors with lower case letters.

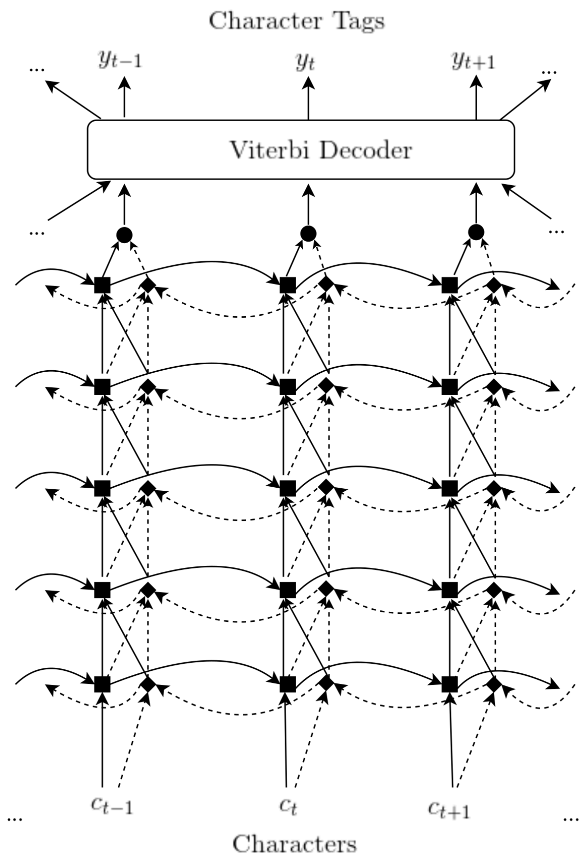

Figure 2: CharNER model: 5-layer Bidirectional LSTM Network with a Viterbi decoder. Each square and diamond node represents a forward and backward hidden LSTM layer, respectively. Circles denote the output layer (i.e. softmax layer). Solid lines show forward connections and dashed lines show backward connections. The model takes characters as an input sequence and each character is represented with a one-hot vector (c_t). A Viterbi decoder takes the sequence of character tag probabilities that is produced by the softmax layer and produces most likely character tag sequence (y) that is consistent at the word level.

3.5 Decoder

The deep BLSTM gives us a tag distribution for each character position. In this section we discuss the final step of turning these character level tag distributions into word level tags.

In early experiments, we observed that the most probable character tags within a word were not always consistent. For example, the model may assign higher probability to person (P) tags in the beginning of a word and organization (G) tags at the end of the same word. Even though the deep BLSTM has access to both left and right input contexts, it is unable to learn word level consistency for output tags. To remedy this, we use a decoder similar to (Wang et al., 2015).

Given a character sequence $c_1, c_2, ..., c_n$ (henceforth denoted with $[c]_1^n$), and a tag set $y \in \mathcal{Y}$, the decoder takes output tag probabilities from the LSTM, $o(c_i)_{y_i} = p(y_i \mid [c]_1^i, [c]_i^n)$, as emission probabilities and exploits transition matrices, A_{ij}, that only allow tags consistent within a word. Three types of transitions can occur between consecutive character tags (y_{i-1}, y_i) according to the position of the character at hand. A character is either followed by a fellow character in the same word ($c \rightarrow c$) or it can be the last character of a word followed by space ($c \rightarrow s$) where c denotes a character inside word boundaries and s denotes the delimiter space. Finally, it can be a space character followed by the first character of the next word ($s \rightarrow c$). The entries in Table 1 show transition matrices A_{ij} for these three states.

	PER	ORG	O	PER	ORG	O	PER	ORG	O
PER	1	0	0	1	0	1	1	0	0
ORG	0	1	0	0	1	1	0	1	0
O	0	0	1	0	0	1	1	1	1
		$c \rightarrow c$			$c \rightarrow s$			$s \rightarrow c$	

Table 1: Example transition matrices for two phrase types (PER, ORG) and Outside (O). c denotes a character inside word boundaries and s denotes the delimiter space.

