Dynamic Entity-Based Named Entity Recognition Under Unconstrained Tagging Schemes

Feng Zhao, Xiangyu Gui, Yafan Huang, Hai Jin, *Fellow, IEEE*, and Laurence T. Yang

Abstract—As increasingly more textual information becomes available, named entity recognition (NER) systems are thriving, benefiting from powerful models and expressive tagging schemes that promote the full use of diverse features at different levels. To improve performance, traditional approaches have focused mainly on changing the structures of NER models but have always ignored the hard constraints and left the NER tagging schemes unchanged. To solve this problem, this article proposes a dynamic entity-based NER approach under unconstrained tagging schemes. To eliminate the constraints, we reorganize widely used tagging schemes and propose two novel unconstrained schemes: one in which tags are assigned to words and entities separately, and one where words and entities are labeled indiscriminately by uniformly taking them as chunks. Associated with the unconstrained tagging schemes, two entity-based neural architectures are also presented that recognize entities at the same time that the sentence is dynamically segmented. Unlike other static NER models that process all the tags after labeling each word, our models address the inputs dynamically by the interactions between the input text and the output labels. The dynamic mechanism can ensure that the entity-level features are included in the NER system, which is helpful for correctly recognizing entities. Except for word embeddings pretrained from unlabeled corpora, no external language-specific knowledge or other resources such as gazetteers are used. The experiments with English, German, Dutch, and Spanish datasets show that our methods can perform very well with different languages. Particularly, the results of the recall rate against the entity's length reveal that the proposed entity-based models are suitable for recognizing entities with long lengths.

Index Terms—Neural network, named entity recognition, entity-based model, unconstrained tagging schemes

1 Introduction

Named entity recognition (NER) is an important topic in the natural language processing (NLP) field. It is fundamental to many artificial intelligence tasks, such as information extraction, machine translation, automatic question answering, and knowledge graphs. NER aims to extract entities from a sequence of raw text and classify them into specific categories, including *Person*, *Location* and *Organization*. In addition to these three categories, in the NER datasets CoNLL2002 [1] and CoNLL2003 [2], a *Miscellaneous* category is also defined for the entities that belong to other categories. NER is generally treated as a sequence-labeling task, giving each word a single tag to indicate whether it is a part of an entity in a unique category. In the existing research, probabilistic graphical models (PGMs) and other statistical learning methods are widely used to predict these tags, such as the hidden Markov model (HMM) [3], support vector machine (SVM) [4], and conditional random field (CRF) [5]. In these methods, handcrafted linguistic features and domain-specific knowledge (e.g., *gazetteers*) are widely used to jointly make decisions, and feature engineering is a primary approach to enhance the capability of the model. Unfortunately, high-quality handcrafted features are complicated to define and are difficult to generalize to other languages or domains, which makes adapting PGMs to different NER tasks a challenge. Thus, there is a need for an efficient approach that is able to dynamically and accurately perform NER.

In recent years, neural networks (NNs) have been adopted in NER. Collobert [6] proposed a fully connected NN solution and achieved excellent performance in most NLP tasks. Rather than precisely defining handcrafted features, more attention has been focused on constructing intricate structures of NNs, such as the bidirectional long short-term memory (LSTM)-CRF model [7], [8] and the bidirectional LSTM-convolutional neural network (CNN) model [9]. The mainstream approach in most existing NER models is to exploit the features in each single word. At the word level, words are embedded into a fixed-dimension feature space by word embedding methods [10], [11], [12]. Rather than initializing the words randomly in NER models, the embedding vectors are fed into the models to represent the hidden features in the words. Word embedding is almost a standard method for most NLP tasks. At the character level, features are represented by feeding the character sequence of a word into an LSTM network [8] or a CNN [9]. However, entity-level features, providing precise information, are seldom taken into account in these models, which influences accuracy. To construct an entity-based model, two technical problems should be solved: one is to explore and discover names of entities from the raw text. For example, given a sentence without any

- F. Zhao, X. Gui, Y. Huang, and H. Jin are with the National Engineering Research Center for Big Data Technology and Systems, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China.
E-mail: {zhaof, guixy, hyfshishen, hjin}@hust.edu.cn.
- L.T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada. E-mail: ltyang@ieee.org.

Manuscript received 27 Nov. 2019; revised 10 Apr. 2020; accepted 6 May 2020. Date of publication 1 June 2020; date of current version 9 July 2022.
(Corresponding author: Feng Zhao.)
Recommended for acceptance by Kim-Kwang Raymond Choo.
Digital Object Identifier no. 10.1109/TBDATA.2020.2998770

annotations, the model must resolve the names from the words out of entities. The other is to find a proper way to represent these names; since these names are of different lengths, it is challenging to represent them in a uniform style.

In addition to entity information, another challenging problem in the NER task is the hard constraint hidden in the tagging scheme. Current labeling schemes make the features overlap, which leads to NER models being complex to learn. In the inside-out-begin (*IOB*) and inside-out-begin-end-single (*IOBES*) formats for NER [13], words are labeled in a "position-category" hybrid annotation style, from which NER systems suffer in at least three aspects: First, features for segmentation and classification are overlapped and arduous for learning in this style. Second, hard constraints are inevitable during the prediction process, which brings about inexplicable NER results. Finally, under these word-level tagging schemes, tags are designed only for words, and it is very difficult to utilize entity-level information in NER models. CRF [6] and SemiCRF [14] are effective models for solving these problems to some extent, but they never optimize the tagging schemes. To address this issue, we propose two unconstrained tagging schemes with different styles that separate positional annotations from categorical annotations. In particular, in a dynamic labeling style, tags are not designed restrictively for words but also for chunks and entities. This tagging strategy is very effective, and almost no hard constraints are found in the predicted tags when cooperating with ripple LSTM, which is verified in our experiments.

In this paper, we propose a dynamic entity-based NER approach under unconstrained tagging schemes. Two entity-based models, the entity-centric model (ECM) and the ripple-LSTM model (R-LSTM), are proposed. In ECM, NER is completed in two phases: the NE extraction phase and NE classification phase. Entities are extracted by segmenting a sentence into chunks of entities and then represented by an entity-character embedding method, which will map the entities of any length into the same semantic vector space of words. These embeddings, which contain hidden features in entities, will be explicitly exploited in the classification phase. The two phases are integrated into a single phase using the R-LSTM model. This strategy takes advantage of a sliding window to detect entities in the sentence. The content in the window, which may be a word, an uncompleted entity chunk or an entity, will be represented in a uniform way by an LSTM network. Although strong features for entities can be obtained by the gazetteer extracted from an external knowledge base [15], our methods are purely neural-based without any specific resources. Specifically, both models are based on an LSTM network. They are segment-level models in which sentence segmentation is an essential part. More importantly, in our models, entities are recognized as a whole, and entity-level features are adopted during the decision-making process. Our main contributions are as follows:

- A dynamic entity-based method is proposed to extract and classify NEs separately. To further exploit word-level and character-level information in NER, this paper focuses on the hidden features in the entity and represents them in a uniform way, irrespective of the length of the entity. Entity-level information is fully used in the classification stage to promote classification accuracy. Specifically, the entity-character representation method can map entities into the same semantic space of words. Although its simple structure is easy to build and train, its performance is greatly improved compared with those of up-to-date complicated NER models.
- Two unconstrained tagging schemes are proposed, and their properties are quantitatively analyzed and compared with those of existing schemes. Our tagging schemes break through the limitation that the tagging target is restricted to a word. Rather, we generalize the target to an entity or even a chunk with arbitrary words. Compared with traditional schemes, the proposed dynamically styled tagging scheme is simple, unconstrained and expressive.
- Two segment-level NER models are constructed, and their performances are tested on four different languages. No external handcrafted features or knowledge bases are used in our models. Together with the unconstrained tagging schemes, no CRF layer is needed for their neural architectures. In particular, in the R-LSTM that we build, a sliding Bi-LSTM structure is innovatively used to simultaneously segment the sentence and recognize entities. The F1 scores are comparable with those of state-of-the-art methods on the four tested languages.

The remainder of this paper is organized as follows. Work related to this paper is discussed in Section 2. In Section 3, we provide an overview of the unconstrained tagging schemes for dynamic NER. In Section 4, the dynamic NER models are discussed. The experiments and results are presented in Section 5. In Section 6, we conclude our paper with directions for future work.

2 Related Work

NER is a traditional problem with flexible methods. In terms of NER models, feature engineering and knowledge representation, NER systems have remarkably varied over time. We summarize the existing NER works as word-level and segment-level works according to the source of NER features. Our entity-based models belong to the segment level but refer to the representation methods used in the word level.

2.1 NER at Word Level

The features for word-level NER models come from words. In early research, statistical learning methods were widely used. In CoNLL2002, Carreras *et al.* [16] achieved the best score on Dutch and Spanish using a combination of several small decision trees. Many other models, such as HMM, SVM and CRF, were also introduced by [3], [5], [17] and obtained acceptable results. The features used in their models were customized for a specific language, which will encounter problems when generalizing to other languages or domains. To overcome the problem of generalization, Agerri and Rigau [18] presented a multilingual NER approach based on a set of cross-language features and obtained superb performance on five different languages.

However, designing proper handcrafted features for these models is difficult and requires a considerable amount of linguistic knowledge.

