Neural Reranking for Named Entity Recognition

Jie Yang and **Yue Zhang** and **Fei Dong**
Singapore University of Technology and Design
{jie_yang, fei_dong}@mymail.sutd.edu.sg
yue_zhang@sutd.edu.sg

Abstract

We propose a neural reranking system for named entity recognition (NER), leverages recurrent neural network models to learn sentence-level patterns that involve named entity mentions. In particular, given an output sentence produced by a baseline NER model, we replace all entity mentions, such as *Barack Obama*, into their entity types, such as *PER*. The resulting sentence patterns contain direct output information, yet is less sparse without specific named entities. For example, "PER was born in LOC" can be such a pattern. LSTM and CNN structures are utilised for learning deep representations of such sentences for reranking. Results show that our system can significantly improve the NER accuracies over two different baselines, giving the best reported results on a standard benchmark.

1 Introduction

Shown in Figure 1, named entity recognition aims to detect the entity mentions in a sentence and classify each entity mention into one out of a given set of categories. NER is typically solved as a sequence labeling problem.

Traditional NER systems use Hidden Markov Models (HMM) (Zhou and Su, 2002) and Conditional Random Fields (CRF) (Lafferty et al., 2001) with manually defined discrete features. External resources such as gazetteers and human defined complex global features are also incorporated to improve system performance (Ratinov and Roth, 2009; Che et al., 2013). Recently, deep neural network models have shown the ability of learning more abstract features compared with traditional

[Barack Obama] $_{PER}$ was born in **[hawaii]** $_{LOC}$.
Rare **[Hendrix]** $_{PER}$ song draft sells for almost $ 17,000 .
[Volkswagen AG] $_{ORG}$ won 77,719 registrations .
[Burundi] $_{LOC}$ disqualification from **[African Cup]** $_{MISC}$ confirmed .
The bank is a division of **[First Union Corp]** $_{ORG}$.

Figure 1: Named Entity Recognition.

statistical models with indicator features for NER (Zhang et al., 2015).

Recurrent Neural Network (RNN), in particular Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), shows the ability to automatically capture history information over input sequences, which makes LSTM a proper automatic feature extractor for sequence labeling tasks. Different methods have been proposed by stacking CRF over LSTM in NER task (Chiu and Nichols, 2016; Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016). In addition, it is possible to combine discrete and neural features for enriched information, which helps improve sequence labeling preformance (Zhang et al., 2016).

Reranking is a framework to improve system performance by utilizing more abstract features. A reranking system can take full advantage of global features, which are intractable in baseline sequence labelling systems that use exact decoding. The reranking method has been used in many NLP tasks, such as parsing (Collins and Koo, 2005), QAs (Chen et al., 2006) and machine translation (Wang et al., 2007; Shen et al., 2004).

Some work has adopted the reranking strategy for NER. Collins (2002) tried both a boosting algorithm and a voted perceptron algorithm as reranking models on named-entity boundaries (without classification of entities). Nguyen et al. (2010) applied Support Vector Machine (SVM) with kernels to reranking model, obtaining significant improvements in F-measure on CoNLL 2003 datasets. Yoshida and Tsujii (2007) used a sim-

Proceedings of Recent Advances in Natural Language Processing, pages 784–792,
Varna, Bulgaria, Sep 4–6 2017.
https://doi.org/10.26615/978-954-452-049-6_101

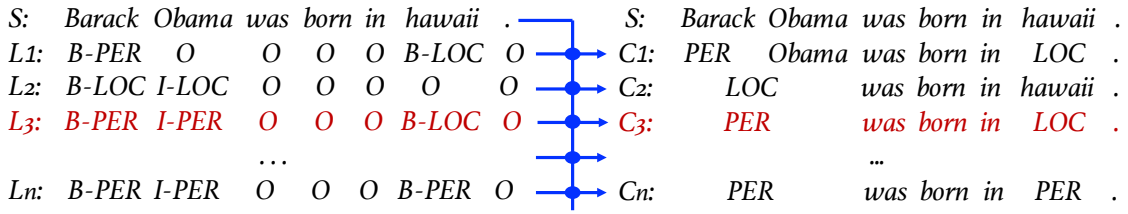


Figure 2: Example of generating collapsed sentence patterns from baseline NER output label sequences.

ple log-linear rerank model on a biomedical NER task, also obtaining slight improvemts. All the above methods use sparse manual features. To the best of our knowledge, there has been no neural reranking model for NER task.