The score of a sentence $[c]_1^n$ along a path of tags $[y]_1^n$ is computed by the product of transition scores and model output probabilities:

$$s([c]_1^n, [y]_1^n) = \prod_{i=1}^{n} A_{y_{i-1}y_i} o(c_i)_{y_i} \tag{1}$$

The path of tags with the highest sentence score is computed by:

$$[y^*]_1^n = \arg\max_{[y]_1^n} s([c]_1^n, [y]_1^n) \tag{2}$$

To find the path of tags with the highest sentence score we use the Viterbi algorithm (Viterbi, 1967). Once a character tag sequence is decoded without violating word boundaries, it is trivial to convert these tags to word-level tags.

4 Experiments

This section presents our experiments and results. We start by describing the evaluation datasets with their properties. We detail our network training and demonstrate how changing the shape of the network effects performance. We present the results of our model on a variety of languages and compare with related work.

4.1 Datasets

In order to evaluate our system, we applied our model to NER datasets in various languages. The shared tasks of CoNLL-2002 and CoNLL-2003 provide NER data for four languages, Spanish, Dutch, English and German. In addition to these datasets, we evaluated our model on languages with rich morphologies: Arabic, Czech and Turkish. The Turkish NER dataset was prepared by (Tür et al., 2003). For Czech, Konkol and Konopík (2013) make the dataset prepared by (Ševčíková et al., 2007) CoNLL compatible. Benajiba et al. (2007) prepared ANERCorp for Arabic Named Entitiy Recognition. All of the datasets except Turkish and Arabic have conventional training, development and test splits. The Turkish dataset is provided with training and test sets. We separated a small fraction of the training set for development. Since the Arabic dataset (ANERCorp) do not have training and test splits, we applied Monte Carlo cross validation (Arlot et al., 2010) with 3 random runs. All of the datasets are in CoNLL format and we used the CoNLL evaluation script to report phrase-level F1 scores. Table 2 gives the number of sentences.

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Train	3988	4644	15806	14041	12152	8323	30000
Dev.	-	572	2895	3250	2867	1915	2237
Test	797[2]	577	5195	3453	3005	1517	3336

Table 2: Number of Sentences for Training, Development and Test sets.

In some morphologically rich languages production of hundreds of words from a given root is common, which increases data sparsity. We present unique, phrase-wide and corpus-wide unknown word percentages in Table 3 where a word is considered unknown if test set contains it but the word is absent in training. The Unique row counts an unknown word only once while the Phrase and Corpus rows consider all the occurrences of an unknown word in NE phrases and the whole test set, respectively.

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Unique	34.88	43.18	43.42	38.92	47.46	26.85	30.08
Phrase	18.82	35.54	39.28	31.27	42.20	18.39	13.82
Corpus	15.52	21.33	10.08	12.18	14.28	6.25	11.22

Table 3: Unknown Word Percentages.

[2]Randomly sampled with replacement for each run.

We give a brief comparison word-level and character-level views in Table 4. The number of unique tokens and average sequence length are indicated for each language. Switching from word-level to character-level view reduces the number of unique tokens drastically while making the input sequences longer. For direct comparison we trained and evaluated the same CharNER architecture (5 layer BLSTM with hidden size of 128) on both word-level and character-level data. The character-level model yields improved results for all datasets.

		Arabic	Czech	Dutch	English	German	Spanish	Turkish
Char	X	136	136	101	85	96	92	100
	μ_l	149	142	70	76	105	173	90
	F1	75.12	76.87	79.03	90.75	71.64	79.02	93.58
Word	X	27050	34869	27803	23623	32932	26099	63703
	μ_l	27	25	12	14	17	31	13
	F1	74.50	56.77	61.72	84.12	50.68	68.09	91.82

Table 4: Comparison of word-level and character-level views. X denotes the number of unique input tokens and μ_l denotes the average sequence length. F1 gives the best scores achieved on development sets when the same architecture is used on both word-level and character-level data.

4.2 Network Training

We use Adam (Kingma and Ba, 2014) for the gradient based training of the network[3] and we find that the default parameters given in the original study work well for this task. We update our network parameters after each mini-batch of 32 sentences. Before mini-batching, we sorted sentences according to character length to group sentences with similar lengths to speed up training. Also, we shuffle the order of mini-batches prior to each epoch. We use 5 BLSTM layers of size 128 (both forward and backward) stacked on top of each other and apply dropout (Srivastava et al., 2014) to outputs of each layer including the input layer. When no dropout is used, the network overfits the training data quickly and loses the generalization power. Notice that dropout at the input layer turns off bits of characters at random positions in the sequence. Applying dropout to character sequences yields +2 F1 score (absolute). To stabilize the network training, we use gradient norm clipping to prevent gradients from diverging (Pascanu et al., 2012). We set the maximum total norm of the gradients as 1. We tested several configurations of hyperparameters of the network and picked parameters that work well across the all seven languages. Although one may obtain further improvements by tuning hyperparameters as well as depth and width of the network for each dataset individually, we kept the hyperparameters and model configuration same for each language to demonstrate that one can achieve adequate results with the same model across many languages.

