An Examination of Named Entity Recognition Regarding Embedding, Attention and CRF Components

Jie Lin, Xin Li

[1] University of Sydney, NSW 2006, Australia
jlin9993@uni.sydney.edu.au
xili9316@uni.sydney.edu.au

1 Introduction

NER is mainly to find the names of entities in the text and classify them. NER applications include question answering systems, knowledge graph construction and so on. The difficulty of NER is that sometimes it is difficult to judge whether a word is a Named Entity. And the Named Entity depends on the context. For example, the same noun may be the name of an organization in some contexts, and a name in other contexts.

In this study, we do Named-Entity Recognition (NER) on re3d dataset from Defense Science and Technology Laboratory, U.K.

We first recorded the results of the baseline model, and then started to improve. The first change is the word vector package of word embedding, and a package that is more suitable for dataset is selected. After that, we introduced PoS Embedding to do syntactic analysis of input sentences. At the model level, we experimented with different layers of bi-LSTM, different Attention Strategies and the influence of CRF.

2 Data preprocessing

No Content.

3 Input Embedding

Syntactic includes the rules of grammar and the correct sequence of words in the sentence. Syntactic analysis processes the grammar structure of the sentence, which include subject determination, place prediction of part of speech and so on.

Semantics conveys that how the word has meanings. Meanings including these types: logical, conceptual, connotative, stylistics, affective, reflected collocative and thematic (Redd et al., 2014). Semantic analysis process words and sentences' meanings.

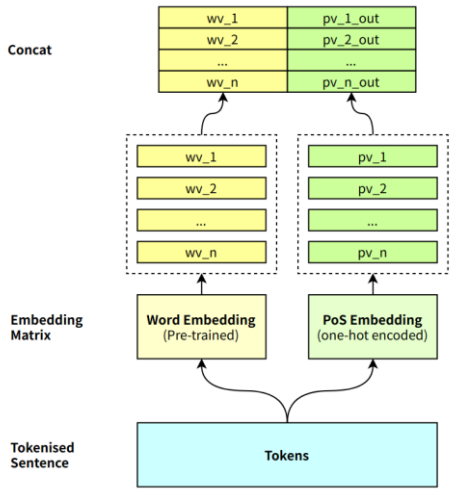


Figure 1 - Illustration of the Embedding Layer

3.1 Syntactic Textual Feature Embedding

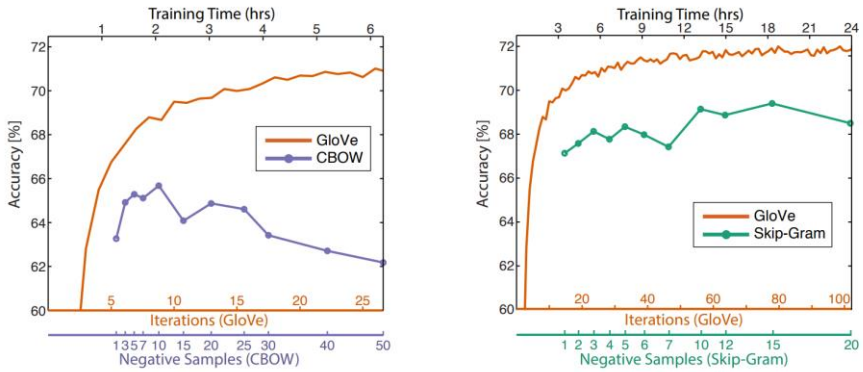
In PoS Embedding, we use the package 'averaged_perceptron_tagger' for tagging words with their part of speech. The package include 45 tag and we use One-hot Encoder to record each word's pos-tag. The reason of using One-hot Encoder of discrete features is to make the distance calculation more reasonable.

3.2 Semantic Textual Embedding

In word embedding, we adapt Global Vectors (GloVe). GloVe is a tool for word representation based on global word frequency statistics. It can express a word as a vector, and these vectors obtain some semantic features between words. Through the operation of the vector, the semantic similarity between two words can be calculated.