Our work is also in line with neural sentence representation topic. Related tasks such as paraphrase detection (Socher et al., 2011), sentiment classification (HLTCOE, 2013; Le and Mikolov, 2014) are all benifiting from neural models. Le and Mikolov (2014) proposed an unsupervised algorithm to learn the representation of sentences, paragraphs and even documents. Convolution Neural Network (CNN) structure has been used to represent sentences for classification task (Kim, 2014). Palangi et al. (2016) embedded sentences with simple LSTM model but got good results on the web document retrieval task. Zhu et al. (2015b) utilized syntactic information into sentence representation using tree-LSTM.

In this paper, we propose a simple neural reranking model for NER. The model learns sentence patterns that involve output named entities automatically, using neural network. Take the sentence *"Barack Obama was born in hawaii ."* as an example, Figure 2 illustrates several candidate sentence patterns such as *"PER was born in LOC ."* (C_3) and *"LOC was born in hawaii ."* (C_2), where *PER* represents entity type *persons* and *LOC* means *locations*. It is obvious that C_3 is a much more reasonable sentence pattern compared to C_2. To generate the sentence patterns above, we replace predicted entities in candidate sequences with their entity type names. This can effectively reduce the sparsity of candidate sequences, as each entity type contains open vocabulary names (e.g. *PER* can be *Donald Trump*, *Hillary Clinton* etc.), which can bring noise when learning the sentence patterns. In addition, since the learned sentence patterns are global over output structures, it is difficult for baseline sequence labeling systems to capture such patterns.

Description	Feature Template
word grams	$w_i, w_i w_{i+1}$
shape, capital	$Sh(w_i), Ca(w_i)$
capital + word	$Ca(w_i)w_i$
connect word	$Co(w_i)$
capital + connect	$Ca(w_i)Co(w_i)$
cluster grams	$Cl(w_i), Cl(w_i w_{i+1})$
prefix, suffix	$Pr(w_i), Su(w_i)$
POS grams	$P(w_i, w_i w_{i+1}, w_{i-1} w_1 w_{i+1})$
POS + word	$P(w_0)w_0$

Table 1: Features of discrete *CRF* for NER, $i \in \{-1, 0\}$.

We develop a neural reranking model which captures candidate pattern features using LSTM and auxilliary neural structures, including CNN (Kim, 2014; Kalchbrenner et al., 2014) and character based neural features. The learned global sentence pattern representations are then used as features for scoring by the reranker. Results over a state-of-the-art discrete baseline using CRF and a state-of-the-art neural baseline using LSTM-CRF show significant improvements. On CoNLL 2003 test data, our model achieves the best reported result.

Our main contributions include (a) leveraging global sentence patterns that involve entity type information for NER raranking, (b) exploiting auxilliary neural features to enrich basic LSTM sequence representation and (c) achieving the best F1 result on CoNLL 2003 data. The source codes of this paper are released under GPL at `https://github.com/jiesutd/RerankNER`.

2 Baselines

Formally, given a sentence S with t words: $S = \{w_1, w_2, ..., w_t\}$, the task of NER is to find out all the named entity mentions from S. The dominate approach takes the task as a sequence

labelling problem, where the goal is to generate a label sequence $L = \{l_1, l_2, ..., l_t\}$, where $l_i = p_i e_i$. Here p_i is an entity label, $p_i \in \{B, I, O\}$, where B indicates the beginning of an entity mention, I denotes a non-beginning word of a named entity mention and O denotes a non-named-entity word [1]. e_i indicates the entity type. In the CoNLL dataset that we use for our experiments, $e_i \in \{PER, ORG, LOC, MISC\}$, where "$PER$" indicates a *person* name; "LOC", "ORG", "$MISC$" represent *location*, *organization* and *miscellaneous*, respectively.

We choose two baseline systems, one using discrete CRF with handcrafted features and one using neural CRF model with bidirectional LSTM structure, both baselines giving the state-of-the-art accuracies among their respective category of models.

