Named Entity Recognition

Taewan Kim[1] and Yeseul Yoon[2]

[1] University of Sydney, NSW 2006, Australia
tkim0934@uni.sydney.edu.au

yyoo0548@uni.sydney.edu.au

1. Data preprocessing

Except building each index set for unique words, NER tags and PoS tags, no further data processing was applied, since the provided data were already lower cased, and split by space for contractions. NER tags were extracted from the provided train and validation data. POS tags were extracted from the 'en_core_web_sm' model of spacy, which is used as PoS tagger for PoS tag embedding discussed in 2.3 section.

2. Input Embedding

2.1. Word Embedding

As what 'GloVe' represents - Global Vectors, GloVe captures not only separate local statistics but also utilises global statistics. Global statistics stand for counting global co-occurrences. Using both local and global statistics it eventually comes up with word vectors. While Word2vec obtains vectors using only surrounding words and works well with the frequent words only, GloVe underlines the words' probability of co-occurrence in the corpus when building word vectors. FastText has a similar approach with Word2vec but breaks down the words into sub-words when obtaining word vectors using each character that composes the words. In the preliminary study, validation F1 scores of the test model with pre-trained Word2vec (word2vec-ruscorpora-300), FastText (fasttext-wiki-news-subwords-300), and GloVe(glove-twitter-100) on given dataset were 0.9711, 0.9714, and 0.9743, respectively. Since it has the highest performance as well as efficiency in its relatively small 100 dimension, GloVe was chosen as our word embedding model. After building a projection matrix with GloVe for the word set built in the previous data preprocessing, the matrix was loaded as an initial value of the word embedding layer so that it can be fine-tuned during training.

2.2. Contextual Embedding

Since word embeddings are not designed for comprehending contextual meaning of words, it is assumed that adopting contextual embedding along with word embedding brings performance improvement. ELMo, BERT and Contextual String Embedding (Flair) were considered to adopt in our model. First, ELMo is one of the most commonly used contextual embedding algorithm that uses bi-directional LSTM, so it has an ability to see both previous and following word. BERT works with bi-directional transformers and uses the mask - the masked word's position is used for prediction of the masked word. Flair uses trained character model to generate word embedding. Among ELMo (small and medium), BERT, and Flair embeddings that were provided by Flair package, only ELMo(small) was successfully loaded due to the memory deficiency in Google Colab environment. From the pre-trained model, one contextual embedding for each word in sentences in train, evaluation and test set was extracted and saved in train, evaluation, and test dataset file, respectively. In training, the contextual embedding train dataset is used as it is without fine-tuning for feasible implementation in Colab.

2.3. Part-Of-Speech(POS) Tagging

PoS Tagging is one of principal downstream works in natural language applications with respect to grasping relationships between words. It tags each word in given corpus to a corresponding tag by its definition or context, thus a particular word might have different tags according to how it is used. Therefore, we hypothesized that it would also improve the NER prediction performance by a similar rationale to the contextual embedding cases. In this project, 'en_core_web_sm' model provided by spacy was applied as the PoS tagger. This model is a small English model trained on written web text (blogs, news, comments) and includes vocabulary, vectors, syntax and entities[1]. A PoS tag for each word in sentences in train, evaluation and test set was extracted and saved as an indexed form in the same dataset files as the contextual embedding. This indexed PoS tags are provided and used as inputs of the PoS embedding layer in models in this project.

3. NER model

Figure 1 shows the overall structure of the proposed model. It has word, contextual and PoS tag embeddings as its inputs. The input provided to Bi-LSTM layer turned into forward and backward hidden states of each token, which is accepted by Transformer encoder as its input. Each logit that Transformer encoder outputs is utilised as an emission score in log space to generate the NER tag by CRF layer.

3.1. Bi-LSTM

Long Short-term Memory Networks (LSTMs) are clearly built to avoid the long-term dependency problem as standard Recurrent Neural Networks (RNNs) have in practice. Their hidden layers are replaced by specialised memory cells, thus they may work better at detecting long-range dependencies. In this project, rather than using a standard type of LSTM, Bi-LSTM that can capture both the forward and backward contents of the word by a combination of two LSTMs was employed. As the number of layers of networks influences the model's performance, three different numbers of layers from one to three were tested and compared in the ablation study to find the optimal number, which was one layer based on F1 scores. In addition, positional encoding is critical for Transformer to detect the different meanings of embeddings depending on where they are used in sentences. Since Bi-LSTM has sequential information inherently in its hidden states, no additional positional encoding is used in this project for Transformers having the hidden states from Bi-LSTM as its inputs.