NNs have taken over many arduous tasks in NER feature engineering. Neural models have shown great potential in NLP since Collober *et al.* [6] refreshed several records in the field, including NER tasks in which a CNN combined with a CRF layer was used. Inspired by this work, Huang *et al.* [7] constructed a bidirectional LSTM-CRF network for sequence-labeling tasks. Chiu and Nichols [9] proposed an LSTM-CNN model and achieved state-of-the-art results in English. In these works, some handcrafted features were still used to augment their models. Words are represented by distributed embeddings that are trained from a large amount of unlabeled corpora using specific models, such as Word2Vec [10] and GloVe [12]. The semantic information hidden in each word can be represented by a fixed-dimension vector. In addition to word embedding, many character embedding methods have been explored in NER tasks. Lample *et al.* [8] utilized a bidirectional LSTM (Bi-LSTM) network to obtain character-level representations. Furthermore, Yadav *et al.* [19] introduced an affix embedding method and obtained state-of-the-art results on three different languages.

In the models that we propose, the word-char embedding method is used to extract orthographic features and language-specific features using the LSTM network as in [8]. In addition, we extend the method to the region of representing entity chunks. Though language modeling methods such as BERT [20] were popular recently, they are beyond the scope of this paper, which focuses on the main architectures for NER.

2.2 NER at Segment Level

In segment-level models, sentences are segmented coinciding with the recognition of entities. Features from segments (generally entities) can be fully utilized. Two-phase methods that separated NER into an extraction phase and a classification phase were proposed in the biomedical NER field [21], [22]. Our ECM model adopts the two-phase architecture and relies on an NN in each phase.

Compared with two-phase models, the SemiCRF [23] model was a more concise model, in which labels were assigned to segments rather than to individual elements. Rather than modeling the constraints hidden in the tags for words in the traditional linear-chain CRF model, SemiCRF concentrated on the transition features in tags for segments. Similar to the CRF model, SemiCRF can be combined with NNs in sequence-labeling tasks. Zhuo *et al.* combined SemiCRF with a gated recursive CNN (grConvs) [24] and applied it in sequence-labeling tasks. Sato *et al.* [25] improved the model and achieved the best NER score in English. However, the encoding and decoding processes were still intricate in the model, especially when pruning the candidate segment lattice for the networks.

Our models are segment-level models and utilize entity-level information in NER. Under unconstrained tagging schemes, entities are labeled as a whole in a dynamic way without any help from the CRF. The function of R-LSTM is similar to the transition-based model Stack-LSTM [8], but their structures are completely different. In Stack-LSTM, four storage and computation-consuming LSTM sequences are employed, which makes it difficult to train the network well. Our R-LSTM resorts to only two LSTM sequences by a sliding window strategy. Moreover, our R-LSTM NER results greatly surpass those of Stack-LSTM in four languages.

3 Unconstrained Tagging Schemes for Dynamic NER

In this section, we analyze the shortage in traditional labeling formats from the perspective of hard constraints and improve them based on the analysis.

3.1 Motivation

With the development of NER, the tagging scheme becomes increasingly important. There is a natural bias, a hard constraint, in tagging schemes, and we define it as follows:

Definition 1 (Hard constraint). *Hard constraint refers to the two-gram labeling that does not conform to the grammar of the tagging scheme.*

In the NER task, *IOB* and *IOBES* are two widely used schemes since the *IOB* format has been developed by [26] for the text chunking task. Regardless of which format is used, the performance of the NER system suffers from hard constraints in the following aspects.

- *Entities are difficult to extract as their length increases.*In traditional tagging schemes, each word acts as a voter for the entity. Strictly speaking, an entity is not recognized until the vote result is consistent and conforms to the schemes' grammatical rules. As the length grows, there are more chances to trigger the hard constraints between the adjacent words.
- *The rules are hard to learn as more information will be conveyed by the tags.*The labeling system becomes more complicated when the tags are set to represent a variety of positions and categories. As the analysis progresses, styles of hard constraints will increase quadratically with the number of entity categories. In this situation, complicated rules between the sequential tags will make the entity unrecognized and threaten the whole system's performance.

A comparison of various tagging schemes is shown in Table 1, in which the *IOB* and *IOBES* formats in their original style are presented. Assume that there are four kinds of entities to be resolved, they are PER (person), LOC (location), ORG (organization) and MISC (miscellaneous). For the *IOB-original* format, 9 tags are needed including B-*c*, I-*c* (*c* indicates the four listed categories), and O. In the *IOBES-original* format, 17 tags are utilized. The hard constraints are easy to find in these schemes. A count reveals that there are 28 types of hard constraints in the *IOB-original* format and even 209 in the *IOBES-original* format, though the total two-tag permutation in the *IOBES-original* format is 289 (17^2).

3.2 Improvements

These hard constraints mainly come from a hybrid location-category annotation style such as B-*c* and I-*c*. Based on the analysis, two unconstrained tagging schemes in a divided and dynamic style are proposed.

TABLE 1
Comparison of Different Tagging Schemes

Style	Format	Target	Tags	#tags	Illegal/Total
original	IOB	word	B-c, I-c, O	9	28/81
	IOBES	word	B-c, I-c, E-c, S-c, O	17	209/289
divided	IOB	word+entity	B, I, O PER, LOC, ORG, MISC	7	**1/9**
	IOBES	word+entity	B, I, O, E, S PER, LOC, ORG, MISC	9	**14/25**
dynamic	WEL	word+chunk+entity	Fusion, Out, Catch-c	6	**1/36**

*Note that B-*c *indicates the four specific tags of B-PER, B-LOC, B-ORG and B-MISC. I-*c*, E-*c*, S-*c *and Catch-*c *are used the same way.*

The IOB/IOBES-divided Format is a variant of the *IOB/IOBES-original* format, by separating the positional information from the categorical information. In the *IOB/IOBES-divided* format, the original tags are divided into two sets: one set for the NE positional information ({B, I, O} or {B, I, O, E, S}) and the other set for the NE categorical information ({PER, LOC, ORG, MISC}), as shown in Table 1. The categorical labels are no longer assigned to a single word but rather to the whole entity. The entity's category will be judged only once by labeling the entity as a whole, disregarding its length. It is notable that only one hard constraint (I after O) is found in the *IOB-divided* format and 14 in the *IOBES-divided* format.

The WEL-dynamic Format is a whole-entity-labeling format that is dynamic. In this format, three actions (*Fusion*, *Out* and *Catch-c*) are used to represent the operations on the labeling object. *Fusion* means that the object is an uncompleted part of an entity and should be fused with the next word. *Catch* is used for the objects that are judged to be an entire entity, and we will classify it into a specific category with a suffix *c*. *Out* is used when a common word outside of an entity is presented. For example, with a three-word entity $[e_1, e_2, e_3]$, the first word e_1 will be labeled by *Fusion*, and then together with e_2, *Fusion* will be assigned to the chunk $[e_1, e_2]$, and finally, the entity $[e_1, e_2, e_3]$ will be caught in category *c* by *Catch-c*. In the *WEL-dynamic* format, only 6 tags are needed. Out of a total of 36 two-tag permutations, the only constraint is *Out* after *Fusion*. In fact, this permutation is also explainable to some extent as the NER model finds the chunk that is not likely to be an entity after fusing its words.

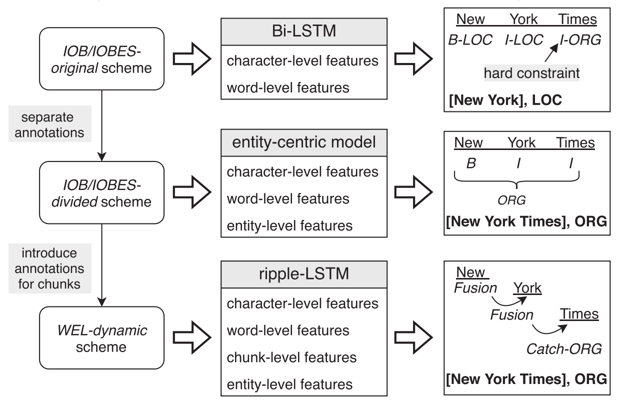


Fig. 1. Dynamic NER models under unconstrained tagging schemes.

4 Dynamic NER Models

As shown in Fig. 1, we propose two dynamic models, ECM and R-LSTM, to process the input under the *IOB/IOBES-divided* format and the *WEL-dynamic* format, respectively. The basic Bi-LSTM model, which is widely used in many tag prediction tasks, is the common foundation of our dynamic models and will be introduced as the premise of tag prediction in Section 4.1. The details of ECM and R-LSTM are presented in Sections 4.2 and 4.3. Together with the word-char embedding, the entity representation strategy is summarized in Section 4.4.

