Enhanced Character Embedding for Chinese Short Text Entity Linking

Li Yang$^{1(\boxtimes)}$, Shijia E$^{2(\boxtimes)}$, and Yang Xiang$^{1(\boxtimes)}$

^{1} Tongji University, Shanghai 201804, P.R. China
^{2} Tencent, Shanghai, China
{li.yang,shxiangyang}@tongji.edu.cn
allene@tencent.com

Abstract. Entity Linking (EL) is the task of identifying concepts mentioned in a text and linking them to a given knowledge base. It includes two subtasks: Entity Recognition (ER) and Entity Disambiguation (ED). While entity linking in English long text has been well studied in previous works, Chinese short text entity linking remains a big challenge due to the lack of explicit word delimiters and rich context. In this paper, we propose an enhanced character embedding based neural approach, which explicitly encodes mention dictionary and mention position information into ER model and ED model respectively, to solve the CCKS2019 Task 2. Extensive experiments show that our proposed models significantly improve the performance of Chinese short text entity linking. Through two model ensemble strategies, our solution achieves a F1 score of 0.79266 on the final test data of CCKS2019 Task 2.

Keywords: Entity linking · Entity recognition · Entity disambiguation · Character embedding.

1 Introduction

Entity linking aims to recognize potentially ambiguous mentions of entities in a text and link them to a target knowledge base (KB). It is an essential step for many NLP tasks, such as knowledge fusion, KB construction and KB-based QA. An EL system typically consists of two subtasks: i) Entity Recognition - extracts all possible entity references (i.e. mentions) from a text fragments and ii) Entity Disambiguation - maps these ambiguous mentions to the correct entities in KB.

Entity linking has been studied for many years and has achieved great advancement with neural networks [6, 10, 13]. While most of the works are designed for English corpus, especially for long texts, the CCKS2019 Task 2 focuses on Chinese short texts instead. Compared with entity linking of English long texts, Chinses short text entity linking is a more challenging task. First, Chinese texts lack explicit delimiters such as whitespace to separate words, making it difficult to recognize mention boundaries. Previous studies for Chinese ER can be mainly divided into two categories: word-based and character-based method. Character-based method has been shown empirically superior to word-based method as it doesn't suffer from word segmentation errors [8, 11]. However it can't fully leverage latent word sequence information, which is potentially useful. To this end, external information is needed to improve its performance. Second, many recent ED models take advantage of global context to capture the coherence among the entities for a set of related mentions in a document [1, 7]. However short texts are often nosiy and less coherent, lacking rich contextual information for global method. Third, mentions' positions should be taken into consideration when they are presented with neural models, otherwise all mentions in the same text are viewed as identical. Previous works [13, 16] usually consider mention's position by splitting local context into two part: the left and right side of the mention and using a duo of neural networks to process each side. However, it is not suitable for short texts because both sides of context will be shorter than the original text, making it more difficult to extract useful semantic information.

To address the above challenges, we propose an ehanced character embedding based neural approach to incorporate additional information into EL system. **1) For entity recognition**, we regard it as a sequence labeling problem and use a character based BiLSTM-CNN-CRF model to solve it. Based on the observation that all extracted mentions must exist in the mention dictionary provided by the knowledge base, we explore several ways to construct feature embeddings for each character to integrate dictionary information into ER model and help identify mention boudaries. Besides, we design some other embeddings that contain rich contextual information. All these embeddings will be combined with the vanilla character embedding to compose enhanced character embedding to improve the performance of ER model. **2) For entity disambiguation**, we model it as a semantic matching problem between mention's context and candidate entity's description text. We use a BiLSTM-CNN model to generate representations for mention and candidate entity, which are further used to model semantic similarity between them with cosine measure. As for mention representation, in order to incorporate position information, we first calculate each character's relative distance from the mention and convert it to a continous position embedding, which represents its relevance to the mention. The position embedding will be combined with vanilla character embedding to compose enhanced character embedding. Besides, we generate representation using hidden states from BiLSTM-CNN only of the mention part, instead of the overall sequence. Experiments show that our enhanced character embedding based method outperforms traditional methods. **3) For ensemble**, we exploit two strategies: model weights ensemble and outputs ensemble, to improve the final performance. Our solution achieves a F1 score of 0.79266 on the final test data of CCKS2019 Task 2.