3.2. Transformer Encoder

Transformer is capable of processing time sequence data without using RNN and LSTM. One of the components that Transformer has is a multi-head attention layer that produces different self-attention using multiple heads. Utilising the multi-head self-attention mechanism is expected to be a complementary solution to the possible issue of long-term dependency problem that could possibly be arisen from Bi-LSTM. The attention calculation method is further discussed in 3.4 section. The other main component is pointwise feed-forward network that is fed by the outputs of self-attention, and then its weights are applied to the token by token maintaining the dimension of the outputs same as the dimension of the inputs. The residual connection and layer normalisation between the components are also essential to avoid gradient exploding of the deep layered structure. For this project, the number of layers was set from one to three to be compared in the ablation study. As a result, one-layer of Transformer encoder is used in the final model as it has the highest F1 scores.

3.3. Conditional Random Field(CRF)

Conditional Random Field (CRF) is a variation of Hidden Markov Model (HMM) designed for global understanding rather than local understanding, and this network requires comprehending all sentences. Since CRF deals with optimal global output, it does not have strict assumptions that HMM requires or label bias that Maximum-Entropy Markov Model (MEMM) might suffer by computing the probability based on each state at each time. Despite its computational complexity, CRF leads the model to improve NER performance by utilising both of emission probabilities and transition probabilities that the previous layers do not support. Unlike the baseline model where CRF layer is stacked directly above Bi-LSTM layer, Transformer encoder layer is sandwiched between Bi-LSTM and CRF to enhance the performance, sending its outputs to CRF model as the emission scores.

3.4. Attention

In this project, two types of attention has been tested as followings.

• Dot-product ($score(s_t, h_i) = s_t{}^T h_i$)
Among several methods to calculate attention score, dot-product attention is considered intuitive and understandable, and one of the two most commonly used attention functions along with additive attention[2] ($score(s_t, h_i) = v_a{}^T tanh(W_a[s_t; h_i])$). Here, attention score is simply computed by doing dot-product between queries and keys. While additive attention empirically results in better performance with a small dimension of data, dot-product is considered more suitable in our projects and examined in the ablation study since it is relatively faster with our embeddings with large dimensions, and it does not require additional weights to a plethora of weights the multi-head attention in Transformer already has.

• Scaled Dot-product($score(s_t, h_i) = \frac{s_t{}^T h_i}{\sqrt{n}}$)
Scaled dot-product attention is very similar to dot-product attention aside from its scaling step. Before computing the softmax, the function scales down the value in order to avoid small gradients, using $\sqrt{n}$ where n is the dimension of the hidden state. This yields an improved performance where word embeddings have high dimension. Since given data for this project has relatively large dimension, scaled dot-product function was examined and compared with dot-product attention in the ablation study. The result led the final model to using scaled dot-product attention based on F1 scores.

4. Evaluation

4.1. Evaluation setup

Dataset For this project, only part of the original CoNLL-2003 dataset that is split into training (3000 sentences), validation (700 sentences), and test (3684 sentences) is used.

Embeddings The dimensions of word, contextual, and PoS tag embeddings are 100, 1,536 and 30, respectively. The dimensions of word and contextual embeddings come from the pretrained embeddings. The dimension of PoS tags is arbitrarily chosen as 30 among the numbers less than 50, which is the size of unique set of PoS tags.

Hyper-parameters The default training epoch and batch size are set to 90 and 1000, respectively. The default learning rate and weight decay is 0.0005 and 0.0001, respectively. The default parameters were adjusted in the ablation study regarding attention and layers due to the GPU memory consumption and convergence issues. 500 sized batches and 180 epochs were used in attention and Bi-LSTM layers ablation study, whereas batch size 100 and learning rate 0.0001 was used in Transformer encoder layers ablation study. In terms of Transformer, we followed the work of (Vaswani et al., 2017) to set its heads, model dimension, and feed-forward network dimension as 8, 512, and 2,048, respectively.

Optimiser To choose the optimiser, we followed previous study of of (Reimers et al., 2017)[3] that showed Adam optimiser performed the best over SGD, Adagrad, Adadelta, RMSProp, and Nadam on CoNLL 2003 dataset.

Evaluation Metrics F1 score is used for evaluation per entity as it is the metric used in the Kaggle leaderboard.

4.2. Evaluation result

4.2.1. Performance Comparison

	F1
Baseline	0.9285
Bi-LSTM	0.9721
Transformer	0.9669
Bi-LSTM + Transformer	**0.9738**

Table 1 : Performance Comparison

Looking at Table 1 above that shows the F1 score of three different combinations of our models and given baseline, the model has both Bi-LSTM and Transformer layer outperformed than others. Hence, we would say that attaching transformer between Bi-LSTM and CRF model helps to elevate the performance from 0.9285 to 0.9738 of F1 score. In addition, the performance comparison graph below indicates that Bi-LSTM + Transformer model already outperforms over the baseline model where epoch is 13.
In the comparison, the original sinusoidal positional encoding in Transformer is used only when Bi-LSTM is not used, which alternatively provides sequential position information to Transformer.