4.1 Basic Premise of Tag Prediction

The main architectures of most tag prediction models are based on LSTM units [27] which are implemented as follows:

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

where σ is the elementwise sigmoid function and $\circ$ is the elementwise product. The gates $\mathbf{f}, \mathbf{i}$ and $\mathbf{o}$ represent the forget gate, input gate and output gate, respectively. $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are the parameters tuned during the training process. Given a sentence containing n words

$$W = (w_1, w_2, \ldots, w_n), \tag{2}$$

each word is properly represented by its a fixed-dimension embedding x_i^w. After being fed into a Bi-LSTM, the hidden states for each word can be obtained from

$$\overrightarrow{\mathbf{h}}, \overleftarrow{\mathbf{h}} = \mathbf{BiLSTM}([\mathbf{x}_1^w, \mathbf{x}_2^w, \ldots, \mathbf{x}_n^w]). \tag{3}$$

The ith word together with its contexts are represented by the concatenation of the ith index of the hidden states

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}]. \tag{4}$$

Then, $\mathbf{h}_i$ is projected by an output layer to obtain the score for each tag

$$S_i = \mathbf{W}\mathbf{h}_i + \mathbf{b}, \tag{5}$$

in which $\mathbf{W}$ is the projection matrix and $\mathbf{b}$ is the bias. We consider $S_i(y)$ to be the score of tag y; then, we can obtain its probability using a *softmax* function

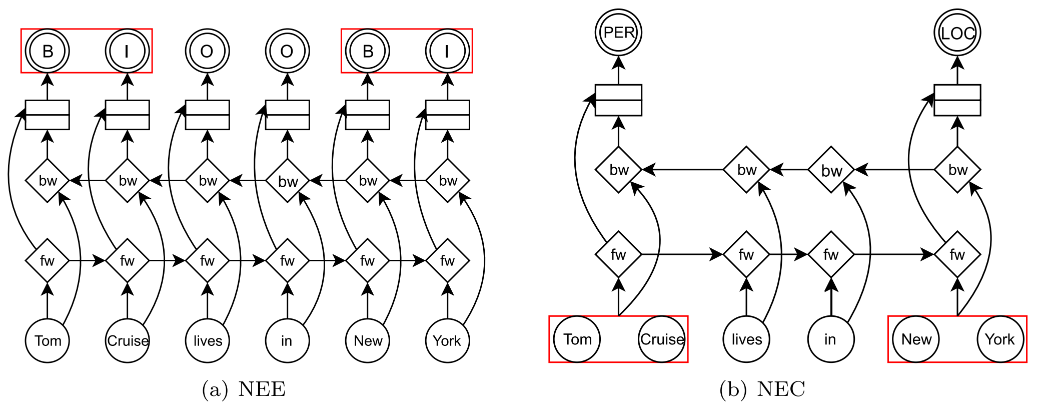


Fig. 2. Entity-centric model.

$$\mathbf{P}_i(y) = \frac{e^{s_i(y)}}{\sum_{\hat{y}\in\mathbf{Y}} e^{s_i(\hat{y})}}. \quad (6)$$

We will maximize the log-probability of each accurate tag during training

$$\log(\mathbf{P}_i(y)) = S_i(y) - \log\left(\sum_{\hat{y}\in\mathbf{Y}} e^{s_i(\hat{y})}\right), \quad (7)$$

where $\mathbf{Y}$ denotes the set of tags. During decoding, we will choose the most probable tag y^* as

$$y^* = \arg \underset{\hat{y}\in\mathbf{Y}}{max}\, \mathbf{P}_i(\hat{y}). \quad (8)$$

For the NER task under the *IOB/IOBES-original* format, this basic Bi-LSTM model is convenient for predicting the tags for each word, and it is the fundamental structure for our dynamic models.

4.2 Entity-Centric Model

For NER under the *IOB/IOBES-divided* format, as shown in Fig. 2, ECM consists of a named entity extractor (NEE) and named entity classifier (NEC) based on a simple LSTM architecture. The extracted entities, which are in the red frame, are fed into the NEC and classified into specific categories. In ECM, the positional feature and the categorical feature are absolutely decoupled. The NEE extracts NEs by labeling each word with tag {B, I, O} or {B, I, O, E, S}. The NEC classifies the extracted candidate entities into specific categories in {LOC, PER, ORG, MISC}. During the process, entity-level information is introduced to the NER system.

Named Entity Extractor. The input for NEE remains at the word level, and the basic model above can be transferred to fit this situation directly. In Fig. 2, the sentence "Tom Cruise lives in New York" is given as an example. For each single word, NEE processes it as a sequence-labeling task and labels it with the segment tags {B, I, O}. The sentence will be segmented by these predictions, and the candidate entities will be extracted according to their boundaries. For example, the entities "Tom Cruise" and "New York" will be extracted as candidate entities to be classified by the classifier. In contrast to the LSTM-CRF model [7], [8], the CRF layer has very little influence on NEE because of our unconstrained tagging schemes.

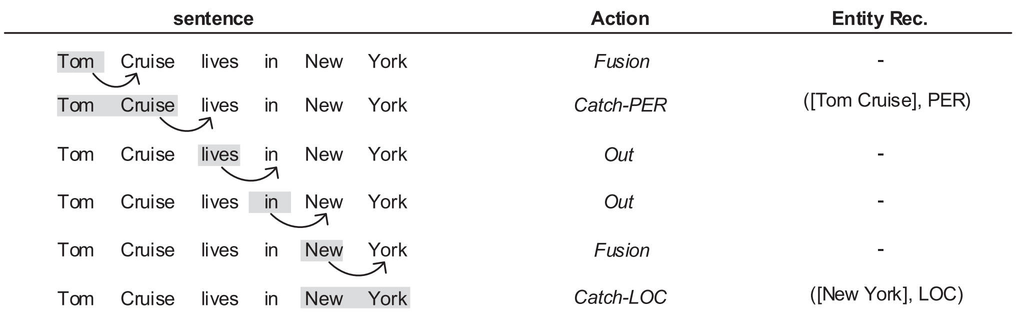
Named Entity Classifier. The input for NEE is a segmented sentence consisting of words and entities, and the output is the category of entities. The key point in this phase is how to represent the candidate entities in a compatible way with words. The entity-character embedding method, which inserts the entity-level semantic meaning into the NER system, will be described in Section 4.4.2. Instead of by piecing together the judgments on each single word, entities are classified according to the definite information they inherently have. The remainder of the work is to compute the probability of each category, which is similar in the basic model. As shown in Fig. 2b, the candidate entities are classified into a specific category by NEC. The NER is completed under the *IOB/IOBES-divided* format.

4.3 Ripple-LSTM Model

With the help of the *WEL-dynamic* format, a segment-level NN is constructed by integrating the entity extraction phase and classification phase into one. With a dynamic LSTM sliding across a fixed LSTM, it is referred to as the ripple LSTM and R-LSTM for simplicity. At each time step, the sentence is segmented into three parts by the detection window. A series of actions defined by the *WEL-dynamic* format are used in the transit of the detection window. The transition algorithm is presented in Section 4.3.1, and the adapted neural architecture is shown in Section 4.3.2.

4.3.1 Transition Algorithm for the Detection Window

As shown in Fig. 3, a detection window (in gray) will segment a sentence into three parts: the left context, the chunk in the window and the right context, and the arrow indicates its transition process. For the content in the detection window, we will assign an action defined in the *WEL-dynamic* format. The primary function of the action is to judge the property of the content and then decide how the window transits next. For example, given the sentence in Fig. 3, the word "Tom" will be inspected by the window first. The action *Fusion* acknowledges the content may be a part of an entity, but it is not time to judge its category because it is incomplete. Fusing together with the next word "Cruise", the chunk "Tom Cruise" will be encompassed by the detection window. Clearly, "Tom Cruise" is a name for a person and should be labeled by the action *Catch-PER*, which means that the content in the window is



sentence	Action	Entity Rec.
Tom Cruise lives in New York	*Fusion*	-
Tom Cruise lives in New York	*Catch-PER*	([Tom Cruise], PER)
Tom Cruise lives in New York	*Out*	-
Tom Cruise lives in New York	*Out*	-
Tom Cruise lives in New York	*Fusion*	-
Tom Cruise lives in New York	*Catch-LOC*	([New York], LOC)

Fig. 3. Actions taken on *Tom Cruise lives in New York*.

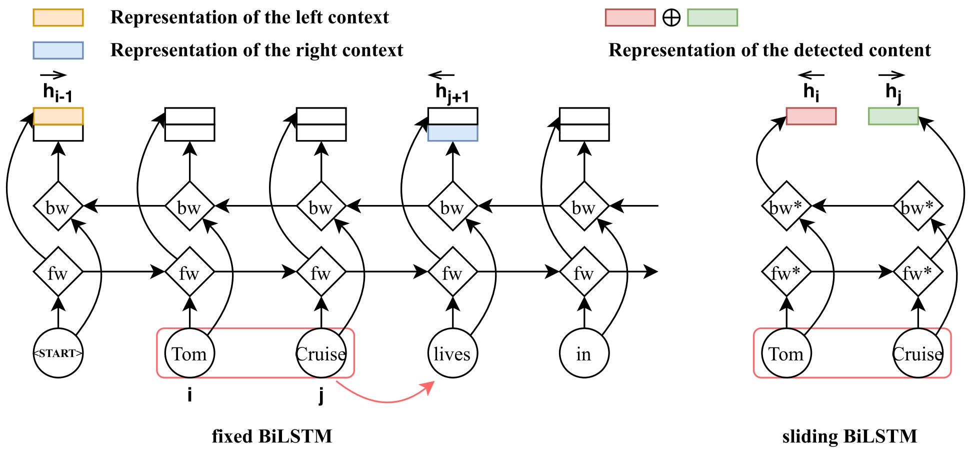

Representation of the left context
Representation of the right context
Representation of the detected content
$\overrightarrow{h}_{i-1}$
$\overleftarrow{h}_{j+1}$
$\overleftarrow{h}_i$
$\overrightarrow{h}_j$
bw
fw
bw*
fw*
<START>
Tom
Cruise
lives
in
i
j
fixed BiLSTM
sliding BiLSTM

Fig. 4. Architecture of the R-LSTM.

an entire entity in the category of *Person*. Then, the detection window will slide to the next word "lives". After the content is labeled by *Out*, the window will slide to the next word.