2 Model Description

2.1 Data

The CCKS2019 Task 2 has provided 90K labeled sentences for training, 10K unlabeled for preliminary testing and 30K unlabeled for final testing. Most of them are short texts with an average length of 22. The knowledge base is also provided, which includes nearly 0.4M entity information. We extract "alias" and "subject" field of each entity information to construct a mention dictionary. We also build a map from mention to entities to generate candidate entities. Furthermore, we concatenate all the tuples of (predict, object) from "data" field into one sentence to represent entity description. Most of the description texts are considerably long, with an average length of 196.

2.2 Entity Recognition Model

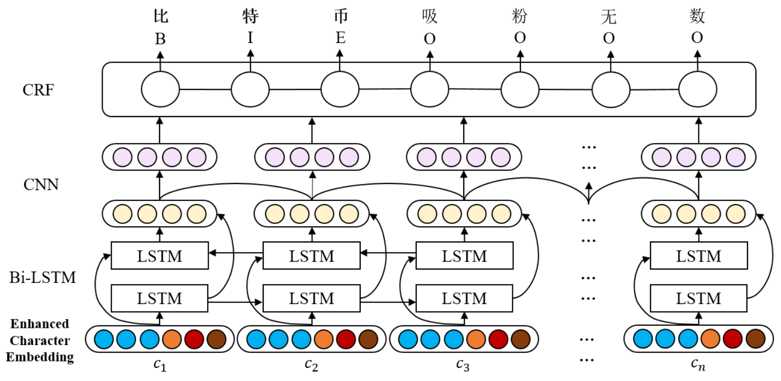

Fig. 1. The overall framework of entity recognition model.

Our proposed ER model is shown in Figure 1. Same as the widely used Chinese ER neural model, we use character-based BiLSTM-CNN-CRF as the main network structure. Formally, we denote a Chinese sentence as: $\mathbf{s} = \{c_1, c_2, \cdots, c_n\}$, where c_i denotes the i-th character. The ER model aims to label each character c_i with BIOES tagging scheme. The differences between our model and a standard character-based model are mainly on the embedding layer. First, same as standard character-based model do, we convert each character c_i to a dense vector, which we call "vanilla character embedding", using pre-trained embedding method. Here we use word2vec [12] to train on the training data and entity description texts to obtain them. Apart from that, we also design 7 extra feature embeddings that capture mention dictionary and rich contextual information to enhance vanilla character embedding for handling rare and unseen cases and identifying mention boundaries efficiently. We use sentence “比特币吸粉无数” (“比特币” as the ground truth), as an example to introduce these embeddings.

Word Embedding Mention boundaries usually coincide with some word boundaries, which suggests that words in character sequence can provide rich boundary information for character-based model. To this end, we first use jieba[3] to load

[3] https://github.com/fxsjy/jieba

the mention dictionary as its default dictionary and then segment the sentences into word sequences. This procedure can ensure a great possibility that each potential mention can be segmented as a single word. For instance, the sample sentence will be cut by jieba into a sequence: [“比特币”, “吸粉”, “无数”]. We then use word2vec to train on these word sequences to obtain word embeddings. Finally we add the same word embedding to each character in the word.

Charater Position Feature After obtaining segmentation results with jieba, we can also provide boundary information with character position features in the word. We use BMES tagging scheme to represent character position in the word. Therefore, the example setence will be tagged as [“B”, “M”, “E”, “B”, “E”, “B”, “E”]. We initialize a dense vector for each position label using uniform distribution and then optimize it during the model training phase. Each character will be assigned with a corresponding position feature vector.