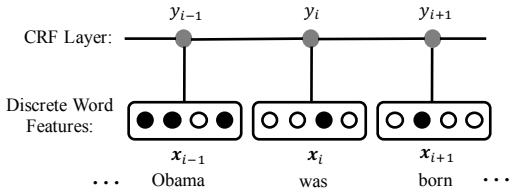
2.1 Discrete CRF

We choose a basic discrete CRF model as our baseline tagger. As shown in Figure 3(a), discrete word features are first extracted as binary vectors (black and white circles) and then fed into a CRF layer. Taking those discrete features as input, the CRF layer can give *n-best* predicted sequences as well as their probabilities. Table 1 shows the discrete features that we used, which follow the definition of (Yang et al., 2016). Here *shape* means whether characters in word are belonging to number, English character or not. *capital* is the indication if word starts with *upper-case* English character, *connect words* include five types: "of", "and", "for", "-" and other. Prefix and suffix include the 4-level prefixes and suffixes of each words.

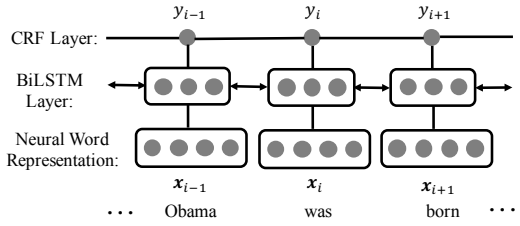
2.2 Neural CRF

A neural CRF with bidirectional LSTM structure is used as our second baseline, which is shown in Figure 3(b). Word representations are represented with continious vectors (gray circles), which are fed into a bidirectional LSTM layer to extract neural features. A CRF layer with *n-best* output is stacked on top of the LSTM layer to decode the label sequences based on the neural features. We use the neural structure of Ma and Hovy (2016), where the word representation is the concatenation of word embedding and a CNN output on the character sequence of the word.

[1] When $p_i = O$, e_i equals to NULL.



(a) Discrete CRF baseline.



(b) Neural CRF baseline.

Figure 3: Baselines.

3 Reranking Algorithms

3.1 Collapsed Sentence Representation

Given the *n-best* output label sequences of a baseline system $\{L_1, L_2, ..., L_i, ..., L_n\}$, where $L_i = \{l_{i1}, l_{i2}, ..., l_{it}\}$, we learn a reranking score $s(L_i)$ for L_i by firsting converting L_i into a sequence pattern C_i, and then learning a representation $h(C_i)$ as its dense representation. To convert candidate sequence L_i to collapsed sequence C_i. We use the following rules to convert each label sequence L_i into a collapsed sentence pattern C_i.

If the L_i include entity labels (e.g. l_{i1} =*B-PER*, l_{i2}=*I-PER*), then the entity labels are replaced with the corresponding entity type name (e.g. $\{l_{i1}, l_{i2}\}$ $\rightarrow$ *PER*, C_{i1} = *PER*), else labels are replaced by its corresponding words ($C_{ix} = w_x$). In the example shown in Figure 2, $S = \{$*Barack Obama was born in hawaii .*$\}$ and $L_3 = \{$*B-PER I-PER O O O B-LOC O*$\}$. The corresponding collapsed sequence is $C_3 = \{$*PER was born in LOC .*$\}$, *Barack Obama* and *hawaii* are regarded as entities and hence are replaced by the entity names, i.e. *PER* and *LOC*, respectively.

3.2 Neural Features

Given a collapsed sentence representation C_i, we use neural network to learn its overall representation vector $h(C_i)$, which is used for the scoring of C_i.

Word Representation: We use *SENNA* (Collobert et al., 2011) embedding to initialize the word embedding of our reranking system. For out

of vocabulary words , embeddings are randomly initialized within $(-\sqrt{\frac{3.0}{wordDim}}, \sqrt{\frac{3.0}{wordDim}})$, where $wordDim$ is the word dimension size (Ma and Hovy, 2016).

Character features are proved useful in capturing morphological features, such as word similarity and dealing with the out-of-vocabulary problem (Ling et al., 2015). As shown in Figure 4(a), we follow Ma and Hovy (2016) by utilizing CNN to extract character-level representation [2]. Input character sequences are firstly passed through the embedding layer to lookup the character embeddings. To extract local features, a *convolution layer* with a fixed window-size is applied on top of the embedding layer. Then we use a *max-pooling* layer to map varying length vectors into a fixed size output vector. Finally, word representation is the concatenation of character CNN output vectors and word embeddings.