3.3 Reason of GloVe

Pennington et al. (2013) did experiment on comparisons of GloVe and word2vec, which found that GloVe performs better than word2vec with same corpus, vocabulary, window size, and training time.



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Figure 2 - Performance of GloVe and word2vec

Since the dataset involves national defence and security, in order to better adapt to this category of corpus, we adopted 'glove-wiki-gigaword-100' as the word vector package. In order to verify our conjecture, another package 'glove-twitter-100' was chosen for experimentation. As can be seen from the picture, the result of using 'glove-wiki-gigaword-100' is better.

	precision	recall	f1-score	support
B-DocumentReference	0.5714	0.2000	0.2963	20
B-Location	0.6467	0.5806	0.6119	186
B-MilitaryPlatform	0.0000	0.0000	0.0000	16
B-Money	0.0000	0.0000	0.0000	5
B-Nationality	0.0000	0.0000	0.0000	8
B-Organisation	0.6965	0.6393	0.6667	280
B-Person	0.8710	0.7941	0.8308	102
B-Quantity	0.5606	0.6727	0.6116	55
B-Temporal	0.7619	0.6809	0.7191	47
B-Weapon	0.1538	0.0526	0.0784	38
I-DocumentReference	0.7778	0.1687	0.2772	83
I-Location	0.7485	0.4604	0.5701	265
I-MilitaryPlatform	0.1250	0.1250	0.1250	16
I-Money	0.8000	0.4000	0.5333	10
I-Nationality	0.0000	0.0000	0.0000	1
I-Organisation	0.7380	0.6298	0.6796	416
I-Person	0.8973	0.7572	0.8213	173
I-Quantity	0.4167	0.1923	0.2632	26
I-Temporal	0.8800	0.7097	0.7857	62
I-Weapon	0.3000	0.0667	0.1091	45
O	0.8473	0.9541	0.8975	3420
accuracy			0.8140	5274
macro avg	0.5139	0.3850	0.4227	5274
weighted avg	0.7963	0.8140	0.7959	5274

	precision	recall	f1-score	support
B-DocumentReference	0.0000	0.0000	0.0000	20
B-Location	0.6400	0.6022	0.6205	186
B-MilitaryPlatform	0.0000	0.0000	0.0000	16
B-Money	0.0000	0.0000	0.0000	5
B-Nationality	0.0000	0.0000	0.0000	8
B-Organisation	0.6978	0.6679	0.6825	280
B-Person	0.8144	0.7745	0.7940	102
B-Quantity	0.6111	0.4000	0.4835	55
B-Temporal	0.7674	0.7021	0.7333	47
B-Weapon	0.3333	0.0263	0.0488	38
I-DocumentReference	0.6875	0.1325	0.2222	83
I-Location	0.7000	0.5547	0.6189	265
I-MilitaryPlatform	0.0000	0.0000	0.0000	16
I-Money	1.0000	0.1000	0.1818	10
I-Nationality	0.0000	0.0000	0.0000	1
I-Organisation	0.7166	0.6442	0.6785	416
I-Person	0.8841	0.7052	0.7846	173
I-Quantity	0.1333	0.0769	0.0976	26
I-Temporal	0.8750	0.7903	0.8305	62
I-Weapon	0.3333	0.0444	0.0784	45
O	0.8466	0.9485	0.8946	3420
accuracy			0.8115	5274
macro avg	0.4781	0.3414	0.3690	5274
weighted avg	0.7871	0.8115	0.7907	5274

Figure 3 - Word Embedding Comparisons of ' glove-wiki-gigaword-100' (Left) and 'glove-twitter-100' (Right)NER model