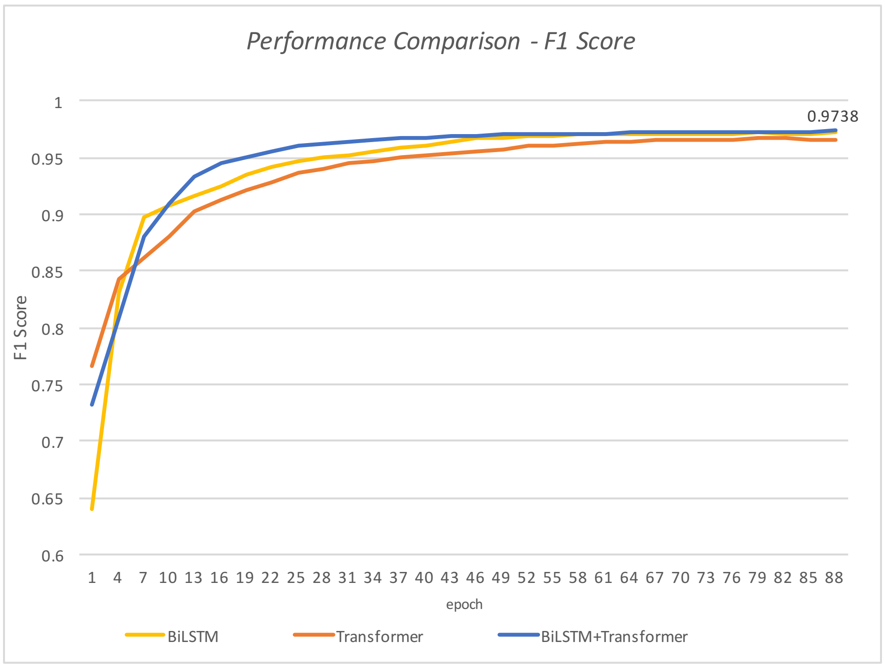

Figure 2 : Performance Comparison - F1 Score

4.2.2. Ablation Study - different embedding model

	F1
Word Embedding	0.9578
Word Embedding + Contextual Embedding	0.9723
Word Embedding + Contextual Embedding + Pos Tagging	**0.9743**

Table 2 : Embedding Comparison

In this experiment, we compared the F1 scores of word embedding itself, word + contextual embedding, and word + contextual + PoS tagging. As the results in Table 2 shown above, we discover that when contextual embedding is added to provide the contextual meaning in the sentence, the model is improved significantly. The graph below illustrates how F1 scores are different on three different combinations of embedding model. Although the influence of PoS tag embedding seems not significant, it still gives the best result when three embeddings are used together.