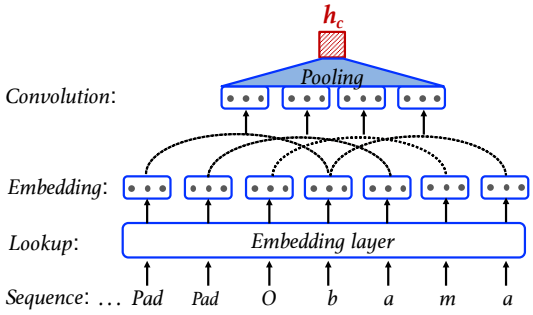
LSTM features: We choose a word-based LSTM as the main network, using it for capturing global sentence pattern information. For input sequence vectors $\{x_1, x_2, ..., x_t\}$, our LSTM model is implemented as follows:

$$
\begin{aligned}
h_t &= tanh(M_t) \odot o_t \\
i_t &= \sigma(W_1 h_{t-1} + W_2 x_t + \mu_1 \odot M_{t-1} + b_1) \\
f_t &= \sigma(W_3 h_{t-1} + W_4 x_t + \mu_2 \odot M_{t-1} + b_2) \\
\widetilde{M_i} &= tanh(W_5 y_{t-1} + W_6 x_i + b_3) \\
M_t &= i_i \odot \widetilde{M_i} + f_i \odot M_{t-1} \\
o_t &= \sigma(W_7 h_{t-1} + W_8 x_t + b_4),
\end{aligned}
$$

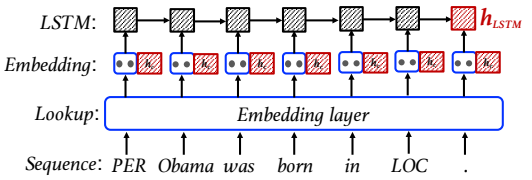
where $\odot$ is the element-wise multiply operator, σ is the sigmoid function, and $\{W, b, \mu\} \in \Theta$ are parameters. i_t, f_t, M_t and o_t are the *input gate, forget gate, memory cell* and *output gate*, respectively. h_t is the hidden vector at step t in the input sentence. As shown in Figure 4(b), word representations are the concatenation of word embeddings and character CNN output (red block). We choose the hidden vector in last word h_{LSTM} as the representation of the input sequence.

CNN features: We introduce CNN to capture local features of the candidate sequences. It consists of a *filter* $W \in R^{h \times k}$ which operates on a context of k words to produce local order features. Max pooling layer is employed over the convolutional layer to extract the most salient features.

[2] Characters are padded into a fixed length by using a special token *Pad*.



(a) CNN character sequence representation for word.



(b) LSTM word sequence representation for sentence.

Figure 4: Representation.

Assume u_j is the concatenation of word representations in Eq. (1) centralized in the embedding z_j in a given sequence $u_1, u_2, ..., u_L$, CNN applies a matrix-vector operation to each window of size k successive window along the sequence in Eq. (2).

$$u_j = (z_{j-(k-1)/2}, ..., z_{j+(k-1)/2}) \quad (1)$$

$$r_i = max_{1<j<L}(Wu_j + b)_i, i = 1, ..., d, \quad (2)$$

where z_j is the j-th word embedding in the given sequence, d is the output dimension of the CNN. $h = [r_1, ..., r_i, ..., r_d]$ is the fixed-size feature representation for the sequence after pooling.

The *CNN* representation structure is similar to Figure 4(a) but its input is word representations rather than character embeddings. We define the CNN features of word sequence as h_{CNN}.

3.3 Score Calculation

After the LSTM and CNN features of collapsed sequence C_i are extracted, we concatenate them together and feed the result into a *softmax* layer.

$$
\begin{aligned}
h(C_i) &= h_{LSTM} \oplus h_{CNN} \\
s(C_i) &= \sigma(W h(C_i) + b),
\end{aligned}
\quad (3)
$$

where $\oplus$ represents the concatenating operation, $h(C_i)$ is the final representation of collapsed sequence C_i and $s(C_i)$ is the output score of C_i.