4 NER Model

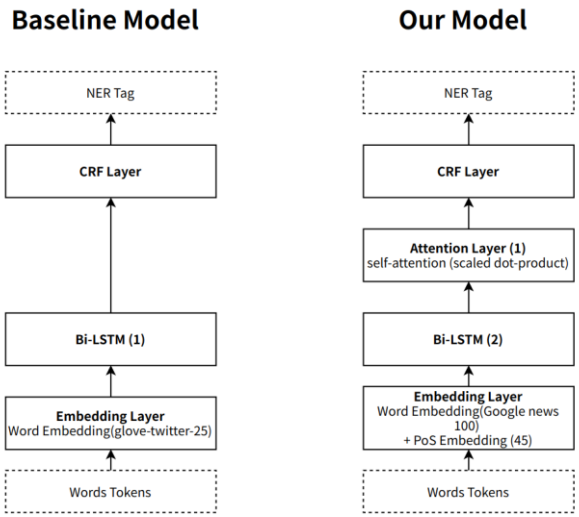


Figure 4 - Comparison of the Baseline Model and Our Model

4.1 Attention Mechanism

The Attention mechanism were introduced to capture the contextual information of the words in a sentence. It can compute the weights that a word contributes to another without worrying about the sequential order. However, it could be its lamination, which does not remember the order of the word occurrence. In our model, we aim to combine RNN with Attention mechanism to utilise the benefits of them.

We design the following experiment to examinate where to place to attention in the model, as shown in Figure 5. Starting from the baseline model, we first replace LSTM model with Attention. Secondly, we stack Attention layer after LSTM model, which is the most common approach in practices. Lastly, we feed the input vector into LSTM and Attention layer separately, then combine they using element-wise dot product.

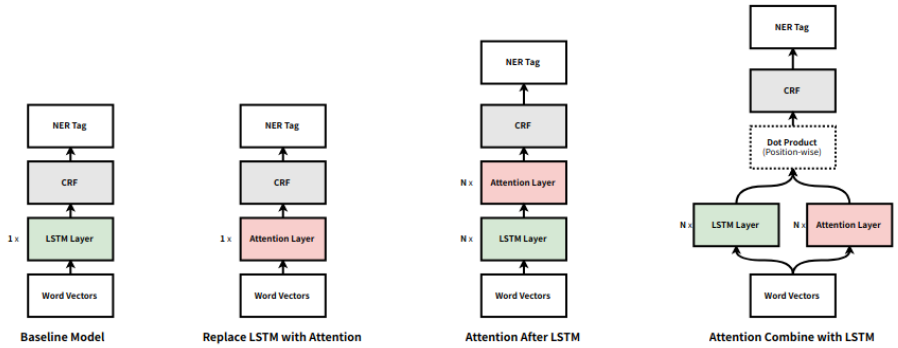


Figure 5 - Different Location of Attention Appeared in Models

From the experimental results we can see that "Attention Layer after LSTM Layer" performs best, followed by the "Attention Combine with LSTM" one. The second one performs worst, since it missed the sequential information of a sentence which can be important for Named Entity Recognition task. By combining LSTM and Attention, the models are better record the order of words, as well as the relationship between these words without worrying about the location.

Model	**F1**
Baseline Model	0.7631
Replace LSTM with Attention	0.7454
Attention After Seq2Seq Model	**0.7860**
Attention Combine with Seq2Seq Model	0.7706

Table 1 - The Performance of Different Attention Mechanisms

4.2 Attention Calculation Methods

In Self-attention calculation (Vaswani et al, 2017), all the Query, Key and Value vectors are identical to the input word vectors. Three matrices are introduced to ‘re-member’ the parameters during training. The computation flow inside the Attention Block is:

1. Compute the outputs of Key, Query and Value

$$K_{out} = M_K K \ , \ Q_{out} = M_Q Q \ , \ V_{out} = M_V V$$

2. Compute the Attention Score

Dot-product:

$$S = K_{out} Q_{out}$$

Scaled Dot-product:

$$S = \frac{K_{out} Q_{out}}{\sqrt{n}}$$

,

where n is the dimension of input vectors.

Cosine Similarity:

$$S = \frac{K_{out} Q_{out}}{||K_{out}|| * ||Q_{out}||}$$

3. Normalise the Attention Score

$$W = F_{softmax}(S)$$

4. Compute the Attention Output

$$Attention = V_{out} W$$

One Attention Block can be stacked over another, using the attention outputs from the former layer as the input vectors.