Although this transition-based labeling format is inspired by the *Shift-Reduce* method introduced in [8], our *WEL-dynamic* format is completely different from their method in these aspects. First, the labeling target, which is the content in the detection window, is definite in our format, while *Shift-Reduce* can be considered as actions at the sentence level, not tags for any unique object. Second, in our format, the right side of the window will slide forward by a fixed length of one word at each step; thus, only the n step is needed when predicting an n-word sentence, whereas up to $2n$ steps can occur in the *Shift-Reduce* method.

4.3.2 Architecture of R-LSTM

To present how the R-LSTM works in the dynamic scene, we first provide a formal description of it in a static way. For a sentence W (Eq. (2)) segmented by the detection window from the ith to the the jth word ($j \geq i$), each word is represented by a fixed-dimension vector x_i^w. The left context, $[w_1, w_2, \ldots, w_{i-1}]$, is fed into a forward LSTM as an embedding, and the hidden states are obtained

$$\overrightarrow{\mathbf{h}} = \overrightarrow{LSTM}([\mathbf{x}_1^w, \ldots, \mathbf{x}_{i-1}^w]). \tag{9}$$

For the right context, $[w_{i+l}, \ldots, w_n]$, it is fed into a backward LSTM in a reverse sequence, and the hidden states are obtained by

$$\overleftarrow{\mathbf{h}} = \overleftarrow{LSTM}([\mathbf{x}_{j+1}^w, \ldots, \mathbf{x}_n^w]). \tag{10}$$

The content in the detection window is placed into another Bi-LSTM network, which is similar to Eq. (3), obtaining the hidden states $[\overrightarrow{\mathbf{h}}', \overleftarrow{\mathbf{h}}']$. Finally, the chunk together with its contexts can be represented by a concatenation of four unique hidden states

$$\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_{i-1}; \overrightarrow{\mathbf{h}}'_j; \overleftarrow{\mathbf{h}}'_i; \overleftarrow{\mathbf{h}}_{j+1}], \tag{11}$$

in which the chunk is uniquely identified by its ending word's index j since the right side of the detection window propagates by one word after each step.

In Eq. (11), the process of obtaining the hidden states of each chunk's context is redundant. In other words, we will repeat the calculation of Eqs. (9) and (10) when the detection window slides. To make the R-LSTM work more effectively, we need to catch the hidden features in a dynamic way. An automatic construction of the representation algorithm is proposed to reduce redundant computations in the NN, as shown in Algorithm 1. Instead of repeating Eqs. (9) and (10) for each chunk, all the possibly used representations of the contexts are calculated by feeding the sentence into a fixed Bi-LSTM network only once. A sliding Bi-LSTM is prepared for the chunk in the detecting window. Because most chunks only consist of one or two words, the overhead for the sliding Bi-LSTM can remain at a low level.

Fig. 4 is the concise architecture of the R-LSTM and attempts to present the dynamic process to obtain the hidden states $\mathbf{h}_j$. The context representation is computed only once by the fixed Bi-LSTM, and then the window slides to the next chunk. The entity-level information is conveyed by the sliding Bi-LSTM to make the model judge more precisely. The final probability over actions is obtained by Eq. (6). A dropout layer is used above the representation to ensure that each part is fully trained. In the prediction period, the maximum probability of the action is greedily chosen left-to-right until the end of the sentence.

Algorithm 1. Automatic Construction of the Representation

```
input: The input sentence $W = (w_1, w_2, \ldots, w_n)$; The set of positions of the detection window sliding on $W$, denoted as $\mathcal{A}$, in which each position is signed by the index of its beginning and ending word, $(i, j), j \geq i$;
Output: The representation h for the content in the detection window and its contexts.
1: $[\overrightarrow{\mathbf{h}}, \overleftarrow{\mathbf{h}}] = \mathbf{BiLSTM}([x_1^w, x_2^w, \ldots, x_n^w])$
2: for $(i, j)$ in $\mathcal{A}$ do
3:     $[\overrightarrow{\mathbf{h}}', \overleftarrow{\mathbf{h}}'] = \mathbf{BiLSTM}'([x_i^w, x_2^w, \ldots, x_j^w])$
4:     $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_{i-1}; \overrightarrow{\mathbf{h}}'_j; \overleftarrow{\mathbf{h}}'_i; \overleftarrow{\mathbf{h}}_{j+1}]$
5: end for
   return h
```

4.4 Input Embedding

The input of dynamic models relates to the semantic representation of words and entities. For both dynamic models, word embedding is a necessary part. For ECM, an entity-character embedding method is proposed to represent entities compatibly.

4.4.1 *Word-Character Embedding*

For the ith word w_i consisting of m characters $[w_i^1, w_i^2, \ldots, w_i^m]$, it can be represented by its word embedding $\mathbf{e}(w_i)$

$$\mathbf{x}_{w_i} = \mathbf{e}(w_i), \tag{12}$$

where $\mathbf{e}$ is the lookup table for word embedding. The orthographic features are represented by a Bi-LSTM (Eq. (1)), in which the character sequence in w_i is fed into two parallel LSTMs in opposite directions. The hidden state $\overrightarrow{\mathbf{h}_{w_i}^m}$ obtained by the rightmost cell in the forward $\overrightarrow{\mathbf{LSTM}}$, together with $\overleftarrow{\mathbf{h}_{w_i}^1}$ obtained by the leftmost cell in the backward $\overleftarrow{\mathbf{LSTM}}$, are concatenated as a character-level representation

$$\mathbf{x}_{w_i}^c = [\overrightarrow{\mathbf{h}_{w_i}^m}; \overleftarrow{\mathbf{h}_{w_i}^1}]. \tag{13}$$

The word-character embedding is the combination of Eqs. (12) and (13)

$$\mathbf{x}_i^w = [\mathbf{x}_{w_i}; \mathbf{x}_{w_i}^c]. \tag{14}$$

4.4.2 *Entity-Character Embedding*

Embedding an entity is much more difficult than embedding a word. In the classifier, the dimension of the entity's representation must be the same as that of the normal words. We construct a structure that can generate an entity-character embedding for any entity with a random length.

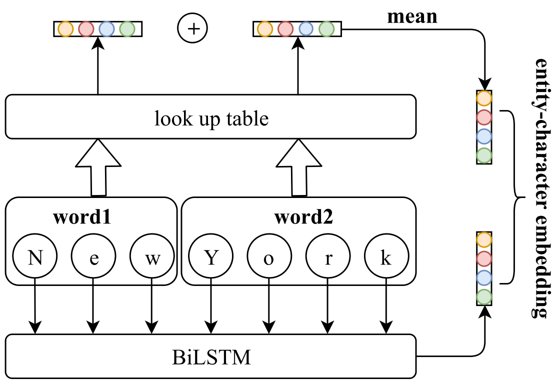


Fig. 5. Example of entity-character embedding.

As shown in Fig. 5, the character-level embedding is obtained by feeding the characters into a Bi-LSTM NN, and the entity "New York" is an example to be processed. We will obtain its embedding in the following ways:

- *Entity Embedding.* Embeddings for words can be obtained from the lookup table. The mean value of these embeddings of words can be calculated as the final embedding of the entity. Thus, the entity embedding has the same dimension as pretrained vectors. For the words out of vocabulary (OOV) in entities, they are initialized as zeros.
- *Character-level embedding.* Inspired by the method of character embedding of words, we ignore the white space between the words of an entity and take the entity as an integral word. For example, in Fig. 5, the entity "New York" is treated as "NewYork". The character embeddings are obtained from a lookup table that is randomly initialized for every character. These embeddings are fed into forward and backward LSTMs with direct and reverse orders, respectively. The final embedding is the concatenation of the result from the forward and backward LSTM networks. Character-level embedding is able to capture the orthographic features of entity chunks.

An entity-character embedding is generated by the concatenation of an entity embedding and its character embedding. In this paper, the dimensions for forward and backward character LSTMs are both set as 100; thus, the character embedding for each entity has a dimension of 200. A 300-dimensional pretrained word embedding is used, which results in a 500-dimensional representation for every entity chunk after concatenation.

5 Experiments

We have performed an extensive performance evaluation of our proposed models over a real CoNLL dataset. Our goals include the following:

- Analyzing the effectiveness of the functional modules of the architecture;
- Evaluating the properties of the extractor and the classifier under different segmentation formalisms, evaluating the performance of the R-LSTM model and ECM and comparing with the state-of-the-art models;
- Assessing the effect of hard constraints.