	Hidden Size		
Depth	64	128	256
1	52.06	56.27	57.98
2	60.31	68.72	70.77
3	67.22	71.09	70.84
4	70.16	71.70	71.60
5	70.79	72.19	70.53

Table 5: F1 scores on Czech test set for networks with different depths and hidden sizes. Depth denotes the number of layers the network has and Hidden Size denotes the number of hidden units for each layer.

4.3 Network Shape

We experimented with different configurations by varying the number of layers the network has (depth) and number of hidden units in each layer (width). We evaluated different volumed networks on Czech dataset which is modest in size and have conventional training, development and test splits. We summarized the results in Table 5.

Table 5 shows that increasing the depth of the model is more beneficial than increasing the width of the model. Although the wider model (width=256) yields better results when the network is shallow, its advantage disappears as the network gets deeper. Nevertheless, performance of narrow model (width=64) is limited compared to the model with medium width (width=128).

4.4 Need for a Decoder

As discussed in Section 3.5, the deep BLSTM does not output probabilities that always favor a single tag within a word. Character level NER is a structured learning problem, i.e. there are dependencies between the outputs that are difficult to capture by the deep BLSTM which only has access to the input sequence. To quantify the effect of capturing these dependencies, we ran a simpler baseline model for comparison. We did not apply Viterbi decoding to the output probability distributions of the network and picked the most probable tag for each character. Then, we used majority voting between characters of the same word to assign single tag to a word. With this scheme, we observed 2 F1 (absolute) performance drop on Czech dataset.

4.5 Results and Discussion

Here we present the performance of our model on languages with diverse characteristics. Since our model only uses the labeled training data, and no external resources such as gazetteers, we selected previous works which report the top scores without use of any additional data to make a fair comparison. We also included the best results which do make use of arbitrary external resources for each language. We summarized all the comparisons in Table 6.

The results highlight that our model attains good performance and it is robust across variety of languages. We achieve similar to or better than the state-of-the-art in Named Entity Recognition that use no external resources. Abdul-Hamid and Darwish (2010) employ a CRF sequence labeling model which is trained on features that primarily use character n-gram of leading and trailing letters in words and word n-grams. They assert that the proposed features help overcome some of the morphological and orthographic complexities of Arabic. Although they do not utilize any external resource, they apply Arabic specific input preprocessing before training which may be the reason for better performance. Demir and Ozgur (2014) employs a window-based classifier approach with language independent features for

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Best	84.30 [1]	75.61 [2]	82.84 [3]	91.21 [4]	78.76 [5]	85.75 [5]	91.94 [6]
	79.90	68.38	78.08	80.79	-	-	82.28
Best w/o External	81.00 [7]	68.38 [2]	78.08 [3]	84.57 [3]	72.08 [3]	81.83 [3]	89.73 [2]
CharNER	78.72	72.19	79.36	84.52	70.12	82.18	91.30

Table 6: Phrase-level F1 scores. The bottom row presents our model's results across seven languages. The middle row consists of models that report the top scores while not using any external resources, comparable to our model. The top row presents state-of-the-art models that achieve the best results in each language utilizing external resources like word embeddings or Gazetteers. The top row also reports the scores of best models when they only use NER training data, if available. Works are numbered in the order of appearance: [1] (Darwish, 2013), [2] (Demir and Ozgur, 2014), [3] (Gillick et al., 2016), [4] (Ma and Hovy, 2016), [5] (Lample et al., 2016), [6] (Seker and Eryigit, 2012), [7] (Abdul-Hamid and Darwish, 2010)

[3]Code repository: https://github.com/ozanarkancan/char-ner

Czech and Turkish. We outperform their results by a considerable margin for both languages. Gillick et al. (2016) reports the top scores with no external resources for Dutch, English, German and Spanish. They achieve higher F1 score for German dataset, however, our model performs better for Dutch and Spanish despite having 50% less parameters. Our result is on par with (Gillick et al., 2016) for English dataset.