Position-aware Character Embedding We also try to combine character sequences with the character position label sequences descrided above. For the example, we can get a sequence: [“比B”, “特M”, “币E”, “吸B”, “粉E”, “无B”, “数E”]. We then again use word2vec to train on these sequences to obtain position-aware character embeddings, yielding multiple embeddings per character. Learning separate embeddings for each positionally tagged character can help distinguish between uses of the same character in different contexts.

Max Match Feature Apart from jieba, we use BiDirectional Maximum Matching algorithm [5] (BDMM) with mention dictionary to segment sentences as well. With BDMM, we can get potential mentions in a coarse granularity. Different from jieba, all characters that mismatch the dictionary will remain as a single token. So by applying BDMM to the example we can get a sequence: [“比特币”, “吸”, “粉”, “无”, “数”]. Similar as character position feature, we use BMEO tagging scheme to indicate whether the character is in a matched mention and its position. We also initialize a dense vector for each label using uniform distribution and optimize it during the model training phase, which we call max match feature.

N-gram Match Feature Inspired by the work on Chinese word segmentation [17], we generate n-gram match feature t_i for each character c_i to represent whether a text segment that contain character c_i and its surroundings is mention or not. We first segment the context of character c_i based on the pre-defined n-gram feature templates, as listed in Table 1. We then use a binary value to indicate whether the text segment is a matched mention or not. Here we use $t_{i,2(k-1)-1}$ and $t_{i,2(k-1)}$ to denote the value of the output corresponding to the k-gram template for c_i. Figure 2 shows an example of n-gram match feature construction of character “币”. In this paper we actually consider 7-gram template at most. So we finally generate a 14-dimensional multi-hot vector for each character.

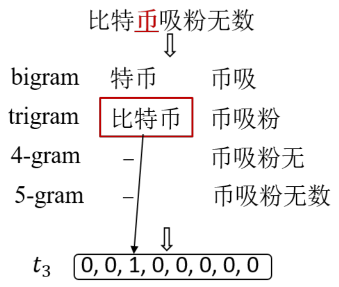

Fig. 2. Example of n-gram match feature embedding construction. The character underline in red is the character c_i. The text segment with red rectangle is the matched mention.

Bigram Embedding Character bigrams have been shown useful for representing characters in word segmention [2, 15]. We again use word2vec to train on bigram sequences of the training data, such as [“比特”, “特币”, “币吸”, “吸粉”, “粉无”, “无数”], to obtain bigram embeddings.

BERT Embedding BERT [4] has become enormously popular in recent NLP studies, which utilizes large-scale unlabeled training data and generates enriched contextual representation. Ever since the release of BERT, there have been several studies foucing on enhanced pretrained language model for Chinese, such as ERNIE [14] and BERT_wwm [3]. Both of them have achieved great improvement on various Chinese NLP tasks. In this paper, we try these 3 bert models to enhance character embedding. We present their contributions in next section.

We concatenate the above embeddings with vanilla character embedding to compose our proposed enhance character embedding, which will be passed to BiLSTM-CNN to obtain hidden state sequence: $\mathbf{H} = \{h_1, h_2, \ldots, h_n\}$. BiLSTM-CNN is able to fully capture both local and long-distance contexts for ER. A CRF layer is then used on top of the hidden state sequence in order to consider the dependencies between successive labels. The probability of a label sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ is calculated as follow:

$$p(\mathbf{y}|\mathbf{s}) = \frac{\exp\left(\sum_i \left(\mathbf{O}_{i,y_i} + \mathbf{T}_{y_{i-1},y_i}\right)\right)}{\sum_{\tilde{y}} \exp\left(\sum_i \left(\mathbf{O}_{i,\tilde{y}_i} + \mathbf{T}_{\tilde{y}_{i-1},\tilde{y}_i}\right)\right)} \tag{1}$$

where $\tilde{y}$ denotes an arbitrarily label sequence, $\mathbf{O}$ is a state score matrix calculated based on the label itself and $\mathbf{T}$ is a transition score matrix calculated based on the adjacent labels.