Parameter	Value	Parameter	Value
n-best	10	peepholes	no
wordDim	50	charDim	50
LSTM hidden	100	dropout	0.2
charCNN filter	50	batch size	128
wordCNN filter	100	λ	0.001
charCNN length	3	*Adam* β_1	0.1
wordCNN length	3	*Adam* β_2	0.999
learning rate	0.001	*Adam* ϵ	1e-8

Table 2: Hyperparameters of reranker.

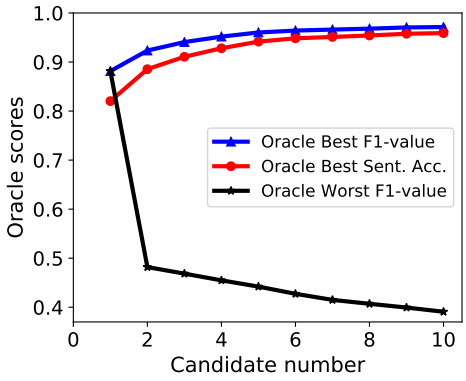


Figure 5: Oracle scores in baseline outputs.

3.4 Decoding

We use a mixture reranking strategy during decoding. Denote the candidate label sequence set on sentence S as $C(S) = \{C_1, C_2, ..., C_n\}$. We take advantage of both the reranker prediction score and the baseline tagger's output probability, using the score

$$\hat{y_i} = \arg \max_{C_i \in C(S)} (\alpha s(C_i) + (1-\alpha) p(L_i)), \quad (4)$$

where $\alpha \in [0, 1]$ is an interpolation weight, which is a hyperparameter tuned on the development set. $p(L_i)$ is the probability of label sequence L_i in the baseline tagger.

3.5 Training

For each training triplet $\{S, L_i, C_i\}$, given the golden sequence L_{golden}, we calculate the tag accuracy $y_i \in [0, 1]$ of each candidate sequence based on L_i and L_{golden}. The same decoding process is applied to each collapsed sequence (C_i, y_i). We use a logistic regression model with mean square error (MSE) as the loss function, with a l_2-regulation term [3] :

$$J(\Theta) = \frac{1}{|\mathcal{D}|} \sum_{(C_i, y_i) \in \mathcal{D}} (y_i - s(C_i))^2 + \frac{\lambda}{2} ||\Theta||_2^2 \quad (5)$$

where Θ are all the parameters to be trained, $\mathcal{D}$ is the training set and λ is the regulation factor.

Adam (Kingma and Ba, 2015) is used to update model parameters.

4 Experiments

4.1 Settings

We use *CRF++* [4] as our discrete baseline CRF implementation and default parameters are used. For neural baseline, we follow the same structure and settings of the state-of-the-art system (Ma and Hovy, 2016). When building the neural reranking system, *SENNA* embedding with 50 dimensions is used to initialize word embeddings. Hyperparameters of reranking system are listed in Table 2.

As we use the mixture strategy in Eq. (4) during decoding, we search the ideal interpolation weight α within $[0, 1]$ in a step of *0.005* based on the preformance under the development set.

4.2 Reranking Data

All of our experiments are evaluated on the standard CoNLL 2003 English dataset (Tjong Kim Sang and De Meulder, 2003), which is a collection of Reuters newswire articles. The CoNLL 2003 English dataset includes 14,987 training sentences, 3,466 development sentences and 3,684 test sentences, annotated into 4 entity types, i.e. *persons(PER)*, *locations(LOC)*, *organizations(ORG)* and *miscellaneous(MISC)*.

To construct the reranking training data, we conduct five-fold jackknifing, spliting the training set into 5 equal parts. In each case, the baseline tagger trains the model with 4/5 of the data and decode the remaining 1/5 to generate *n-best* candidate label sequences. For the reranking development and test data, the full training set is used to encode baseline tagger and decode development/test sentences with *n-best* output. All the *n-best* candidate sequences are converted into collapsed sequences following Section 3.1.

4.3 Baseline Oracle Results

The discrete baseline achieves 92.13% of F1-measure in development set and 88.15% in test set. Our neural baseline gives 94.58% and 91.25% on

[3] We also tried *max-margin* criterion like (Zhu et al., 2015a), while the results are similar with regression model.