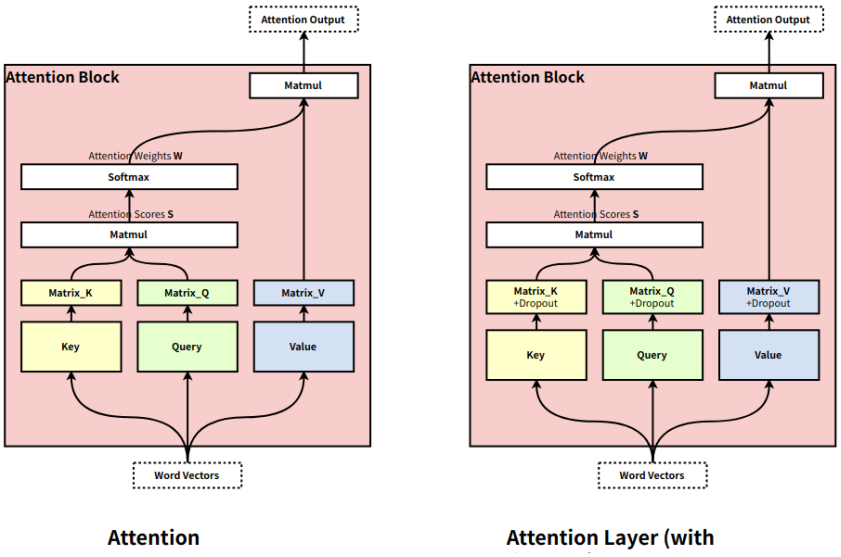


Figure 6 - Computation graph inside the Attention Block

In the experiment, we find that the extra attention parameters tend to cause over-fitting problem during training. In other words, with the attention, the training accuracy and f1-score can easily reach 98%, but fail to generalise the sample from train set. Therefore, dropout layers are added before computing the Attention Score. In the experiment, we find that dropout rate in range 0.3~0.4 would be optimal choices.

5 Evaluation

5.1 Evaluation setup

Input Embedding	Epochs	Learning Rate	Weight Decay	Optimiser
100 + 45	50	0.001	0.0001	SGD

Table 2 - Hyper parameters used for training

The input embedding consists of two parts, word embedding and PoS tag embedding. The baseline model only contains word embedding. For all training task, we set the learning rate with a relatively small number and train with 50 epochs. With this experimental setting, we find that the loss and accuracy can finally converge.

We train the models in Colab. Each run takes about 1 hour with CPUs.

5.2 Evaluation Result

5.2.1 Performance Comparison

The experiment shows that our model performs better than the baseline model, as more syntactic and contextual information are kept in the model. From the classification report we find that some tags that do not recognized by the baseline model can be classify by our model.

	precision	recall	f1-score	support
B-DocumentReference	0.0000	0.0000	0.0000	20
B-Location	0.6176	0.5645	0.5899	186
B-MilitaryPlatform	0.0000	0.0000	0.0000	16
B-Money	0.0000	0.0000	0.0000	5
B-Nationality	0.0000	0.0000	0.0000	8
B-Organisation	0.6833	0.5857	0.6308	280
B-Person	0.8876	0.7745	0.8272	102
B-Quantity	0.5789	0.2000	0.2973	55
B-Temporal	0.7714	0.5745	0.6585	47
B-Weapon	0.0000	0.0000	0.0000	38
I-DocumentReference	0.0000	0.0000	0.0000	83
I-Location	0.6599	0.4906	0.5628	265
I-MilitaryPlatform	0.2000	0.0625	0.0952	16
I-Money	1.0000	0.1000	0.1818	10
I-Nationality	0.0000	0.0000	0.0000	1
I-Organisation	0.6935	0.5601	0.6197	416
I-Person	0.9185	0.7168	0.8052	173
I-Quantity	0.3636	0.1538	0.2162	26
I-Temporal	0.7679	0.6935	0.7288	62
I-Weapon	0.0000	0.0000	0.0000	45
O	0.8175	0.9506	0.8790	3420
accuracy			0.7912	5274
macro avg	0.4267	0.3061	0.3377	5274
weighted avg	0.7495	0.7912	0.7631	5274