Given the labeled training data $\{(\mathbf{s}_j, \mathbf{y}_j)\}_{j\in\overline{1,T}}$, we minimize the sentence-level negative log-likelihood loss to train our ER model:

$$L = -\sum_j \log\left(p\left(\mathbf{y}_j|\mathbf{s}_j\right)\right) \tag{2}$$

Viterbi algorithm is used to find the highest scored label sequence after training.

2.3 Entity Disambiguation Model

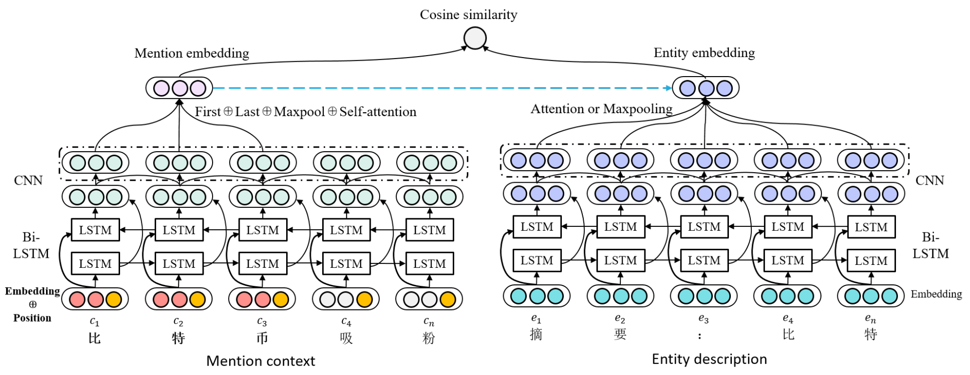

Fig. 3. The overall framework of entity disambiguation model.

With ER model, we recognize a list of mentions in a text. For each mention, we retrieve candidate entities using the map from mention to entities that we previously construct. The task of ED is to find the most matching entity from the candidates. Formally, given the mentions $\{m_i\}_{i\in\overline{1,T}}$ extracted from sentence $\mathbf{s} = \{c_1, c_2, \ldots, c_n\}$, where each mention is a subsequence of $\mathbf{s}$: $m = \{c_q, \ldots, c_r\}$, the output of EL model is a list of mention-entity pairs: $\{(m_i, e_i)\}_{i\in\overline{1,T}}$.

In this paper, we investigate entity disambiguation using only local information provided with mention context and entity description. The architecture of our proposed ER model is shown in Figure 3. We again use BiLSTM-CNN as the main network structure to generate fixed-sized representations for mention and candidate entity. After that, we utilize cosine function to calculate semantic similarity score for each pair of mention and candidate entity. The candidate entity with the highest score is chosen as the matching one. Note that we do not use siamese structure due to the significant length difference between mention context and entity description.

Mention Representation We propose two ways to take mentions' positions into consideration when representing them. First, at the embedding layer, we concatenate vailla character embedding with position embedding for each character. The position of a context character is defined as its relative distance from the mention in the sentence and the position of a mention character is regared as 0. Each positional value is initialized with a dense vector (i.e. position embedding) using uniform distribution and optimized during the model training phase. Second, suppose the hidden state sequence output from the left BiLSTM-CNN in Figure 3 is $\mathbf{H}^m = \{h_1^m, h_2^m, \ldots, h_n^m\}$, we produce representation only using the hidden states of mention subsequence instead of the overall sequence, because the former is more informative. To be specific, given a mention $m = \{c_q, \ldots, c_r\}$, we concatenate the first and the last hidden state as well as the results of max-pooling and self-attention over the hidden states of mention subsequence:

$$g^m = \left[h_q^m; h_r^m; h_{maxpool}^m; h_{self-attend}^m\right] \tag{3}$$

The self-attend embedding $h_{self-attend}^m$ is calculated as follow:

$$\begin{aligned} \alpha_i^m &= w_\alpha^T h_i^m \\ \alpha_i^m &= \frac{\exp\left(\alpha_i^m\right)}{\sum_{k=q}^{r} \alpha_k^m} \\ h_{self-attend}^m &= \sum_{k=q}^{r} \alpha_k^m h_k^m \end{aligned} \tag{4}$$