[4] https://taku910.github.io/crfpp/

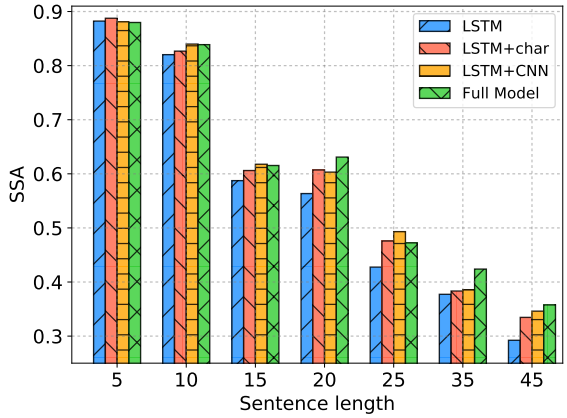


Figure 6: SSA with sentence length.

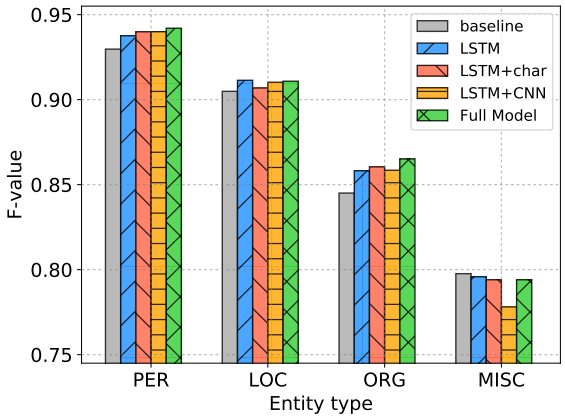


Figure 7: F1-value comparision by entity types.

development and test data, respectively. The discrete baseline for example. Figure 5 shows different oracle scores varying with *n-best* in discrete baseline. The oracle best is obtained by always chooseing the best sequence in the *n-best* candidates, and *vice versa* for the oracle worst. The orcale best sentence accuracy (OBA) [5] represents the accuracy of the sequence choice within the *n-best* candidates under oracle best assumption, and the orcale best F1-value (OBF) is the corresponding F1-value similarly. Orcale worst F1-value (OWF) is the F1-value under the worst choice situation.

As the figure shows, the larger n is, the better is the OBA, which means that a potentially better reranking result is possible. On the other hand, the OWF also drops, which means that the reranking task is more difficult. In our experiments, *n-best* is set as 10, the orcale best F1-value of test set achieves 97.13% (+8.98%) while its orcale worst F1-value drops 49.07% to 39.08%.

4.4 Influence of Sentence Length

We perform development experiments to evaluate model performance on various sentence lengths. Figure 6 shows results by reranking the discrete baseline. Here sentence select accuracy (SSA) is calculated using the corrected number of sentences divided by the total number of sentences. The *x-axis* is the sentence length range (e.g. 10 means sentence length range from 5 to 10), while the *y-axis* corresponds to SSA within 10-best candidates before the mixture strategy (without mixing baseline output probability).

[5]Notice this is different with accuracy which represents the correct rate of tags, OBA represents the correct rate in sentence level.

As shown in the Figure 6, the accuracies of all model settings drop as the size of the sentence increases, which demonstrates that longer sentences are more challenging to our neural rerankers as they are to the baseline models. This can be because for longer sentences, candidate collapsed sequences have higher overlapping proportions and hence are more difficult to distinguishing by reranker. Both character information and CNN local features are useful for enhancing the SSA over a LSTM-only baseline. With the integration of character information and CNN features, our full model reranker can improve its performance on all sentence length ranges, especially for long sentences.

4.5 Influence of Entity Type

Figure 7 shows the comparision of models on different entity types. Compared with the baseline, entities with type of *PER* and *ORG* receive the most improvements, showing that sentence patterns are useful for those types. The performance on entities with type of *MISC* decreases slightly, since *MISC* includes various entity types which bring noise on learning sentence patterns. We believe that our model can benifit more from the NER corpus with fine-grained entity type. Table 3 shows the F1-value and SSA (after mixing baseline output probability) of our reranker on test data with different neural features on the discrete baseline. The word based LSTM reranker achieves the F1-value of 88.75%, with 0.6% absolute improvement over the baseline tagger. Cooperating with CNN features on word only does not make much improvement, while character CNN features are more effective (+0.78%). However, the full

Model (%)	F1	ΔF1	SSA	ΔSSA
Baseline	88.15	0	83.31	0
LSTM	88.75	0.60	84.41	1.10
LSTM+CNN	88.79	0.64	84.63	1.32
LSTM+char	88.93	0.78	84.69	1.38
Full model	**89.25**	1.10	**85.12**	1.82

Table 3: F1-value and SSA on test set.

combination of character representation and word CNN features improves the F-value to 89.25% (+1.10%) with the significance level of $p < 0.05$ with *t-test*. The trend of SSA is the same as the F1-value, the accuracy is improved from the baseline 83.31% to 85.12% using the full model reranker, with an absolute improvement of 1.82%.