	precision	recall	f1-score	support
B-DocumentReference	0.5714	0.2000	0.2963	20
B-Location	0.6467	0.5806	0.6119	186
B-MilitaryPlatform	0.0000	0.0000	0.0000	16
B-Money	0.0000	0.0000	0.0000	5
B-Nationality	0.0000	0.0000	0.0000	8
B-Organisation	0.6965	0.6393	0.6667	280
B-Person	0.8710	0.7941	0.8308	102
B-Quantity	0.5606	0.6727	0.6116	55
B-Temporal	0.7619	0.6809	0.7191	47
B-Weapon	0.1538	0.0526	0.0784	38
I-DocumentReference	0.7778	0.1687	0.2772	83
I-Location	0.7485	0.4604	0.5701	265
I-MilitaryPlatform	0.1250	0.1250	0.1250	16
I-Money	0.8000	0.4000	0.5333	10
I-Nationality	0.0000	0.0000	0.0000	1
I-Organisation	0.7380	0.6298	0.6796	416
I-Person	0.8973	0.7572	0.8213	173
I-Quantity	0.4167	0.1923	0.2632	26
I-Temporal	0.8800	0.7097	0.7857	62
I-Weapon	0.3000	0.0667	0.1091	45
O	0.8473	0.9541	0.8975	3420
accuracy			0.8140	5274
macro avg	0.5139	0.3850	0.4227	5274
weighted avg	0.7963	0.8140	0.7959	5274

Figure 7 - Classification report of the baseline model (left) and our model (right)

Model	F1
Baseline (LSTM + CRF)	0.7631
Ours (LSTM+Attention+CRF)	0.7983

Figure 8 - The f1-score of the baseline model and our model

5.2.2 Different input Embedding Model

In this section, we aim to find out whether the changes of input embedding, either the embedding dimension size or embedding type could affect the performance. The first one is the baseline model, with glove-twitter-25 pretrained embedding in 25 dimensions. The second one and additional PoS tag Embedding to the baseline model and find that the result is slightly better. The third one change the word embedding of the baseline model to higher embedding dimension using glove-twitter-100. Generally speaking, higher embedding dimension and with PoS tag embedding help increase the performance of the model. However, there is a limit.

Model	F1
Word Embed (glove-twitter-25) + LSTM + CRF	0.7631
+ PoS Tag Embed	0.7790
Word Embed (glove-twitter-100) + LSTM + CRF	**0.7860**
+ PoS Tag Embed	0.7836

Table - the F1-score of the Model using Different Input Embedding.

5.2.3 Different Attention Strategy

From Table, the model with attention strategy being 'Cosine Similarity' has slightly higher f1-score.

Attention Strategy	**F1**
Dot	0.7931
Scaled Dot	0.7959
Cosine Similarity	0.8001

Table 3 - F1-Scores from different Attention strategy

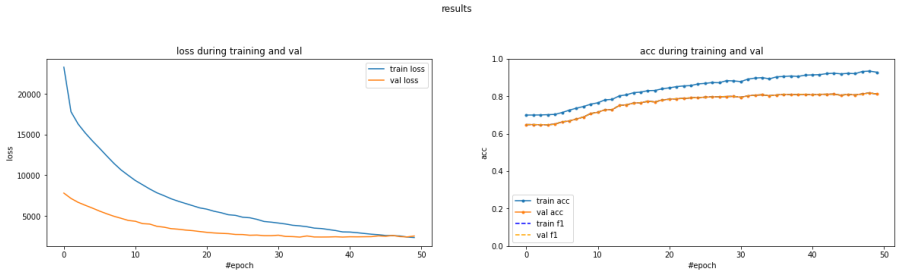


Figure 9 - Result of Attention Strategy being 'Dot'