We then project g^m to final mention representation r^m with the same size as entity representation (see below) using a simple dense layer:

$$r^m = \text{Dense}\left(g^m\right) \tag{5}$$

Entity Representation Suppose the hidden state sequence output from the right BiLSTM-CNN in Figure 3 is $\mathbf{H}^e = \{h_1^e, h_2^e, \ldots, h_n^e\}$. Apart from using max-pooling over the hidden states $\mathbf{H}^e$, we also try to utilize attention mechanism to generate entity representation. Different from the self-attention used above, here we use mention representation r^m to attend to the hidden states, aiming to find out the most relevant characters in entity description. Entity representation r^e based on attention mechanism is calculated as follow:

$$\begin{aligned} \alpha_i^e &= \text{score}\left(r^m, h_i^e\right) \\ \alpha_i^e &= \frac{\exp\left(\alpha_i^e\right)}{\sum_{t=1}^{n} \alpha_t^e} \\ r^e &= \sum_{t=1}^{n} \alpha_t^e h_t^e \end{aligned} \tag{6}$$

where α_i^e is the attentive score for r^m and i-th hidden state h_i^e. We explore 3 ways to calculate α_i^e: i) additive attention: $\alpha_i^e = v_\alpha^T \tanh\left(W_a\left[r^m; h_i^e\right]\right)$; ii) multiplicative attention: $\alpha_i^e = r^m W_a h_i^e$; iii) scaled-dot attention: $\alpha_i^e = {r^m}^T h_i^e / \sqrt{d}$. The performances of these 4 encoding strategies are shown in next section.

For training ED model, we first obtain negative samples from the candidate list excluding ground truth entity, and then minimize the margin-based loss:

$$L\left(r^m, r_+^e, r_-^e\right) = \sum_j \sum_{i=0}^{n} \max\left(m + \text{cosine}\left(r^{m_j}, r_-^{e_{j,i}}\right) - \text{cosine}\left(r^{m_j}, r_+^{e_j}\right), 0\right) \tag{7}$$

where n is the number of negative samples and m denotes margin. The idea behind is to encourge the cosine similarity score of positive sample to be at least m higher than those of negative samples.

2.4 Model Ensemble

We exploit 2 ensemble strategies to further improve the performance of our proposed approach.

Weights Averaging Inspired by [9], during a single training process, we maintain a copy of model's weights w_a in memory, which is used to keep track of the averaged weights. After each epoch of training, we update the weights of the copy:

$$w_a = \frac{w_a \cdot n_{models} + w}{n_{models} + 1} \tag{8}$$

where n_{models} is the number of models already included in the average and w represents the weights of the model being trained. This amounts to storing the running average of the models seen at the end of each epoch of training. Experiments show that this "smooth" version of model almost always outperforms the best single model obtained during a single training.

Outputs Averaging Another simple yet effective strategy to improve performance of neural models is to train multiple models instead of a single model and combine the predictions from these models. In this paper, we design various ER and ED models with different model configurations and then simply average their outputs to obtain final predictions. We show the details in next section.

3 Experiments

3.1 Experimental Setup

Our proposed approach is implemented by Keras[4] with Tensorflow backend. We use Adam as the optimizer and set the initial learning rate to be 0.001. The batch size is 32. All hidden states of BiLSTMs, feature maps of CNNs and the pre-trained embeddings based on word2vec have 300 dimensions, while the randomly-initialized embeddings have 50 dimensions. During training, all the pre-trained embeddings and BERT model are untrainable, while the randomly-initialized embeddings are fine-tuned. As for ED model training, we get 5 negative samples for each mention-entity pair and set margin $m = 4$. As for weights ensemble, we start model weights averaging after 3 and 5 epochs of training ER and ED model respectively.

We randomly split the labeled data into training set and dev set with a 9:1 ratio. Earlystopping is applied to avoid overfitting: training will stop when we observe no performance improvement on dev set after 3 epochs, then we store the model with best performance on dev set. Next we show performances of our proposed models on dev set, preliminary and fianl test set using F1-score metric.