4.6 Effectiveness of Reranking

Table 4 shows our rerank results on two baselines and the comparison with state-of-the-art systems. Our reranker on discrete baseline compares favourably to the best discrete models, including the use of external corpus (Kazama and Torisawa, 2007; Suzuki and Isozaki, 2008). It also outperforms Nguyen et al. (2010) which builds a discrete reranking model by utilizing SVM with kernels. Ratinov and Roth (2009)* achieves 90.57% in discrete model by combining global features and abundant external lexicons, while its performance drops to 88.55% when removing the global features (Ratinov and Roth, 2009). Luo et al. (2015) gives the best discrete result (91.20%) by jointing NER with disambiguation task together.

Collobert et al. (2011) builds a first neural NER model with comparable performance to discrete models on CoNLL 2003 corpus. Most state-of-the-art neural NER models utilize bidirectional LSTM with a CRF layer (Huang et al., 2015). Lample et al. (2016) and Ma and Hovy (2016) concatenate character representation with word embedding and Chiu and Nichols (2016) even merge lexicon features into word representation. Passos et al. (2014) obtain a 90.90% by combining discrete features and neural word embeddings in a CRF model. Our neural baseline, which takes the same features as Ma and Hovy (2016), achieves 91.25% in F-value. Our reranker on this baseline outperforms all the previous models with the F-value of 91.62%, which is the best reported F-score on CoNLL 2003.

Discrete Model (%)	F1
Kazama and Torisawa (2007)	88.02
Suzuki and Isozaki (2008)	89.92
Nguyen et al. (2010)	88.16
Ratinov and Roth (2009)	88.55
Ratinov and Roth (2009)*	90.57
Luo et al. (2015)	91.20
Discrete baseline	88.13
Our reranker	89.25
Neural Model (%)	**F1**
Collobert et al. (2011)	89.59
Passos et al. (2014)	90.90
Huang et al. (2015)	90.10
Chiu and Nichols (2016)	90.77
Lample et al. (2016)	90.94
Ma and Hovy (2016)	91.21
Neural baseline	91.25
Our reranker	**91.62**

Table 4: Comparison of state-of-the-art systems.

4.7 Examples

Figure 8 gives some example outputs on the development dataset for which discrete baseline gives incorrect outputs yet the reranker corrects the mistake. Our reranker learns better sentence patterns by correcting both named entity boundary errors and named entity type errors.

In the first case, example 1 shows that *"U.N. Ambassador Albright"* in sentence *"U.N. Ambassador Albright arrives in Chile ."* is incorrectly tagged as a *organization* by the baseline and the entity boundary is incorrect either. By building the collapsed sentences as the input of our reranker, entities such as *"U.N. Ambassador Albright"* are replaced as a single entity name *"ORG"*. Our reranking model learns that *"... PER arrives in LOC ..."* is more possible compared to *"... ORG arrives in LOC ..."*, thereby the candidate with the reasonable entity boundary and type is picked by our reranker.

For the second case, the entity type of *"EL SALVADOR"* in example 3 *"SOCCER - U.S. BEAT EL SALVADOR 3-1 ."* is incorrectly recognized as *organization* by baseline. Our reranker corrects this entity type error by giving higher score to sentence pattern *"... LOC BEAT LOC ..."* rather than pattern *"... LOC BEAT ORG ..."*.