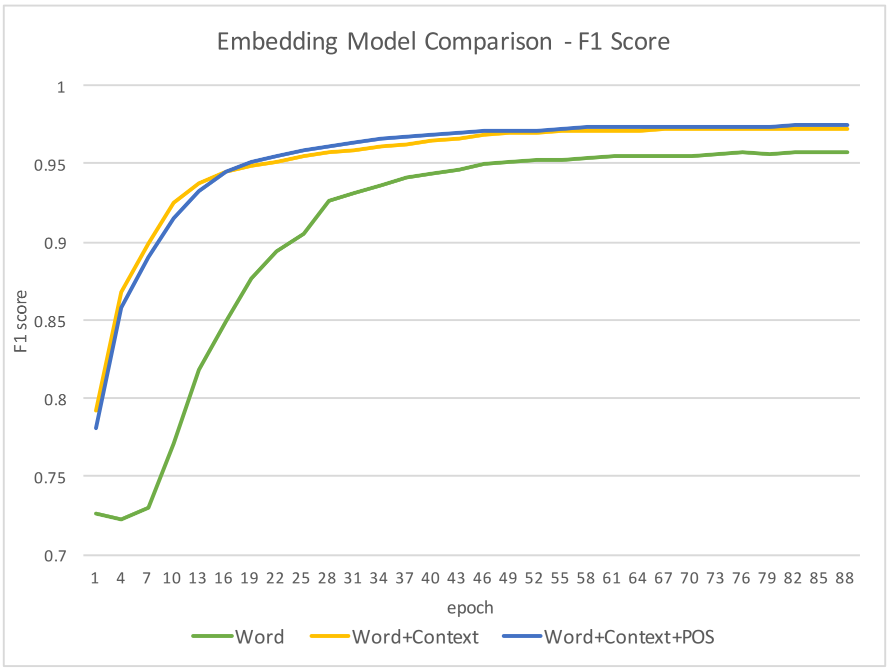

Figure 3 : Embedding Comparison - F1 Score

4.2.3. Ablation Study - different attention strategy

	F1
Dot-Product	0.9734
Scaled Dot-Product	**0.9758**

Table 3 : Attention Comparison

To study the influence of attention calculation methods on the model performance, two different attention strategy dot-product and scaled dot-product were applied and examined in the multi-head attention in Transformer encoder in our model. As shown in Table 3 above, scaled dot-product that regulates the dot-product values shows a bit better performance.

4.2.4. Ablation Study - different layer strategy

Number of Bi-LSTM layers	F1
1	**0.9768**
2	0.9746
3	0.9745

Table 4 : Bi-LSTM Layers Comparison

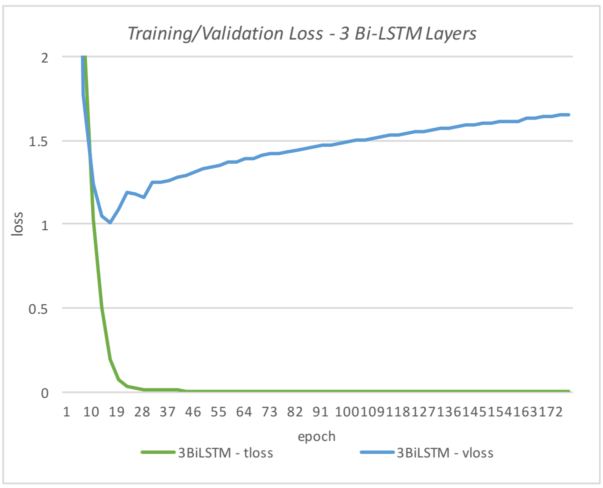

Figure 4 : Training/Validation Loss - 3 Bi-LSTM layers

We further conduct the experiment to compare the number of Bi-LSTM layers of 1,2, and 3. As Table 4 above indicates, when Bi-LSTM consists of one layer only, the model performs the best compared to the model with two and three layers. Furthermore, we come up with over-fitting issue when the model has 3 layers of Bi-LSTM. Looking at the graph (Figure 4) above, while the training loss keeps going down to 0, the validation loss starts increasing at some point.

Number of Transformer layers	F1
1	**0.9735**
2	0.9708
3	0.9655

Table 5 : Transformer Layers Comparison

We also bring up the different number of Transformer encoder layers of 1, 2, and 3. As Table 5 shows, the one layer of Transformer encoder achieves the highest F1 score among three different experiments.

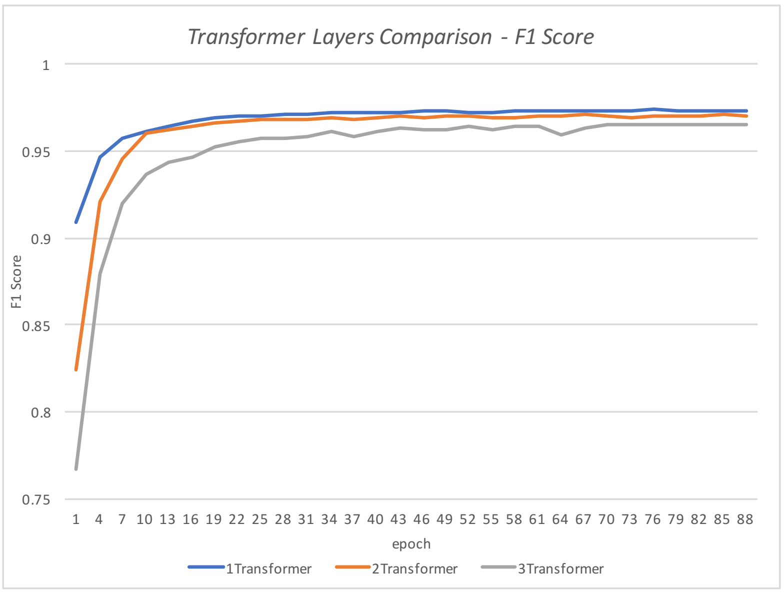

Figure 5 : Transformer Layers Comparison

References

[1] en_core_web_sm. Spacy.io. (2020). Retrieved 2 June 2020, from https://spacy.io/models/en#section-en_core_web_sm.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[3] Reimers, N., & Gurevych, I. (2017). Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. ArXiv, abs/1707.06799.