Table 1. F1 scores of ER models on dev set.

Model	**Dev F1**	
		Weights ensemble
vanilla character	0.77316	0.77524
enhanced character (BERT)	0.83145	0.83323
enhanced character (ERNIE)	**0.83175**	**0.83355**
enhanced character (BERT-wwm)	0.83019	0.83126
ER model ensemble	-	**0.83836**

Table 2. Ablation study of enhanced character embeddings of ER model on dev set.

Model	**Dev F1**	
		Weights ensemble
enhanced character (ERNIE)	0.83175	0.83355
- word embedding	0.82878	0.83042
- character position feature	0.83080	0.83173
- position-aware character embedding	0.83159	0.83303
- n-gram match feature	0.83067	0.83185
- max match feature	0.83009	0.83116
- bigram embedding	0.83148	0.83282
- bert embedding	0.82969	0.83034

3.2 Results of ER models on dev set

The experimental results of ER models are presented in Table 2. We compare our proposed enhanced character embedding based model with vanilla character embedding based model. The differece between the 2nd to 4rd model is the bert embedding they use. As indicated from the table, enhanced character embedding does bring a significant performance gain, highlighting the utility of incorporating mention dictionary and rich contextual information for entity recognition. Among the enhanced models, the one using ERNIE as bert embedding stands out from the others. We believe it is because ERNIE is trained on not only Wikipedia data but also many web texts, which is useful for informal texts like the dataset used in this task. Besides, by applying weights ensemble strategy, we can always get another weights-averaged model for each single training. We also list their performances in the table. We can find that the weights-averaged model achieves substantial improvements over the original model, suggesting that model weights averaging leads to better generalization.

To gain a better understanding of the impact of each enhanced embedding, we perform an ablation study and present the results in Table 3. We observe performance degradation when eliminating any embedding, showing that each embedding contribues to the ER performance. Interestingly, the contributions of bigram embedding and positon-aware character embedding is much less than those of the others. As a result, we further train various models with and without

[4] https://keras.io/

using bigram embedding and position-aware character embedding as input. We then combine all these models using outputs ensemble strategy and observe further performance boost. The result of ER model ensemble is shown in the last row of Table 2.

3.3 Results of ED models on dev set.

Table 3. F1 scores of ED models on dev set.

Model			**Dev F1**	
Input	**Mention**	**Entity**		**Weights ensemble**
no pos	over all	maxpool	0.89582	0.90096
no pos	over 2 sides	maxpool	0.89414	0.89971
no pos	over subseq	maxpool	0.89902	0.90296
pos	over all	maxpool	0.89727	0.90169
pos	over 2 sides	maxpool	0.89682	0.90125
pos	**over subseq**	**maxpool**	**0.89925**	**0.90344**
pos	over subseq	add attend	0.89844	0.90249
pos	over subseq	mul attend	0.89891	0.90308
pos	over subseq	scaled-dot attend	0.89837	0.90278
ED model ensemble			-	**0.90965**

We present performances of different ED models on dev set in Table 4. The "Input" column denotes whether to enhance character embedding with position embedding. The "Mention" and "Entity" column denotes the ways to generate representation from hidden state sequence. For mention representation, "over all" means applying maxpooling over all the hidden states, "over 2 sides" means applying maxpooling over hidden states of the left and right side of mention resepectively and "over subseq" is our proposed method which only process over the hidden states of mention subsequence. From the results of the first 6 models, we can see that with "pos" input and "over subseq" strategy to generate mention representation, our approach achieves the best performance, indicating the importance of integrating mention position information. Although "over 2 sides" strategy takes mention's position into consideration as well, but it performs even worse than "over all" strategy. We believe it is because both sides of context are shorter than the original text, which makes it more difficult to capture usefull semantic information. For entity representation, we compare the 4 strategies that we introduce in Section 2.3. From the results of the 6th to 9th models, we surprisingly find that "maxpool" strategy outperforms all the attention-based strategies. We assume it is because entity description is too long for attention mechanism to effetively extract the most discriminative character. We also apply weights ensemble for each ED model training and observe substantial improvements. Finally, we ensemble the outputs of all the single models in Table 4 to obtain final predictions. And the result is shown in the last row.