TABLE 2
The Number of Tokens (Entities) in CoNLL2003 and CoNLL2002 Dataset

Datasets	CoNLL2003		CoNLL2002	
	English	German	Dutch	Spanish
train	204,567 (23,499)	207,484 (11,851)	202,931 (13,344)	264,715 (18,797)
dev	51,578 (5,942)	51,654 (2,867)	37,761 (2,616)	52,923 (4,351)
test	46,666 (5,648)	52,098 (3,673)	68,994 (3,941)	51,533 (3,558)

5.1 Experimental Setup

We experiment on the four well-known NER datasets: English and German from CoNLL2003 [2] and Spanish and Dutch from CoNLL2002 [1]. All of these datasets contain four types of entities, persons, locations, organizations and miscellaneous entities, that do not belong to any previous types. The details are as follows:

- *CoNLL2002*. The CoNLL2002 [1] consists of language-independent named entities of Spanish and Dutch. The data consist of three files per language: one training file and two test files: testa (development set) and testb (test set). The Spanish data are a collection of news wire articles made available by the Spanish EFE News Agency, and the Dutch data consist of four editions of the Belgian newspaper "De Morgen" from 2000.
- *CoNLL2003*. The CoNLL2003 [2] consists of independently named entities of English and German. The English data are a collection of news wire articles from the Reuters Corpus, and the German data are a collection of articles from Frankfurter Rundschau.

The statistics of these datasets are shown in Table 2. The size of the development sets and test sets is approximately one-fourth of the training sets. The NER models should handle a tremendous amount of raw text in the training set, which is up to 200,000 labeled tokens, to learn the features of entities and nonentities. Our models are trained on the training dataset and selected according to their performance on the development dataset. Based on the datasets, pretrained word embedding is executed. To obtain pretrained word embedding, we apply the *Wikipedia2vec* method on the Wikipedia dump with approximately 2 billion tokens and 73 million anchors, as done in [28]. For the other three languages, we use the ready-made *Wikipedia2vec*-based pretrained word embeddings posted on the Internet, which can be downloaded openly.[1] These pretrained embeddings will be fine-tuned during the training process.

Furthermore, before all the tokens are fed into NNs, it is essential to preprocess the raw massive data [29], [30]. First, all digits are replaced by a uniform symbol, which is taken as a single word in a sentence. Second, OOV words are replaced by *UNK* and initialized to zeros in the same dimension with pretrained word embeddings. OOVs will be fine-tuned during the training process. For these OOVs, character embeddings play an important role in disambiguating them from each other. Third, all sentences in the training set are shuffled and minibatched before each training epoch, which will ensure that the model learns better. Finally, for the classifier in the ECM, the words before and after the entity will also be truncated. We choose the size to be 50 for most sentences in the dataset to be shorter than 100 words. This operation will accelerate the training process.

1. https://wikipedia2vec.github.io/wikipedia2vec/

For evaluation, the F1 measurement metrics are used to evaluate the final result of our systems, as described in [1] and [2]. The F1 score can be calculated as

$$\frac{2}{f}=\frac{1}{p}+\frac{1}{r}, \tag{15}$$

in which f, p, r represent the F1 score, precision and recall rate, respectively. Similarly, the intermediate result produced by the NE extractor can also be evaluated by the F1 score. The performance of the classifier is evaluated by its accuracy, which is the ratio of the number of correctly classified entities to the total number of entities.

After pretrained word embedding and token preprocessing, the training is performed based on the backpropagation algorithm. As reported by [8], stochastic gradient descent (SGD) with gradient clipping can obtain a better performing model. However, the convergence speed is very slow compared to enhanced methods, such as Adam [31]. Our models are trained under the Adam method, and the learning rate is set to 0.001. To avoid coadaptations of the concatenation-styled representation, we set a dropout layer above the final embedding layer just before feeding them into Bi-LSTM networks. During the training process, the dropout rate is set to 0.5. The training will be stopped early if there are no improvements in the development set in 5 epochs.

In the R-LSTM model, the dimensions for the fixed Bi-LSTM and sliding Bi-LSTM are set to 300. Consequently, the representation of the detected content with its contexts is represented as a 1200-dimensional vector. The proportion of the representation for the detected content is equal to that of the context, although the detected content always consists of very short words in a long sentence. The setting is reasonable because the actions mainly act on the content in the detection window. In the ECM, the Bi-LSTM's dimensions are set to 300, both in the extractor and the classifier. Fine-tuning these dimensions has little influence on the final results. All of our experiments are completed on NVIDIA Tesla M40 GPU with the TensorFlow framework.

5.2 Analysis on Functional Modules

In this section, we analyze the functional modules, such as character embedding and pretrained embedding in NEE, NEC and R-LSTM. In our opinion, it is necessary because with a fundamental change in the NER pattern under unconstrained schemes, the effects of each functional part become uncertain. In Fig. 6, all the models are trained on the training set and tested on the development set. The effectiveness of character embedding and pretrained word embedding are tested on the three models. For NEE, the effect of the CRF layer will also be tested, and IOB is utilized as its segmentation tags.

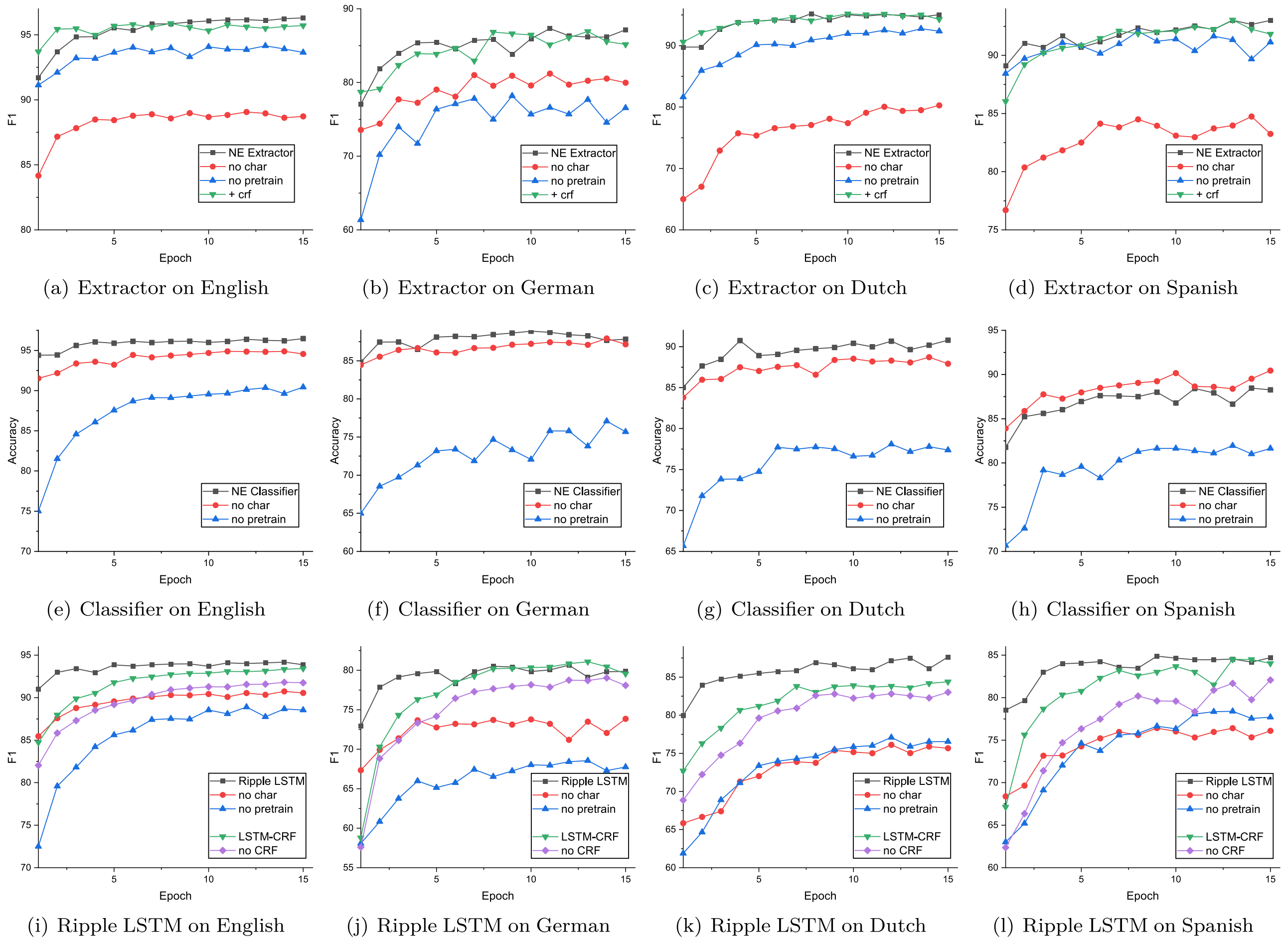


Fig. 6. Effectiveness of each functional part in our models: NEE (a-d), NEC (e-h) and R-LSTM (i-l).

The effectiveness of the CRF layer in the extractor is shown in Figs. 6a, 6b, 6c, and 6d. The line in black (NEE) is almost at the same level as the line in green (NEE with CRF), which means that NEE can perform well without the help of the CRF layer. This is because our models separate the segmentation tags with the categorical information, and NEE can complete its work under an unconstrained labeling format. To make the result more convincing, the LSTM-CRF model is tested under the constrained *IOB-original* format and shown in Figs. 6i, 6j, 6k, and 6l. In all the four languages, the LSTM-CRF model suffers from an obvious performance degradation without the CRF layer, which does not happen under the unconstrained schemes.