Baseline 1	**[U.N. Ambassador Albright]** $_{ORG}$ arrives in **[Chile]** $_{LOC}$.
Reranker 1	**[U.N.]** $_{ORG}$ Ambassador **[Albright]** $_{PER}$ arrives in **[Chile]** $_{LOC}$.
Baseline 2	West **[Indian]** $_{MISC}$ all-rounder **[Phil Simmons]** $_{PER}$ took four ...
Reranker 2	**[West Indian]** $_{MISC}$ all-rounder **[Phil Simmons]** $_{PER}$ took four ...
Baseline 3	SOCCER - **[U.S.]** $_{LOC}$ BEAT **[EL SALVADOR]** $_{ORG}$ 3-1 .
Reranker 3	SOCCER - **[U.S.]** $_{LOC}$ BEAT **[EL SALVADOR]** $_{LOC}$ 3-1 .
Baseline 4	... prisoners are held in **[Rangoon]** $_{LOC}$'s **[Insein Prison]** $_{PER}$.
Reranker 4	... prisoners are held in **[Rangoon]** $_{LOC}$'s **[Insein Prison]** $_{LOC}$.
Baseline 5	**[PAKISTAN]** $_{LOC}$ WIN TOSS , PUT **[ENGLAND]** $_{ORG}$ INTO BAT.
Reranker 5	**[PAKISTAN]** $_{LOC}$ WIN TOSS , PUT **[ENGLAND]** $_{LOC}$ INTO BAT.

Figure 8: Output examples. The first two examples illustrate the correction of entity boundary errors and the followings show the correction of entity type errors.

5 Conclusion

We proposed a neural reranking architecture for NER by exploiting neural structure to learn sentence patterns. Given the candidate label sequences generated from a baseline tagger, we replace the predicted entity words with the corresponding entity type names to build collapsed sentences, which are used as inputs of a neural reranking model. A mixture reranking strategy is used to combine both the knowledge of the probability from the baseline tagger and the reranker score. Experiments on both discrete and neural baselines show our reranking system improves NER performance significantly, obtaining the best results on CoNLL 2003 English task .

One problem of current method is that all the candidates share the same non-entity words, which lead the neural representations similar, especially for long sentences. In future work, we will develop neural tree structures based on entity position, which can enlarge the difference between candidate sequences. Intuitively, we believe the entities contribute more than non-entity when modeling the sequence vector, *attention* model (Bahdanau et al., 2015) may help collect more information from the intermediate vector of sentences.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and Zhiyang Teng, Zhongqing Wang for their meaningful discussion. Yue Zhang is the corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations* .

Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *HLT-NAACL*. pages 52–62.

Yi Chen, Ming Zhou, and Shilong Wang. 2006. Reranking answers for definitional qa using language modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1081–1088.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370. https://transacl.org/ojs/index.php/tacl/article/view/792.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 489–496.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1):25–70.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

JHU HLTCOE. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. *Atlanta, Georgia, USA* 312.

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

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences pages 655–665. http://www.aclweb.org/anthology/P14-1062.

Junichi Kazama and Kentaro Torisawa. 2007. Exploiting wikipedia as external knowledge for named entity recognition. In *EMNLP-CoNLL*. pages 698–707.

Yoon Kim. 2014. Convolutional neural networks for sentence classification pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference for Learning Representations* .

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition pages 260–270. http://www.aclweb.org/anthology/N16-1030.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1188–1196.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation pages 1520–1530. http://aclweb.org/anthology/D15-1176.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proc. EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

Truc-Vien T Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2010. Kernel-based reranking for named-entity extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 901–909.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(4):694–707.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution pages 78–86. http://www.aclweb.org/anthology/W/W14/W14-1609.

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*. Association for Computational Linguistics, pages 147–155.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*. pages 177–184.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*. pages 801–809.

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

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147.

Wen Wang, Andreas Stolcke, and Jing Zheng. 2007. Reranking machine translation hypotheses with structured and web-based language models. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, pages 159–164.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features forsequence labeling. In *International Conference on Intelligent Text Processing and Computational Linguistics*.

Kazuhiro Yoshida and Jun'ichi Tsujii. 2007. Reranking for biomedical named-entity recognition. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*. Association for Computational Linguistics, pages 209–216.

Meishan Zhang, Jie Yang, Zhiyang Teng, and Yue Zhang. 2016. Libn3l: a lightweight package for neural nlp. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on EMNLP*. pages 612–621.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 473–480.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015a. A re-ranking model for dependency parser with recursive convolutional neural network pages 1159–1168. http://www.aclweb.org/anthology/P15-1112.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015b. Long short-term memory over recursive structures. In *International Conference on Machine Learning*. pages 1604–1612.