Investigating Named Entity Recognition on Chinese Social Media Dataset

Chenkuan Liu
Natural Language Processing
University of Texas at Austin
ckliu@utexas.edu
May 2021

Abstract

Named entity recognition (NER) is one of the fundamental tasks in natural language processing. With the introduction of deep learning and its various architectures, the performance of NER systems has experienced a significant boost. Despite being well studied, challenges associated with NER still remain in different languages and domains. In this report, I am going to investigate the NER task on Chinese social media dataset. I will first give a brief introduction on the chosen dataset, and then summarize a specific deep neural network model proposed by a recent paper on this dataset. After that, I will demonstrate a model I have constructed that draws inspirations from that model as well as other research papers, and show that it can surpass or achieve similar performance without the complex regimes introduced by that model. Finally, I will give an analysis on the model I have constructed as well as the Chinese social media NER task in general, and provide possible further directions associated with it.

1 Introduction

The goal of NER is to recognize specific named entities in a given text. Currently, most NER models utilize a combination of deep neural networks and conditional random fields (CRF) and have achieved high performance on various dataset. However, NER is a broad task and challenges still remain across different languages and domains. At here, I am studying the NER on Chinese social media data. Unlike many Indo-European languages, Chinese is a logographic language. In other words, it is a character-based language and each character can be treated as a single unit with its own meaning. In addition, unlike news report or Wikipedia, social media involves a lot of slang and ungrammatical, informal expressions, which pose special challenges to the NER task. In this report, I am going to look at the Weibo dataset, which is a widely used dataset for Chinese NER task and the language data inside it comes from Sina Weibo, a Chinese microblogging website and one of the largest Chinese social media platform.

This dataset contains 1350 training sentences, 270 dev sentences and 270 test sentences. It is annotated with four different entity types (person, location, organization and geopolitical entities). The longest sentence has 80 characters, so all sentences can be padded to have length 80. Therefore, for each sentence, we can treat the input as a $d_e \times 80$ data, where d_e is the embedding dimension of each character. For the output, it is of size 9×80, where 9 is the cardinality of the entity set {O, B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG, B-GPE, I-GPE}. Figure 1 provides a brief overview of the task.

Task	Hilton			leaves		Houston			Airport	
	希尔顿			离开		休斯顿			机场	
Chinese NER	希	尔	顿	离	开	休	斯	顿	机	场
	B-PER	I-PER	I-PER	O	O	B-LOC	I-LOC	I-LOC	I-LOC	I-LOC
CWS	希	尔	顿	离	开	休	斯	顿	机	场
	B	I	E	S	S	B	I	E	B	E

Figure 1: An example of Chinese NER task and CWS task (discussed in Section 2.1)

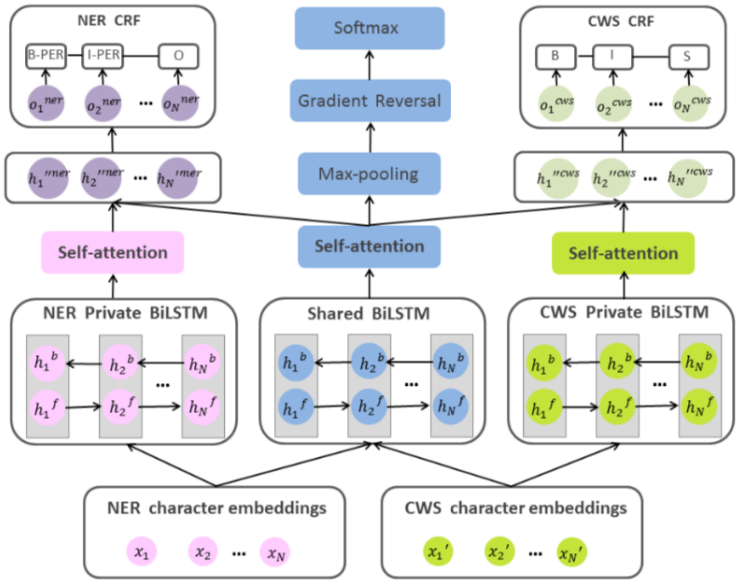

Figure 2: Model proposed by Cao et al. (2018)

2 Proposed Model

2.1 Model Description

The baseline model for this task uses LSTM to extract sentence information and CRF to make inference and decode. However, many improvements can be made on the baseline and bring improvements on the performance. Specifically, the model I am looking at is proposed by Cao et al. (2018), which uses a combination of LSTM, self-attention, transfer learning, adversarial learning and CRF for this task. Figure 2 below shows the general structure of the proposed model. In particular, besides the LSTM, self-attention and CRF structure, it employs transfer learning from the Chinese word segmentation (CWS) task and adversarial learning by constructing a discriminator to tell whether the current training sentence is for the NER task or the CWS task.

