Named Entity Recognition Using Bi-LSTM and CTF

COMP5046 A2 Group 46: Songlin Liu[1] and Xingxiang Luo[2]

University of Sydney, NSW 2006, Australia
sliu3056@uni.sydney.edu.au
xluo5082@uni.sydney.edu.au

Abstract

This project works on implementing a Bi-LSTM CTF model with attention strategies to solve a Named Entity Recognition (NER) task for the defense corpus. Input embeddings benefits from word level information and Part-of-Speech (PoS) labels. Our work is the first to experiment bidirectional LSTM (BI-LSTM) for the NER task. Attention is then applied but does not show significant improvement. Conditional Random Field (CRF) layer is extended and shows that it could capture dependencies between labels and improves the performance of NER model. Our system is commendable on the RE3D dataset and shows a significant performance improvement than a baseline model.

1 Data preprocessing

Dataset used is RE3D (DST 2017) which consists of 1,009 sentences and is relevant to the defense and security analysis domain. It contains 10 kinds of entities, including Document Reference, Location, Military Platform, etc.

Data preprocessing techniques such as contraction expansion and punctuation removal are not used as they would change sequence length, adding complexity to a N-to-N classification task like NER. The following is applied to shorten long sentence:

If a sentence exceeds MAX_LENGTH, we split it to multiple sentences with overlapping. Long sentence of size more than 64 are cut into sentences of size 64 with 20 characters overlapping. This is to reduce the span of sentence length to capture better syntactic embeddings.

2 Input Embedding

The final input embedding has 60 dimensions consists of the following features.

- **Semantic:** Word2Vec outputs from Glove-wiki-50 (50 Dimensions)
- **Syntactic:** Part-of-Speech (PoS) Tag from Spacy (4 Dimensions)
- **Syntactic**: Dependency Parsing from Spacy (4 Dimensions) **(Excluded)**

- **Domain Feature:** Casing feature outputs (6 Dimensions) describing

    1. Start with capital letter
    2. All capital letters
    3. All lower case letters
    4. All digits
    5. Mix of words and digits
    6. Contain hyphen '-'

Justification for word-level embedding: Glove-wiki-50 chosen, BERT excluded

Word embedding feature is obtained by using a pretrained model from Glove-wiki-50. Glove Skip-Gram and BagOfWord embeddings are assumed as effective in sematic tasks. These models learn geometrical encodings (vectors) of words from their co-occurrence information. As such, we tried to include such a count-based vector representation by testing Glove-tweet-50, Glove-tweet-100, Glove-wiki-50, and Glove-wiki-100. Our best model use glove-wiki-50 as it works best as the optimised embedding source to provide an effective pretrained word embedding for our task.

A prediction-based vector representation for the word embedding is also tried but not included. The best model is tested by including a word embedding obtained using a pretrained model 'bert-large-cased'. BERT is a SOTA pre-trained model with proven and robust results for improving the model performance of any NLP tasks. However, based on the ablation studies, inclusion of BERT embedding, does not improve our best model accuracy but significantly undermines the performance by more than 2%. Moreover, inclusion of BERT embedding (dimension 1024) slows down the model and thus is excluded in our final model.

Justification for Sentence-level embedding: PoS + Case chosen, DP excluded

Word level embeddings such as vector representation of PoS tags and casing features are concatenated for each word, and the concatenated vector are put together to construct a sentence-level embedding for each sentence. PoS tag, case feature and dependency across label should be included because such meaningful information provides syntactic links to help the NER task in this assignment. Note that the PoS and DP information is converted to vector via Pymagnitude.

The PoS tags are supposed to be useful as words appearing after words such as "with" and "in" are more likely to be tagged as Beginning of a noun entity such as B-Organisation or B-Location. Dependency information provides some coreference relationship between the named-entity and is supposed to improve the NER model's accuracy. The casing feature is supposed to be useful as word that starts with capital letter, or all capital letters are more likely to be tagged as named entity. Ablation study shows that, inclusion of PoS and case features improves the performance of our model whereas dependency does not improve performance and is thus excluded.

3 NER model

- Baseline model: NER model (Bi-LSTM and CRF) provided in the Lab(09)

Our model is based on the baseline model. The model consists of four components: a word embedding (Layer 1), a BI-LSTM layer (Layer 2), an attention layer (Layer 3) and a CRF layer (Layer 4).