Character embedding is an effective way to capture the orthographic and morphological features of words, whether using a CNN [9] or a Bi-LSTM [8]. NER models for western languages benefit greatly because the words of an entity are generally capitalized and the affix of words may contain some important information. Consequently, the NEE (red line in Figs. 6a, 6b, 6c, and 6d) and R-LSTM (red line in Figs. 6i, 6j, 6k, and 6l) have very poor performances without character embedding. Unlike the extractor, the NEC can benefit from character embedding to a very small extent, as shown in Figs. 6e, 6f, 6g, and 6h. To classify an entity, the semantic meaning of the word plays a more important role than its morphology. In Fig. 6h, the NEC without character embedding even performs better in Spanish. This abnormal phenomenon may come from the insufficient training process of NEC.

The pretrained word embedding maps the words into a semantic vector space that is convenient to be fed into NNs. Using pretrained word embedding makes the NN outperform the randomly initialized network. The word embedding model will assign to each word a specific dimension vector trained on unlabeled texts. The words with similar semantics tend to be closer to each other in the embedding space. This property makes the pretrained word embedding quite useful in NEC. As shown in Figs. 6e, 6f, 6g, and 6h, without external information from a pretrained embedding, the accuracy of NEC decreases to a relatively lower level (line in blue). In Figs. 6i, 6j, 6k, and 6l, the R-LSTM's performance is seriously degraded without pretrained word embedding. However, in Figs. 6a, 6b, 6c, and 6d, the performance of NEE is not greatly affected without word embedding (line in blue), except for that in German. This result reveals that the extractor can perform well only with the orthographic and morphological features in words.

5.3 Performance Evaluation

This section first presents the performances of the extractor and the classifier. Then, it gives the NER results of ECM and R-LSTM. Finally, it provides the comparison with other models.

TABLE 3
Performance of NEE on Four Different Languages

Language	Format	Precision	Recall	F1
English	-iob	94.58	95.17	**94.87**
	-iobes	94.91	94.74	94.82
German	-iob	87.52	80.97	84.12
	-iobes	89.46	80.21	**84.58**
Dutch	-iob	94.99	94.37	94.68
	-iobes	96.02	94.37	**95.19**
Spanish	-iob	93.37	93.81	93.59
	-iobes	95.45	94.29	**94.87**

5.3.1 Performances of NEE and NEC

NEE is constructed based on the Bi-LSTM without a CRF layer. In this experiment, two kinds of segmentation strategies for tags are evaluated. The metric for NEE is the F1 score (Eq. (15)) on the test dataset. As shown in Table 3, for each language, two different segmentation formats (*IOB* and *IOBES*) are evaluated for the NEE results. NEE achieves extremely high F1 scores of approximately 95 in English, Dutch and Spanish. In German, the F1 score is relatively lower than in the other languages, which means that extracting NEs from German is not as easy as from other languages. Regarding the aspect of the labeling format, we find that using *IOBES* as segment tags is generally better than using *IOB*. This result indicates that *IOBES* is more expressive than *IOB* in the segmentation problem. We will explore the reason why *IOBES* is a better choice even though there are more hard constraints in *IOBES* (in Section 5.4).

We will feed all sentences containing entities into NEC and evaluate the classification accuracy. Entities are classified into four different categories. The results from the four languages are presented in Table 4. The accuracy of NEC in English surpasses the accuracy in all other languages and reaches 94.64. With the help of an external knowledge base, our classifier's property may be further improved. However, this is not the key point in this paper, and we will leave it for future research.

5.3.2 Performance of NER

In this section, we evaluate our models on different languages and compare the performances with those of recent state-of-the-art works. Our models are trained only on the training sets given by the corpus and using the early stopping strategy to choose the best model according to their performance on the development sets. For fair comparisons, the methods are classified into three categories, and NER results on CoNLL2003 and CoNLL2002 in four different languages, EN (English), GE (German), DU (Dutch) and SP (Spanish), are shown in Table 5. The existing models are classified into three categories according to the style of feature extraction. The tagging schemes for these models are listed, and N/A indicates that the scheme is unknown according to the corresponding paper. Additionally, the symbol * is used to indicate the models that use external knowledge, such as gazetteers. Although no external language-specific knowledge is used in our models, our models can achieve comparable results.

TABLE 4
Performance of NEC on Four Different Languages

Language	Accuracy
English	94.64
German	89.11
Dutch	90.38
Spanish	91.43

- *Auto-feature-inferring neural network models* are those that use NNs to extract features from a raw corpus without human intervention. Our ECM and R-LSTM models belong to this class. With these methods, Kuru *et al.* [35] and Gillick *et al.* [36] completed NER tasks purely based on characters, and their character-based methods have shown great generalizability but with relatively lower NER results. In the CoNLL2003 English set, Sato *et al.* [25] utilized a segment-level NN model and achieved the best score. The pipeline is very complicated in the model: using a word-level model to generate segment lattices, using SemiCRF at the segment level to obtain the category of the lattices and finally judging the results by a linear-chain CRF at the word level. The process is similar to the extraction and classification phase in our ECM, but ECM is more concise, relying on no CRF layers. In the other three languages, Yadav *et al.* [19] obtained state-of-the-art results by a combination of word-character embedding with affix information. In our models, entity-level information is instead input into the NN. The R-LSTM model can obtain comparable results and even better results in German and Dutch.
- *Neural networks with handcrafted features* are those neural NER models that use human-defined features to augment the performance. Although the F1 score can reach 91.62 in English with the method of Chiu and Nichols [9], many types of handcrafted features (such as POS tag features, capitalization features, lexicons and other features) are employed. It is unfortunate that no experimental NER results are presented for the remaining languages. In our opinion, this may be because it is difficult to generalize these features into other languages. Our models only exploit NNs without any external knowledge. The experimental results show that our models can be applied to different languages with robust performance.
- *Feature-engineered machine learning systems* rely heavily on handcrafted features. NER results are relatively lacking in early research without sufficient external resources, such as the results by Carreras *et al.* [16] on CoNLL2002 and Florian *et al.* [38] on CoNLL2003. With a powerful external knowledge base extracted from Wikidata, Luo *et al.* transferred the NER problem into entity linking and obtained a superb F1 score of 91.20 in English. Agerri and Rigau [18] promoted a common set of features for different languages and tested on five different languages. Although it performs well in English, our models outperform it in the other three languages.

TABLE 5
NER Results on Four Different Languages

Auto-feature-inferring neural network models	Tagging scheme	EN	GE	DU	SP
Arora *et al.* (2019) [14]	category	90.76	-	-	-
Gregoric *et al.* (2018) [32]	N/A	91.48	-	-	-
Yadav *et al.* (2018) [19]	IOB	90.86	**79.01**	**87.54**	**87.26**
Sato *et al.* (2017) [25]*	IOBES for segments	**91.55**	-	-	-
Yang *et al.* (2017) [33]*	N/A	91.26	-	-	-
Ma and Hovy (2016) [34]	IOBES	91.21	-	-	-
Lample *et al.* (2016) LSTM-CRF [8]	IOBES	90.94	78.76	81.74	85.75
Lample *et al.* (2016) S-LSTM [8]	Shift-Reduce	90.33	75.66	79.88	83.93
Kuru *et al.* (2016) [35]	tags for char	84.52	70.12	79.36	82.18
Gillick *et al.* (2015) - BTS [36]	span annotation	86.50	76.22	82.84	82.95
Santos and Guimarães (2015) [37]	IOB	-	-	-	82.21
Neural network models with handcrafted features					
Chiu and Nichols (2015) [9]*	IOBES	**91.62**	-	-	-
Huang *et al.* (2015) [7]*	IOB	90.10	-	-	-
Collobert *et al.* (2011) [6]*	IOBES	89.59	-	-	-
Feature-engineered machine learning systems					
Agerri and Rigau (2016) - Perception [18]*	IOBES	**91.36**	**76.42**	**85.04**	**84.16**
Luo *et al.* (2015) - SemiCRF [15]*	N/A	91.20	-	-	-
Florian *et al.* (2003) [38]*	IOB	88.76	72.41	-	-
Carreras *et al.* (2002) [16]*	IOB	-	-	77.05	81.39
Our models with baselines					
Bi-LSTM	IOB	89.06	76.78	85.30	84.03
-	IOBES	89.71	76.98	86.09	85.41
ECM	IOB-div	**90.80**	77.13	86.51	85.71
-	IOBES-div	90.63	77.60	87.20	86.87
Ripple LSTM	WEL-dyn	90.67	**79.08**	**87.65**	**86.90**
Ripple LSTM + affix	WEL-dyn	**91.05**	**79.40**	**87.88**	**87.41**

In the experimental results, the basic Bi-LSTM model described in Section 4.1 is tested as the baseline model under the *IOB-original* and *IOBES-original* formats. ECM is operated with the *IOB-divided* and *IOBES-divided* formats. Without the CRF layer, our ECM surpasses the baseline model to a great extent because of the unconstrained tagging scheme. As in NEE, the NER results are better under *IOBES* than under *IOB*, regardless of the original or divided format. This result confirms that the *IOBES* format is more expressive than the *IOB* format. The tagging format for R-LSTM is the *WEL-dynamic* format, which is inspired by the Shift-Reduce format in [8]. Compared with their S-LSTM, R-LSTM exploits a lean structure and exceeds the model by 3.42 in German, 7.77 in Dutch and 2.97 in Spanish. Compared with ECM, it is a one-phase model with even better performance. This result reveals that our R-LSTM is a solid model and that the *WEL-dynamic* format is a concise but expressive format. To further optimize the model, the affix feature introduced by Yadav *et al.* [19] is integrated into the input embedding as an additional 100-dimensional vector. The F1 scores on the four languages are all improved and surpass the best scores obtained by Yadav *et al.* [19] under the IOB format. This is due to the entity-level information in R-LSTM under unconstrained schemes.