The idea behind this paper is the similarity of NER and CWS task, which can also be observed in Figure 1. In general, since Chinese is a character-based language and there is no blank spaces between the characters, the CWS task is aimed to segment a sentence and tell if a character is at the beginning of a word (B), in the middle of a word (I), at the end of a word (E), or is a single word by itself (S). From Figure 1, we can observe that there are certain similarities between these two tasks. Specifically, most named entities could form a word block of [B,I,E] (there could be multiple I's within) in CWS task. In addition, by telling if the sentence

comes from NER task or CWS task, the adversarial learning regime can balance the difference between these two tasks and allow the CWS task to be better transferred to improve the performance of the original NER task. Figure 3 compares the performance of the proposed model with the previous work by Peng and Dredze (2015) and Peng and Dredze (2016), which did not involve significant construction of neural network. Figure 4 compares the performance of the proposed model with its simplified versions. The character embeddings used in these models are pretrained on this dataset using skip-gram.

Models	P(%)	R(%)	F1(%)
CRF (Peng and Dredze, 2015)	56.98	25.26	35.00
CRF+word (Peng and Dredze, 2015)	64.94	25.77	36.90
CRF+character (Peng and Dredze, 2015)	57.89	34.02	42.86
CRF+character+position (Peng and Dredze, 2015)	57.26	34.53	43.09
Joint(cp) (main) (Peng and Dredze, 2015)	57.98	35.57	44.09
Pipeline Seg.Repr.+NER (Peng and Dredze, 2016)	64.22	36.08	46.20
Jointly Train Char.Emb (Peng and Dredze, 2016)	63.16	37.11	46.75
Jointly Train LSTM Hidden (Peng and Dredze, 2016)	63.03	38.66	47.92
Jointly Train LSTM+Emb (main) (Peng and Dredze, 2016)	63.33	39.18	48.41
BiLSTM+CRF+adversarial+self-attention	55.72	**50.68**	**53.08**

Figure 3: Comparison of the performance by Cao et al. (2018) with the previous work by Peng and Dredze (2015) and Peng and Dredze (2016)

Models	SighanNER			WeiboNER		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
BiLSTM+CRF	89.84	88.42	89.13	58.99	44.93	51.01
BiLSTM+CRF+transfer	90.60	89.19	89.89	60.00	46.03	52.09
BiLSTM+CRF+adversarial	90.52	89.56	90.04	61.94	45.48	52.45
BiLSTM+CRF+self-attention	90.62	88.81	89.71	57.81	47.67	52.25
BiLSTM+CRF+adversarial+self-attention	**91.73**	**89.58**	**90.64**	55.72	**50.68**	**53.08**

Figure 4: Comparison of the full model by Cao et al. (2018) with its simplified variations.

2.2 Model Analysis

This model has utilized many state-of-the-art architectures by its time and achieved a significant improvement compared to previous work. However, in order to train on the Weibo dataset, it transfers from another CWS dataset called MSR dataset, which contains more than 85000 CWS training sentences and is much larger than the Weibo dataset. Moreover, the Weibo dataset has experienced an update within recent years. After the update, the sentences are provided along with the character position indices. For instance, take the character "好"(lit. means "good") as an example. In the word "好人"(lit. means "good man"), "好" is at the start position (we count from zero), "人"(lit. means "man") is at the first position, and so this word is provided as "好0 人1". In another word such as "你好"(lit. means "hello"), "你"(lit. means "you") is at the start position and "好" is at the first position, so this word is provided as "你0 好1". To summarize, in the updated Weibo dataset, each character is provided along with a position index. We can see from the above example that the same character will has a different "char+pos" combination depending on the specific position it is inside the word. Moreover, the updated character embedding is pretrained on the "char+pos" combination rather than the character alone. With all these additional information, as well as the continuing progress in deep learning, a better way to perform NER task on this dataset is to only use the information provided in this dataset alone, rather than transferring from other data resource which is much larger than the task we are dealing with. With this in mind, I have constructed a model based on the ideas in Cao et al. (2018) as well as other proposed deep learning methods, which is displayed in the next section.