Most state-of-the-art NER models are obtained by handcrafting good word-level features and generally utilizing external information sources. Models usually resort to additional training resources: (1) when training and test set are not from the same data generating distribution (English) or (2) training set is small (Arabic and Czech). Ma and Hovy (2016) report the best published F1 score (91.21%) for CoNLL-2003 English dataset. Without pretrained word embeddings however, their model loses approximately 10 F1 score (absolute). Lample et al. (2016) also relies on unsupervised word representations extracted from unannotated corpora. They announce the best results for German and Spanish datasets. On the other hand, Demir and Ozgur (2014) utilize pretrained word embeddings along with their corresponding word cluster ids to achieve top performance on Czech dataset. Darwish (2013) outperforms (Abdul-Hamid and Darwish, 2010) by 3.4 F1 score by exploiting cross-lingual features and knowledge bases from English data sources. Moreover, language specific expertise can be incorporated to improve the performance. The current state-of-the-art system for Turkish (Seker and Eryigit, 2012) achieves 91.94% F1 score by using language dependent features along with gazetteers. Finally, Gillick et al. (2016) achieves the best result for Dutch by using concatenated Dutch, English, German, Spanish training sets as one. Even though we do not use any additional training source, the performance of our model is competitive for Czech, Dutch, Spanish and Turkish.

5 Contributions

We describe a character-level tagger employing a deep bidirectional LSTM architecture and evaluate it on the Named Entity Recognition task. We showed that taking characters as the primary representation is superior to considering words as the basic input unit. Our main contribution is to show that the same deep character level model is able to achieve good performance on multiple languages without hand engineered features or language specific external resources.

In our current research, we are exploring ways to boost the performance of our model using semi-supervised and transfer learning. Moreover, our method may be promising for languages written without space characters such as Chinese and Japanese since word segmentation error affects the score of NER. Also, there is nothing specific to NER in our model, we are planning to evaluate it on other tasks such as part-of-speech tagging and shallow parsing.

Acknowledgements

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) grants 114E628 and 215E201.

References

Ahmed Abdul-Hamid and Kareem Darwish. 2010. Simplified feature set for arabic named entity recognition. In *Proceedings of the 2010 Named Entities Workshop*, pages 110–115. Association for Computational Linguistics.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Sylvain Arlot, Alain Celisse, et al. 2010. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. *arXiv preprint arXiv:1508.00657*.

Yassine Benajiba, Paolo Rosso, and José Miguel Benedíruiz. 2007. Anersys: An arabic named entity recognition system based on maximum entropy. In *Computational Linguistics and Intelligent Text Processing*, pages 143–153. Springer.

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.

Xavier Carreras, Lluis Marquez, and Lluís Padró. 2002. Named entity extraction using adaboost. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics.

Kareem Darwish. 2013. Named entity recognition using cross-lingual resources: Arabic as an example. In *ACL (1)*, pages 1558–1567.

Hakan Demir and Arzucan Ozgur. 2014. Improving named entity recognition for morphologically rich languages using word embeddings. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 117–122. IEEE.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Felix Gers, Jürgen Schmidhuber, et al. 2000. Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194. IEEE.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306, San Diego, California, June. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

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

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D Manning. 2003. Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 180–183. Association for Computational Linguistics.

Michal Konkol and Miloslav Konopík. 2013. Crf-based czech named entity recognizer and consolidation of czech ner research. In *Text, Speech, and Dialogue*, pages 153–160. Springer.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Gökhan Akin Seker and Gülsen Eryigit. 2012. Initial explorations on using crfs for turkish named entity recognition. In *COLING*, pages 2459–2474.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*, pages 665–673.

Gökhan Tür, Dilek Hakkani-Tür, and Kemal Oflazer. 2003. A statistical information extraction system for turkish. *Natural Language Engineering*, 9(02):181–210.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Andrew J Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*.

Tong Zhang and David Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 204–207. Association for Computational Linguistics.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Magda Ševčíková, Zdeněk Žabokrtský, and Oldřich Krůza. 2007. Named entities in czech: annotating data and developing ne tagger. In *Text, Speech and Dialogue*, pages 188–195. Springer.