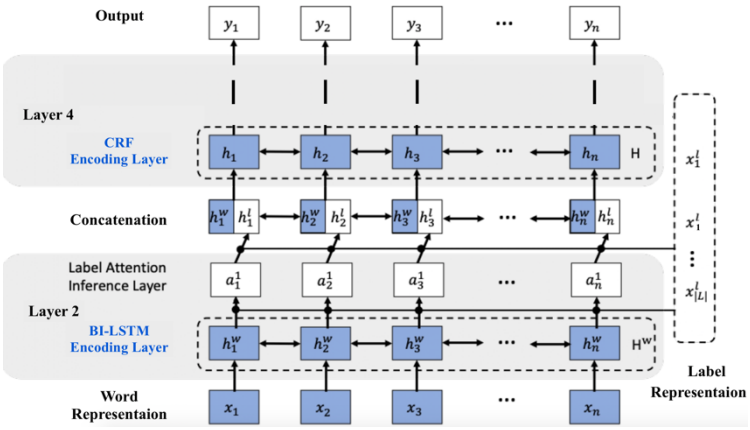

Figure 1: Architecture of the model

3.1. Stacked Seq2Seq model

The BI-LSTM model is a well-studied solution as a neural network to take vast amount of context (Graves et al., 2013). For the input in BI-LSTM, we use pre-trained word embedding from Glove as word-level features and joined vectors as sentence level features. Pre-trained word embedding provides the distributional evidence of words, and other pretrained vector represents PoS tags and case features in a vector space. This layer is coupled with a BI-LSTM (Layer 2) on top of it to generate a context-based representation of words.

Stacked LSTMs is considered as a stable technique for this sequence prediction problems. We tried the Stacked LSTM architecture by stacking the LSTM model to get multiple LSTM layers. Addition of LSTM layers adds levels of abstraction of input observations over time and provides a sequence output rather than a single value output, producing one output per input time step, rather than one output time step for all input time steps. Such addition allows for greater model complexity and is supposed to produce better accuracy using deeper learning architecture.

However, single-layer BI-LSTM is found to be optimum within 10 epochs. The optimal number of stacked BI-LSTM is found by testing 1 layer, 2 layers and 3 layers of BI-LSTM. One possible reason that single layer outperforms others is that multiple-layer neural network models may overfit on our current input sequence and parameters setup. Given that all evaluations results were produced based on the same hyper-parameter setting, it would be unfair to conclude deeper Bi-LSTM perform worse than single-layer Bi-LSTM. However, due to time constraint, we are unable to identify optimum hyper-parameter settings for every layer, so the single-layer Bi-LSTM architecture was adapted in the final model.

3.2 Attention

The output from the last layer of BI-LSTM contains aggregated information from the input sequence, and self-attention ((Karim, 2019)) with 3 calculation methods are applied respectively to represent the importance of each token (with respect to the NER tasks) in the input sentence. Attention output is concatenated with output from last layer of BI-LSTM to produce the inputs to CRF. Final CRF layer uses the input produced by the concatenation to predict the correct label sequence.

Self-calculation (Tamura, 2021) is used in our attention layer (Layer 3). It is an attention mechanism relating different positions of a single sequence to compute a representation of the same sequence. It is good at modelling dependencies between words and is used to understand the syntactic function between words in the sentence.

Different calculation methods are used and the self-attention with scaled dot-product calculation is illustrated as below:

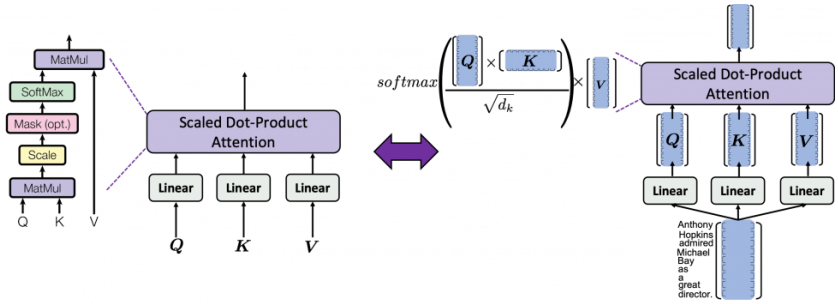

Figure 3: Illustration of attention using dot-product calculation (Tamura, 2021)

Attention score calculation methods used:

Dot Product：This is the simplest of the functions to produce the alignment score. We take the hidden states of the encoder (LSTM output) as Q, K, V and compute the attention score as

$$Score = (Q \cdot K^T) \cdot V$$

Scaled Dot Product：This is similar to dot product to produce the alignment score, but shows better improvement than dot product. For large values of dimension of K, d_k, the dot products grow large in magnitude and the SoftMax function is applied to produce regions of extremely small gradients. To counteract this effect, we scale the dot products by $\sqrt{d_k}$. We take the hidden states of the encoder (LSTM output) as Q, K, V and compute the attention score as

$$Score = \frac{Q \cdot K^T}{sqrt(dimension\ of\ K)} \cdot V$$

General Attention: We can also define attention as managing and quantifying the interdependence between the input and output elements instead of input only (self-attention). We take the hidden states of the encoder (LSTM output) as $H_{encoder}$ and target output as $H_{decoder}$ and compute the attention score. It is also similar to the dot function, except that a weight matrix is added into the equation as well.

$$score_{alignment} = W(H_{encoder} \cdot H_{decoder})$$

3.3 CRF Attachment

Attention output is concatenated with output from last layer of BI-LSTM to produce the inputs to CRF. Final CRF layer uses the input produced by the concatenation to predict the correct label sequence.