3 Constructed Model

3.1 Model Description

Inspired by the adversarial transfer learning framework in Figure 2, I construct a model which involves several different neural network structure but only requires the data from Weibo dataset alone. The model structure

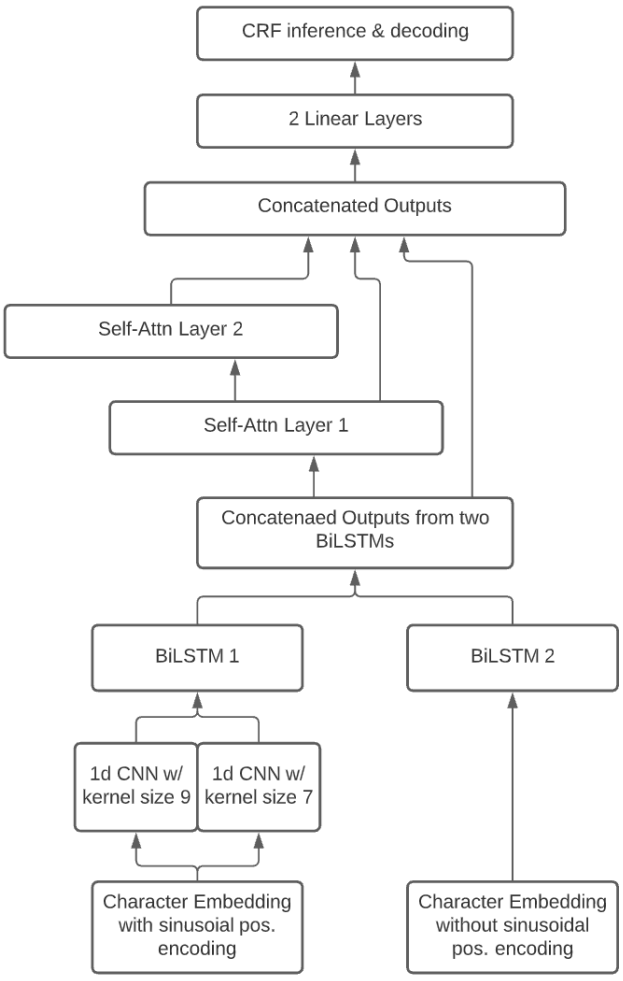

Figure 5: Constructed model

is displayed in Figure 5. I also use multiple embeddings and Bi-LSTMs in the model, which are updated separately. However, they are targeted towards the same task as there is only one CRF layer at the end, and the difference between the two embeddings comes from the ways they are constructed.

The embedding on the left has 470-dim in total and is concatenated from 5 elements. The first element is a 100-dim pretrained embedding on characters only. The second element is another 100-dim pretrained embedding on the combination of character and position described in Section 2.2. The third element is a 30-dim embedding on character position inside the word only, which is initialized randomly and will be updated during training. The fourth element is a 40-dim embedding on character position inside the sentence, with 0 assigned to the first character inside a sentence and 79 assigned to the last character inside a sentence. It is also initialized randomly and will be updated during training. The fifth element is a frozen 200-dim sinusoidal position embedding. It is inspired by the position encoding in transformer and follows the style proposed by Vaswani et al. (2017), where $PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$, $PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$, and $d_{model} = 200$. The reason to add different kinds of positional embeddings is that the position information is available to both known characters and unknown characters that might be encountered during development and testing, while character information is only available to known characters. These position information will help the model better generalize to unknown characters. The embedding on the right side has 270-dim in total. The only difference is that it does not contain the 200-dim hard-coded sinusoidal position encoding.

The 1d-convolution layers applied on the left embedding is inspired by the Inception Network from Szegedy et al. (2015). The original Inception Network uses concatenated 2d-convolution layers for image recognition. At here, though the setup is different, the idea to add 1d-conv. layers with different kernel sizes is to help better

generalizing the information provided in the concatenated embedding and dealing with potential unknown characters encountered during development and testing. The kernel sizes of the 1d-conv. layers are chosen to be 9 and 7 respectively; this is based on empirical results as well as the fact that in Chinese sentences, most named entities have length less than 7. Same padding is used to ensure the outputs still have length 80. The 1d-conv. outputs are then concatenated together and fed into the left Bi-LSTM network.