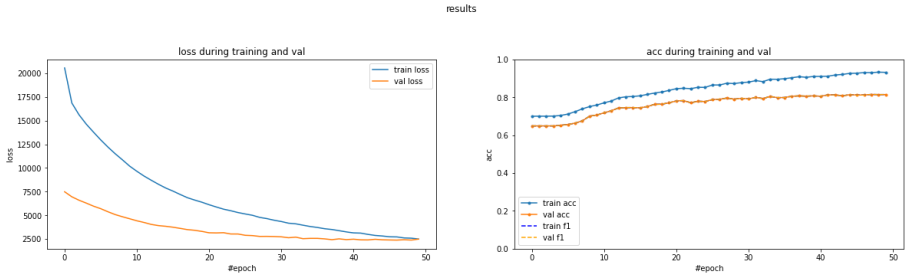


Figure 10 - Result of Attention Strategy being 'Scaled-Dot'

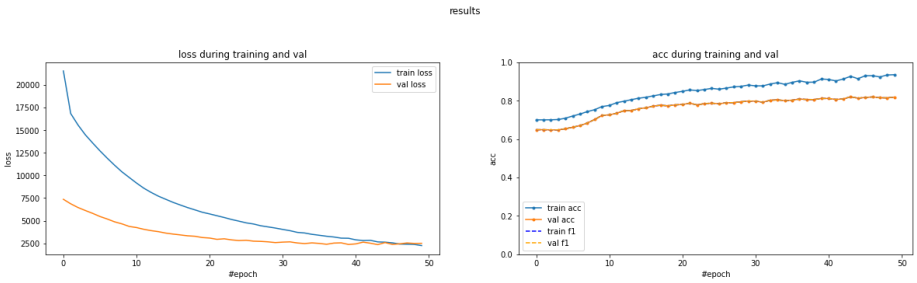


Figure 11 - Result of Attention Strategy being 'Cosine Similarity'

5.2.4 Different Stacked Layer

From Table, the model with 1 LSTM layer has slightly higher F1-Score.