CRF is used to make use of neighbor tag information in predicting current tags and our experiment shows that inclusion of CRF shows improvement in accuracy. CRF allows the model to focus on sentence level instead of individual positions by maximizing probability of certain features to reduce the likelihood of unreasonable tagging and ensure our desired logic is preserved. It is accurate as input include semantic information, features and logic.

4 Evaluation

4.1 Evaluation setup

4.1.1 Hyper-Parameters

The model is trained using back-propagation algorithm. Parameter optimization is performed with stochastic gradient descent (SGD) with a learning rate of 0.01. Adam may converge faster but is not used as we select our hyper-parameters based on the performance of the development dataset, accuracy.

The optimal values of the hyper-parameter chosen for the final evaluation are shown at right. We used a naïve approach to select the learning rate. We tried a few different values 0.1,0.05,0.001 and see which one gives us the best loss without sacrificing speed of training. The optimal learning rate is still 0.01. The rest hyper-parameters are also selected based on the performance of the dataset and overall accuracy. Note that our ablation study uses the same hyper-parameters throughout this project.

```
Epochs: 15 with early stopping

Learing rate: 0.01

Weight decay: 1e-4

Optimiser: SGD

LSTM dropout: 0.1

Hidden dimensions: 128
```

4.2 Evaluation result

4.2.1 Performance Comparison

Models	Accuracy
Base model (lab 9)	0.82006
Our model:	0.837737

Table 2: NER results for the (part of) re3d dataset from Defense Science and Technology Laboratory. The model is trained without the use of external labeled data.

The table 2 shows the accuracy scores of the lab 9 Bi-LSTM with CRF model and our model the on the re3d dataset from Defense Science and Technology Laboratory. Our model outperforms the baseline model and without using any of the external labeled data.

4.2.2 Ablation Study – different input embedding model

Models	Accuracy
Base model (glove-wiki)	0.812802
Syntactic Textual Feature Embedding: PoS tag information	0.82527
Domain Feature Embedding: case features	0.83904

Table 3: Results of the model using different embedding model, pos refer to Part-Of-Speech tag information and case indicates the casing features.

Table 3 shows the results of different chosen embedding model. The GloVe wiki gigaword 50 pre-trained model with casing features and Part-Of-Speech tag vector outperforms the other combination of word embedding. We used this combination of word embedding for our final evaluation of the final performance of the model. The more appropriate features we have used, the higher accuracy has been produced from the model.

4.2.3 Ablation Study – different attention strategy

Models	Accuracy
Base model (no attention)	0.828247
Dot product:	0.822851
Scaled dot product:	0.837365
General:	0.824898

Table 4: Results of the model using different attention score calculation.

Based on table 4, the scaled dot product attention score function outperforms the other alignment attention score function. The dot product and general score functions perform worse compare with the base model which is not apply attention mechanism om our model. We applied the scaled dot product score function on our self-attention mechanism in our final model.

4.2.4 Ablation study – different stacked layer

Models	Accuracy
Base model (1 layer)	0.838854
2 Stacked layer:	0.8262
3 Stacked layer:	0.821362

Table 5: Results of the model with different stacked layer

We have test two different numbers of stacked layers which is 2 layers and 3 layers. Based on table 5, the 1 stacked layer performs best. And the 2 and 3 layers performs much worse and requires more training time. We used 1 stacked layer for our final model.

4.2.5 Ablation Study – with/without CRF

Models	Accuracy
Base model (with CRF)	0.839784
Without CRF:	0.754001

Table 6: Results of the model with or without CRF

Table 6 shows the result of our model with or without CRF, it's clear that CRF attachment can improve the accuracy a lot. The CRF attachment has been used in our final model.

Conclusion

In retrospect, we propose a neural architecture for NER based on the BI-LSTM with CRF. The results show that our model outperforms the baseline model and without any external labeled data. Moreover, the evaluation conclude that extra embedding and self-attention mechanism can improve the overall performance.

References

Alan Graves, Abdel Rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 6645–6649.

Tamura, Y., & Tamura, Y. (2021, April 7). Multi-head attention mechanism: “quries”, “keys”, and “values,” over and over again. Data Science Events. https://data-scienc-blog.com/blog/2021/04/07/multi-head-attention-mechanism/

Karim, R. (2019, November 18). Illustrated: Self-Attention. Towards Data Science. https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a