The outputs of the two Bi-LSTMs are then concatenated together and fed into two self-attention layers. Inspired by the Residual Network (He et al., 2016), skip-connections are used at here so that the outputs from BiLSTMs, first self-attention layer and second self-attention layer are all concatenated and then fed into the feed-forward Linear layer. The Linear layer will output a $80 \times B \times 9$ tensor, where B is the batch size and 80×9 represents the emission log-probability of each of the 9 tags at each position. This tensor is then fed into CRF inference and decoding to generate the output entity labels.

For CRF, the transition matrix is initizliaed by setting large negative values for illegal transition moves and zero values for other moves. The initial starting values and end values for each tag are all set to zeros. There is no single best choice of other hyperparameters regarding this model. I have experimented with batch size 8~20, learning rate 0.0005~0.0008 (with Adam optimizer), random shuffling, 1d-conv, layer output channels 150~300, BiLSTM hidden dimension 250~400, dropout rate 0.30~0.45, self-attention head number 8~20 and other hyperparameter settings. With 40~50 epochs training, the general performance range of the model on dev set and test set, as well as the best performance I am getting so far, are all listed below. Simpler variations such as LSTM+self-Attn+CRF and CNN+LSTM+self-Attn+CRF are also experimented and their general results are also provided below for better comparison.

Model Structure	Sinusoidal Pos. Encoding	Dev F1 (%)	Test F1 (%)
LSTM+Self-Attn+CRF	✗	~51	~49
CNN+LSTM+Self-Attn+CRF	✗	52.0~53.5	50~52
Figure 5 Model	✓	54~56	50.5~52.5

Table 1: Dev and Test Performance Range

Set	Precision (%)	Recall (%)	F1 (%)
Dev	56.16	56.00	**56.08**
Test	58.96	47.38	**52.54**

Table 2: Current Best Performance using Figure 5's Model

3.2 Model Analysis

The first thing noticeable is the performance difference between dev set and test set. From the table, we can see that for most cases the F1-score for dev set will be a few percentage higher than that for test set. One key reason is that the dev set contains fewer unknown characters than the test set. Since the provided embedding is mostly pretrained on the training set, it means that the dev set has more similarities to the training set and it is easier to achieve slightly better performance on the dev set than the test set. Another possible explanation is that the sentence structure and word distribution of the dev set is closer to the training data than the test set. As a result, all these reasons may lead to the fact that the test set is harder to predict than the dev set.

Now we can compare the results at here to those provided in the last section. The performance in Figure 3 and Figure 4 are not specified but are presumably evaluated on the test set. If we take both dev set and test set into account, we can see that, the best F1-score we have achieved is 56.08, which is several percentage higher than the final performance provided in Table 3 by Cao et al. (2018). If we only take test set into account, we can see that our performance has surpassed all the simplified versions in Table 3 and is only about 0.5 lower than the final model in it. As a result, it means that we can achieve comparable performance using only the information provided in this dataset, without perform explicit transfer learning from other dataset that has significant larger size than our target dataset. This is valuable because applying explicit transfer learning from other large dataset will cost significant computation resources. Though further work still needs to be done to close the performance gap between dev set and test set results, the results at here demonstrates that the model is able to

achieve comparatively decent performance using only the local information provided in the Weibo dataset.

4 Further Extension

With the introduction of transformer (Vaswani et al, 2017) and its various usage, more recent paper (Li et al., 2020) has employed transformer encoder along with LSTM and CRF and achieved 61.01 F1-score performance. The model structure is demonstrated in Figure 6 below. This model has much simpler structure compared to the models in Figure 2 and Figure 5. I have tried to implement this as well with 6 self-attention layers inside transformer encoder and 1 BiLSTM layer, and the results are listed below along with the performance displayed in the paper.

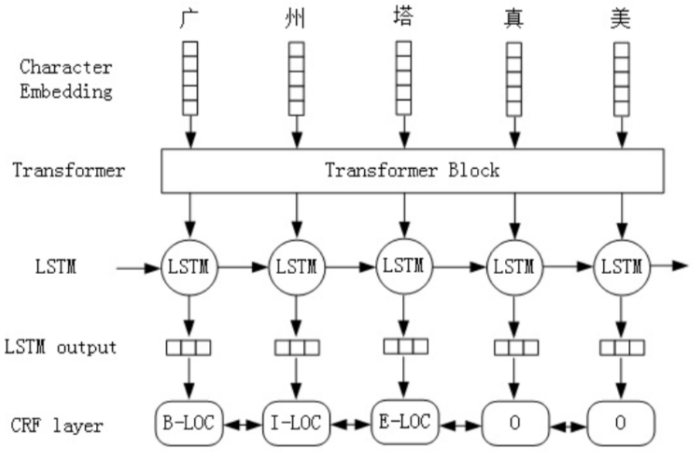

Figure 6: Transformer encoder structure in Li et al. (2020)