Layer Type	Num of Layer	F1
LSTM	2	0.7959
LSTM	1	0.8028
LSTM	4	0.7479

Table 4 - F1-Scores from Different Number of LSTM Layer

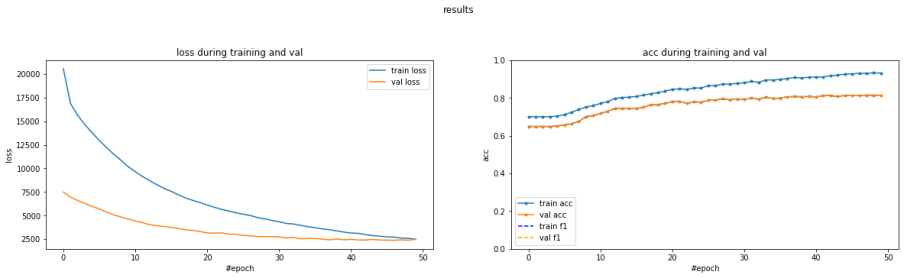


Figure 12 - Result of Number of LSTM Layer being 2

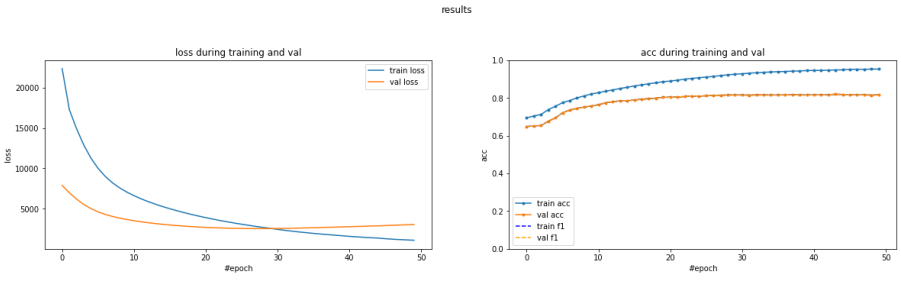


Figure 13 - Result of Number of LSTM Layer being 1

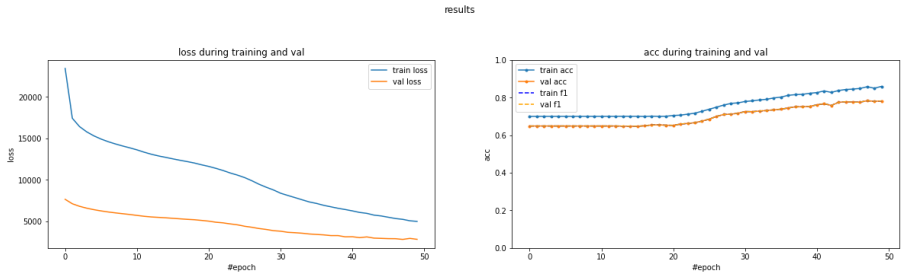


Figure 14 - Result of Number of LSTM Layer being 4

5.2.5 With/Without CRF

CRF are used to decode the co-occurrence of NER tags. In other words, find out the most likely sequence of output tags given input features. From the experiment we can see, without CRF layer, the loss ceases to decrease after several epoch, and the accuracy and f1-score do not increase over the training period. However, with CRF layer, the model is able to record the entire output features from LSTM layer as a whole and to infer the optimal path or sequence of tags.

Model	**F1**
Word Embed (glove-twitter-25) + LSTM	0.5104
Word Embed (glove-twitter-25) + LSTM + CRF	0.7631

Table 5 - Comparison of the F1-score of Models with and without CRF

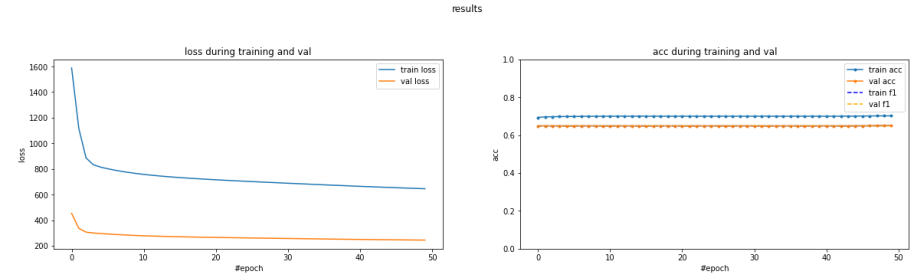


Figure 15 - The Training Log of the Model without CRF Layer

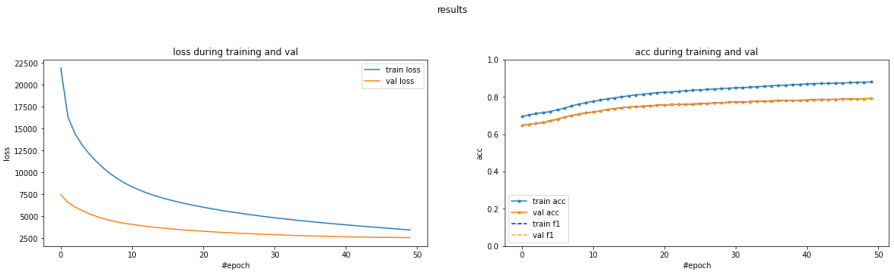


Figure 16 - The training log of the model with CRF layer

6 Conclusion

In this report, we purpose a model with attention mechanism that can outperform the baseline model.

In the first section, we compare and contrast the models with different input embedding and find that larger word embedding dimension tend to perform better.

For the model design, we compare different attention mechanism as well as three different attention calculation methods. The performance differences of the calculation methods are minor.

Attention does well in capturing the contextual information. However, the complex model and might cause overfitting problem easily and thus hyper parameters should be chosen carefully. In addition, Dropout is applied to reduce overfitting problem.

References

1. Mallamma V. Redd, Hanumanthappa M (2014). Semantical and Syntactical Analysis of NLP. *(IJCSIT) International Journal of Computer Science and Information Technologies*, Vol. 5 (3), 3236 - 3238

2. Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.3115/v1/d14-1162

3. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 4-9 December 2017, Long Beach, CA, USA, pages 5998–6008.

4. Zhang, K., Ren, W., & Zhang, Y. (2018). *Attention-Based Bi-Lstm for Chinese Named Entity Recognition*. Springer International Publishing.