The efficiency is also an important aspect, especially when deploying these models in practical applications. In Fig. 7, the models are tested under the same experimental conditions on the English training set. The time overhead is obtained by recording the training time every 5 epochs in the training phase. In order to be fair, the dimensions of the input embeddings are set as 500 and the output of the LSTMs are set as 300. As expected, the naive Bi-LSTM model is faster than other complicated models, and approximately 2 times faster without the CRF layer in our experiments. In ECM, the time overhead, which is the sum of NEE's and NEC's cost, remains at the same level as that of the LSTM-CRF. R-LSTM models, realized in static and dynamic ways, are also tested. Optimized by Algorithm 1, the dynamic R-LSTM, which reduces

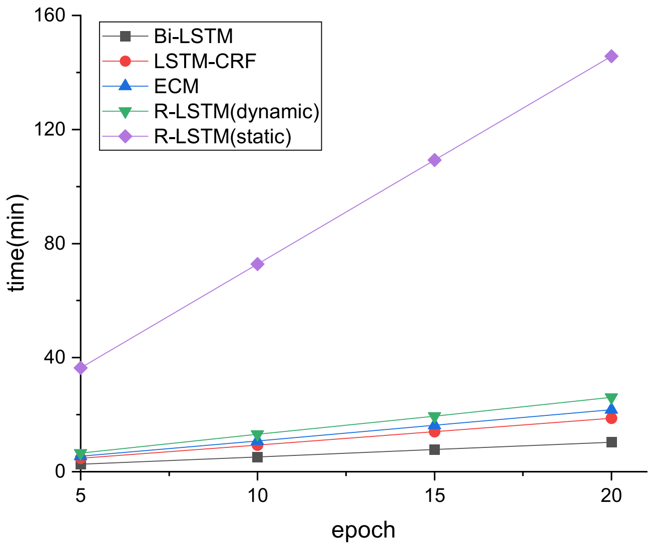


Fig. 7. Time overhead versus the training epoch.

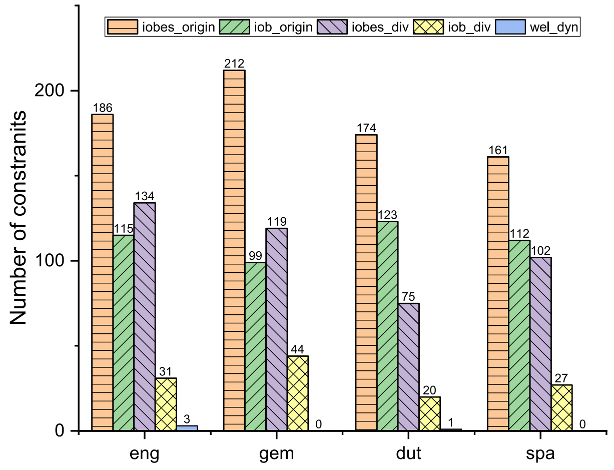


Fig. 8. Number of constraints under different tagging schemes.

the redundant representation process of the context, can perform considerably faster than the static one. Compared with the Bi-LSTM model, the extra cost in the dynamic R-LSTM may come from the additional computing cost of the sliding Bi-LSTM and the dynamic process. It is shown that both ECM and the dynamic R-LSTM model are efficient enough under the unconstrained schemes.

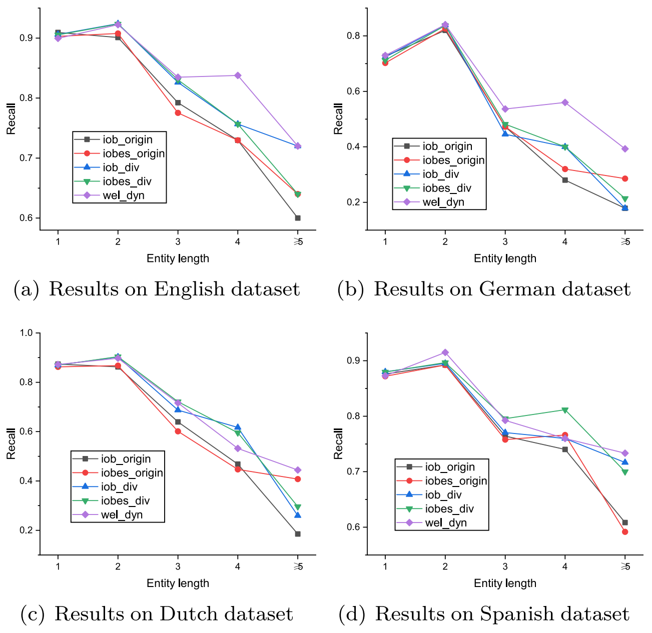
5.4 Assessment on Hard Constraints

In this section, we investigate the property of the prediction results under different tagging schemes to verify the motivation (Section 3.1) experimentally. For the *IOB/IOBES* format, the Bi-LSTM model is used to obtain the predictions for the tags. For the *IOB/IOBES-divided* format, the ECM model is used. All the results will be compared with R-LSTM's results in the aspect of hard constraints in the predictions and the recall rate of the NEs.

5.4.1 Hard Constraints in the Predicted Tags

We investigate the total number of hard constraints that occur in the prediction sequence under different tagging schemes. Under the *IOB/IOBES-original* format, the Bi-LSTM model is used to predict tags for each word. Their divided format is processed by our ECM to obtain the final results. The R-LSTM is the model utilized for the *dynamic* format. As shown in Fig. 8, the constraints are counted on the four languages. The number of hard constraints in the *IOB/IOBES-original* format is approximately two times larger than that in the *IOB/IOBES-divided* format because the constraints in categorical information are eliminated in divided formats. With fewer hard constraints in the prediction sequences, the ECM model performs much better than the Bi-LSTM model. However, irrespective of the original or divided results, we can obtain a better prediction result under the *IOBES* format with even more hard constraints in their predictions. This means that the hard constraint has a certain positive effect in NER tasks, which will be discussed later.

The hard constraints in the *dynamic* format with the R-LSTM model are almost 0 for all languages. It is a great property that indicates that the prediction process has a high degree of consistency. The only constraints in English and Dutch occur when the model fuses several words and then labels the chunk with action *Out*. We can only ignore this type



(a) Results on English dataset (b) Results on German dataset

(c) Results on Dutch dataset (d) Results on Spanish dataset

Fig. 9. Recall rate on 4 different languages against NE's length.

of fault when we decode the prediction sequences because the model does not provide any categorical information for the chunk. However, in the *IOB/IOBES* format, these faults from hard constraints are ignored manually when we decode the result sequence. For example, when a word is predicted as inside of an entity of Person and labeled as *I-PER*, and the former word is labeled as **O**, a hard constraint occurs in this situation. Instead of recognizing the word as an entity, we only ignore the ungrammatical part. If we adopt a radical strategy and accept the raw judgments from the models, the performance will decrease to some extent, especially under the *IOBES* format. Such a problem will not exist in our R-LSTM model under the unconstrained *WEL-dynamic* format.

As analyzed previously, the entity will suffer from hard-constrained error as its length increases because the model needs to judge n-times for an n-word entity. Any hard constraint in the prediction of these words will lead to the entity being unrecognized. Therefore, we investigate the recall rate against the length of the entity on the four different languages. As shown in Fig. 9, the models are tested under specific tagging schemes. For the entities in only one word, these models are comparable with respect to recall rate. However, when the entities' length is 2 or larger, our divided format and dynamic format can perform better than the original *IOB* or *IOBES* format. In particular, when the length is equal to 5 or more, our R-LSTM model with a dynamic tagging scheme can perform much better than any other NER style, which means that the model more easily recognizes longer entities. Notably, there is an entity with 14 words in it, and only under the *IOB-divided* and *WEL-dynamic* formats is it recognized. The declines in the recall rate are more gentle in the unconstrained tagging schemes than in the original ones. The line in red, which indicates the performance of the Bi-LSTM model under the *IOBES-original* format, performs better than the *IOB* format with a black line. This result once again demonstrates a better attribute of the *IOBES* format than the *IOB* format, although it contains many more constraints. Therefore, we will discuss the effect of hard constraints on NER tasks.

5.4.2 The Effect of Hard Constraints

In this section, we focus on the *IOB* and *IOBES* formats to investigate the effect of hard constraints. In the *IOBES* format, the entities are recognized more explicitly than in the *IOB* format. The model with the *IOBES* format needs to clarify the beginning and the ending of each entity. However, in the *IOB* format, only the beginning of the word should be specifically labeled. When a word is labeled as **B** in the *IOB* format, an entity must be extracted from this word, and the only difference is the terminal position of this entity. However, in the *IOBES* format, the ending position must be confirmed by tag **E**; otherwise, the probable entity will be ignored. A reconfirmation mechanism in the *IOBES* format will make the entity be more accurately recognized. Consequently, a more expressive tagging scheme *IOBES* format is better than the simple *IOB* format, regardless of their original or divided style. This is confirmed in the previous experimental results.