3.4 Results of EL model

Table 4. F1 scores of the proposed EL model on dev, preliminary test and final test set.

EL Model	Dev F1	Pre Test F1	Final Test F1
ER model ensemble + ED model ensemble	0.77442	0.77275	0.79266

Combining the above ensemble ER and ED model is our proposed EL model for CCKS2019 Task 2. We present its result on dev set, preliminary test set and final test set in Table 5. Since these datasets are sampled from the same distribution, the performances are very similar. Note that the performance gain in the final test set is mainly due to organizers' careful review to the final test set and removal of some wrong labels.

4 Conclusion

In this paper, we set out to investigate the utility of external information for Chinses short text entity linking. To this end, we propose a enhanced character embedding based neural approach which explicitly incorporates mention dictionary and mention position information into the models. We achieve significant improvements over a collection of baselines on CCKS2019 Task 2, verifying the value of such information.

Acknowledgments This work was supported by 2019 Tencent Marketing Solution Rhino-Bird Focused Research Program.

References

1. Cao, Y., Hou, L., Li, J., Liu, Z.: Neural collective entity linking. CoRR **abs/1811.08603** (2018), http://arxiv.org/abs/1811.08603
2. Chen, X., Qiu, X., Zhu, C., Liu, P., Huang, X.: Long short-term memory neural networks for Chinese word segmentation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1197–1206. Lisbon, Portugal (Sep 2015)
3. Cui, Y., Che, W., Liu, T., Qin, B., Yang, Z., Wang, S., Hu, G.: Pre-training with whole word masking for chinese BERT. CoRR **abs/1906.08101** (2019), http://arxiv.org/abs/1906.08101
4. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
5. Gai, R.L., Gao, F., Duan, L.M., Sun, X.W., Li, H.Z.: Bidirectional maximal matching word segmentation algorithm with rules (2014)

6. Ganea, O.E., Hofmann, T.: Deep joint entity disambiguation with local neural attention. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 2619–2629. Copenhagen, Denmark (Sep 2017)
7. Globerson, A., Lazic, N., Chakrabarti, S., Subramanya, A., Ringgaard, M., Pereira, F.: Collective entity resolution with multi-focal attention. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 621–631. Berlin, Germany (Aug 2016)
8. He, J., Wang, H.: Chinese named entity recognition and word segmentation based on character. In: Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing (2008)
9. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D.P., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. CoRR **abs/1803.05407** (2018), http://arxiv.org/abs/1803.05407
10. Kolitsas, N., Ganea, O.E., Hofmann, T.: End-to-end neural entity linking. In: Proceedings of the 22nd Conference on Computational Natural Language Learning. pp. 519–529. Brussels, Belgium (Oct 2018)
11. Li, H., Hagiwara, M., Li, Q., Ji, H.: Comparison of the impact of word segmentation on name tagging for chinese and japanese. In: LREC (2014)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
13. Phan, M.C., Sun, A., Tay, Y., Han, J., Li, C.: Neupl: Attention-based semantic matching and pair-linking for entity disambiguation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 1667–1676. CIKM '17, ACM, New York, NY, USA (2017)
14. Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Zhu, D., Tian, H., Wu, H.: ERNIE: enhanced representation through knowledge integration. CoRR **abs/1904.09223** (2019), http://arxiv.org/abs/1904.09223
15. Yang, J., Zhang, Y., Dong, F.: Neural word segmentation with rich pretraining. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 839–849. Vancouver, Canada (Jul 2017)
16. Zeng, W., Tang, J., Zhao, X.: Entity linking on chinese microblogs via deep neural network. IEEE Access **6**, 25908–25920 (2018)
17. Zhang, Q., Liu, X., Fu, J.: Neural networks incorporating dictionaries for chinese word segmentation (2018)