	Set	Precision (%)	Recall (%)	F1 (%)
Paper performance	/	57.34	64.18	61.01
Re-implementation	Dev	58.73	52.86	55.64
Re-implementation	Test	55.40	41.62	47.53

Table 3: Transformer encoder model performance

We can see that the performance at here is several percentage lower compared to the results declared in the paper, which is not specified but presumably evaluated on the test set. The paper does not publicize the character embedding it has used. Therefore, a reasonable explanation is that it uses a more comprehensive embedding with much less unknown characters than my implementation. However, another thing noticeable is that, there is a significant gap between the dev set performance and the test set performance in my implementation, which is much higher than the gap observed in Table 1 and Table 2 in Section 3. This is partially due to the fact that the learning from self-attention and transformer encoder can better generalize to the dev set that is more similar to the training data than the test set. I have also tried to add convolution layers inside. However, as there are already a lot of weight parameters inside the transformer encoder, adding additional conv. layers will make the weight update more difficult and the performance will stuck at a very low percentage for a long time.

However, if we only look at the dev set, we can see that it is only 0.44 lower than the dev set performance in Table 2 using Figure 5's model. As a result, the transformer encoder+LSTM+CRF structure is also a potential candidate to achieve better performance than the previous work done by Peng and Dredze (2015), Peng and Dredze (2016) and Cao et al. (2018) in Figure 3 and Figure 4. But, as we have observed, more efforts are needed to adjust the details of the model to close the gap between dev set performance and test set performance.

5 Summary and Future Directions

Although NER has been thoroughly studied, the social media and Chinese language setup demonstrated in this report still pose a challenge for this task. Unlike news reports or other language data, social media consists of many informal grammatical structures and expressions. Unlike many Indo-European languages, there is no "capitalized letters" that can serve as indicators for named entities in Chinese. Despite these challenges, various models involving deep neural networks, such as the ones in previous sections, have been proposed and led to improved performance. However, there is still a lot of room for further study and investigation. Some of the possible directions are listed below.

The first direction is the embedding construction. In my implementation, the embeddings are pretrained on training set and there are several unknown characters encountered during development and test stage, which lowers the performance. All the recent pretraining techniques like ELMo and BERT, as well as the combined embeddings, such as character+position, character+POS tag, can also be experimented at here to provide more information of the characters. However, it is also worthy to notice that, with more complex embeddings, it might become hard to fine-tune and fit into this specific setup.

The second direction is the neural network architecture. In my construction of the model as well as the models proposed by different papers, various blocks such as LSTM, CNN, self-attention, transformer, transfer learning and adversarial learning have been constructed and combined together. It is worthy to investigate which structure actually helps to boost the performance, and which structure does not bring much improvements. In this report, as well as more recent research on this dataset, it is generally sufficient to improve the performance only using the sentences in this data, without extensive transfer learning from other datasets. This will help us focus more on how to develop the models that can best extract the information and make named entity predictions based on the given data.

Another direction is the connection between different neural network blocks and training regimes. Different combinations of learning rates, batch sizes, activation functions, drop out layers, batch normalization, skip-connection, concatenation and other techniques can be experimented. Some of them might have significant impact on the performance while others might not. In addition, beside the first direction's approach to make more comprehensive embeddings, another technique is to specifically set a certain percentage of characters as unknown during training. This might make the model more robust against unknown characters during development and test stage, but the setup needs to be very specific since the unknown characters might be too generalized and leads to harder learning and slower convergence during training.

There are definitely more options beyond these directions. With the combination of these techniques, it is possible to push further and provide a more satisfactory result on the task of Chinese social media named entity recognition.

References

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao and Shengping Liu. 2018. Adversarial transfer learning for Chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 182-192.

Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770-778.

Jiazheng Li, Tao Wang and Weiwen Zhang. 2020. An improved Chinese named entity recognition method with TB-LSTM-CRF. In *2020 2nd Symposium on Signal Processing Systems*, pages 96-100.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for Chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 149–155.

Christian Szegedy et al. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1-9.

Ashish Vaswani et al. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.