However, the constraints in NE's categorical information are harmful to the performance of NER systems because each entity should be judged many times with its category. It is more reasonable to specify the entity's boundary and then classify it only once as a whole. Our proposed formats in divided and dynamic styles realize it in different ways.

6 Conclusion

In this paper, dynamic entity-based NER models are proposed based on simple LSTM NNs. They are segment-level models in which entity-level features are represented and utilized explicitly. The F1 scores of NER results in the four languages show that our models are robust and perform competitively with state-of-the-art models that have handcrafted features and external knowledge bases. In particular, in R-LSTM, the scores on the German and Dutch test sets even surpass the previous best results. Furthermore, we derive two unconstrained tagging schemes in the divided and dynamic styles. Both of them have superb properties for NER labeling. With almost no hard constraints in the predicted tags, the tagging scheme with a dynamic style has shown considerable potential in NER tasks, which might be useful in other sequence tagging NLP scenarios such as sentence chunking. In future work, the entity-based models can be further optimized. For ECM, a more complicated structure and the interaction between NEE and NEC may make sense. Moreover, the embedding of the entity is obtained just by the mean of the embeddings of the words that constitute the entity. Thus, how to train the entity embedding model from a large unlabeled corpus and represent extracted entities directly is an important issue. Regarding other aspects, it is meaningful to investigate the application of entity-based models to other languages, areas and NLP tasks.

Acknowledgments

This work was supported in part by National Key R&D Program of China under Grants no. 2018YFB1404302, National Natural Science Foundation of China under Grants No.61672256 and Guandong Science and Technology Plan under Grants no. 2017B030305003.

References

[1] E. F. Tjong Kim Sang, "Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition," in *Proc. 6th Conf. Natural Lang. Learn.*, 2002, pp. 1–4.

[2] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. 7th Conf. Natural Lang. Learn.*, 2003, pp. 142–147.

[3] G. Zhou and J. Su, "Named entity recognition using an HMM-based chunk tagger," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 473–480.

[4] H. Isozaki and H. Kazawa, "Efficient support vector classifiers for named entity recognition," in *Proc. 19th Int. Conf. Comput. Linguistics*, 2002, pp. 1–7.

[5] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proc. 7th Conf. Natural Lang. Learn. HLT-NAACL*, 2003, pp. 188–191.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. 76, pp. 2493–2537, 2011.

[7] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, *arXiv:1508.01991*.

[8] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2016, pp. 260–270.

[9] J. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 357–370, 2016.

[10] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[12] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[13] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *Proc. 13th Conf. Comput. Natural Lang. Learn.*, 2009, pp. 147–155.

[14] R. Arora, C.-T. Tsai, K. Tsereteli, P. Kambadur, and Y. Yang, "A semi-Markov structured support vector machine model for high-precision named entity recognition," in *Proc. 57th Conf. Assoc. Comput. Linguistics*, 2019, pp. 5862–5866.

[15] G. Luo, X. Huang, C. Y. Lin, and Z. Nie, "Joint entity recognition and disambiguation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2015, pp. 879–888.

[16] X. Carreras, L. Marquez, and L. Padró, "Named entity extraction using AdaBoost," in *Proc. 6th Conf. Natural Lang. Learn.*, 2002, pp. 1–4.

[17] H. L. Chieu and H. T. Ng, "Named entity recognition: A maximum entropy approach using global information," in *Proc. 19th Int. Conf. Comput. Linguistics*, 2002, pp. 1–7.

[18] R. Agerri and G. Rigau, "Robust multilingual named entity recognition with shallow semi-supervised features," *Artif. Intell.*, vol. 238, pp. 63–82, 2016.

[19] V. Yadav, R. Sharp, and S. Bethard, "Deep affix features improve neural named entity recognizers," in *Proc. 7th Joint Conf. Lexical Comput. Semantics*, 2018, pp. 167–172.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2018.

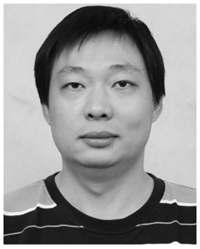
[21] K. J. Lee, Y. S. Hwang, and H. C. Rim, "Two-phase biomedical NE recognition based on SVMs," in *Proc. ACL Workshop Natural Lang. Process. Biomed.*, 2003, pp. 33–40.

[22] L. Li, R. Zhou, and D. Huang, "Two-phase biomedical named entity recognition using CRFs," *Comput. Biol. Chem.*, vol. 33, no. 4, pp. 334–338, 2009.

[23] S. Sarawagi and W. W. Cohen, "Semi-Markov conditional random fields for information extraction," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2005, pp. 1185–1192.

[24] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*.

[25] M. Sato, H. Shindo, I. Yamada, and Y. Matsumoto, “Segment-level neural conditional random fields for named entity recognition,” in *Proc. 8th Int. Joint Conf. Natural Lang. Process.*, 2017, pp. 97–102.
[26] L. A. Ramshaw and M. P. Marcus, “Text chunking using transformation-based learning,” in *Natural Language Processing Using Very Large Corpora*. Berlin, Germany: Springer, 1999, pp. 157–176.
[27] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[28] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Takefuji, “Wikipedia2Vec: An optimized tool for learning embeddings of words and entities from Wikipedia,” *arXiv*, vol. abs/1812.06280, 2018.
[29] W. Yang, X. Liu, L. Zhang, and L. T. Yang, “Big data real-time processing based on storm,” in *Proc. 12th IEEE Int. Conf. Trust Secur. Privacy Comput. Commun.*, 2013, pp. 1784–1787.
[30] X. Wang, L. T. Yang, H. Liu, and M. J. Deen, “A big data-as-a-service framework: State-of-the-art and perspectives,” *IEEE Trans. Big Data*, vol. 4, no. 3, pp. 325–340, Sep. 2018.
[31] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv*, vol. abs/1412.6980, 2014.
[32] A. Z. Gregoric, Y. Bachrach, and S. Coope, “Named entity recognition with parallel recurrent neural networks,” in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 69–74.
[33] Z. Yang, R. Salakhutdinov, and W. W. Cohen, “Transfer learning for sequence tagging with hierarchical recurrent networks,” *arXiv*, vol. abs/1703.06345, 2017.
[34] X. Ma and E. H. Hovy, “End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF,” in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1064–1074.
[35] O. Kuru, O. A. Can, and D. Yuret, “CharNER: Character-level named entity recognition,” in *Proc. 26th Int. Conf. Comput. Linguistics: Tech. Papers*, 2016, pp. 911–921.
[36] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya, “Multilingual language processing from bytes,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2016, pp. 1296–1306.
[37] C. dos Santos and V. Guimarães, “Boosting named entity recognition with neural character embeddings,” in *Proc. 5th Named Entity Workshop*, 2015, pp. 25–33.
[38] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang, “Named entity recognition through classifier combination,” in *Proc. 7th Conf. Natural Lang. Learn. HLT-NAACL*, 2003, pp. 168–171.


Feng Zhao received the MS and PhD degrees in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2006, respectively. He is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, China. He is a member of the ACM. He has published more than 60 papers at various conferences and journals, including SRDS, the *IEEE Transactions on Services Computing*, TBD, *Information Systems Journal*, *Journal of Information Science*, *Future Generation Computing Systems* and the *Communications in Mathematics and Applications*. His research interests include knowledge discovery, information retrieval, data mining, and natural language processing.

Xiangyu Gui received the BS degree in optical information science and technology from the Huazhong University of Science and Technology, Wuhan, China, in 2015. He is currently working toward the doctoral degree in computer science at the Huazhong University of Science and Technology, Wuhan, China. His research interests include information retrieval and knowledge graph.

Yafan Huang received the BS degree in computer science from Hunan University, Changsha, China, in 2018. He is currently working toward the master’s degree in computer science at the Huazhong University of Science and Technology, Wuhan, China. His research interests include information retrieval and knowledge graph.

Hai Jin (Fellow, IEEE) received the PhD degree in computer engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1994. He is currently a Cheung Kung scholars chair professor of computer science and engineering with the Huazhong University of Science and Technology (HUST) in China. He received German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany in 1996. He worked with the University of Hong Kong between 1998 and 2000, and as a visiting scholar with the University of Southern California between 1999 and 2000. He received the Excellent Youth Award from the National Science Foundation of China in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientist of National 973 Basic Research Program Project of Virtualization Technology of Computing System, and Cloud Security. He has coauthored 22 books and published more than 800 research papers. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security. He is a fellow of the CCF and a member of the ACM.

Laurence T. Yang received the BE degree in computer science and technology and the BSc degree in applied physics both from Tsinghua University, Beijing, China, and the PhD degree in computer science from the University of Victoria, Victoria, Canada. He is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, China, as well as with the Department of Computer Science, St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, big data. His research has been supported by the National Sciences and Engineering Research Council of Canada (NSERC) and the Canada Foundation for Innovation.