COMP5046 - Assignment 2

Group 124 - Christopher Veldman and Shuhao Liang

University of Sydney, NSW 2006, Australia
cvel5570@uni.sydney.edu.au
slia7223@uni.sydney.edu.au

1 Data preprocessing

Stemming and lemmatisation are both popular methods for language preprocessing and there is evidence that suggests that they can be used to improve the performance of a Named Entity Recognition (NER) model [2]. We therefore started by evaluating the performance of our baseline model with and without stemming and lemmatisation. Our finding was that both stemming and lemmatisation marginally decreased the F1 score for the given dataset relative to the baseline as shown in Figure 1 below and so we chose not to proceed further with any data preprocessing in our modelling.

Baseline model with no pre-processing:

	precision	recall	f1-score	support
2	0.8012	0.7326	0.7654	187
3	0.9775	0.9919	0.9847	5790
4	0.9698	0.9543	0.9620	875
5	0.9234	0.9212	0.9223	419
6	0.9048	0.7333	0.8101	285
accuracy			0.9674	7556
macro avg	0.9153	0.8667	0.8889	7556
weighted avg	0.9665	0.9674	0.9666	7556

Baseline model with lemmatisation:

	precision	recall	f1-score	support
2	0.8652	0.6524	0.7439	187
3	0.9685	0.9486	0.9584	875
4	0.9739	0.9934	0.9836	5790
5	0.8793	0.7158	0.7892	285
6	0.9238	0.9260	0.9249	419
accuracy			0.9656	7556
macro avg	0.9222	0.8472	0.8800	7556
weighted avg	0.9643	0.9656	0.9642	7556

Baseline model with stemming:

	precision	recall	f1-score	support
2	0.8521	0.6471	0.7356	187
3	0.9646	0.9040	0.9333	875
4	0.9675	0.9936	0.9804	5790
5	0.8493	0.6526	0.7381	285
6	0.8928	0.9141	0.9033	419
accuracy			0.9574	7556
macro avg	0.9053	0.8223	0.8581	7556
weighted avg	0.9557	0.9574	0.9555	7556

Figure 1 - Baseline model classification reports with and without pre-processing

2 Input Embedding

We considered input embeddings constructed from the concatenation of three different feature sets: a word embedding; a part-of-speech (POS) embedding; and a dependency parse tree embedding. We also considered concatenating the vector of TFIDF scores representing each word in the sentence to the input embedding but

found that this did not offer much improvement and was a low dimensional feature set in comparison to the other embeddings - one dimension per word as opposed to several hundred for the word embedding - so was not expected to have much of an influence on the model predictions.

GloVe word embeddings have been shown to achieve the best performance on NER tasks. Reimers, N. & Gurevych, L. [1] evaluate a range of hyperparameters for LSTM sequence labelling tasks and show that for NER tasks, the GloVe embedding of 100 dimensions trained on Wikipedia and Gigaword, is very close to the best performance with an F1 of just -0.73% below the best. We therefore selected this pre-trained word embedding model and implemented it using the gensim library.

POS tagging can be especially relevant to NER tasks, particularly for words or tokens that can have multiple POS tags. These tags can provide additional information that helps to identify a named entity based on their part of speech, For instance, the word "google" can be used as both a noun referring to the named entity that is the technology company as well as a verb referring to the act of using a web search engine and for NER tasks, this context is relevant.

In order to recognise named entities we also expect it to be useful to parse the dependency tree from the top down and identify the noun phrases. The expectation was that by adding these dependency tree and POS features to our word embedding, it should make it easier for our models to identify named entities in the dataset. We use spaCy, a Python-based, open-source, natural language processing library to perform POS tagging and dependency parsing. It is generally believed to produce state-of-the-art accuracy [3] and is considered to be the fastest NLP parser available.

3 NER model

LSTM models show strong performance for many linguistic sequence tagging tasks and are commonly used. When evaluating hyperparameter choices for LSTM sequence tagging tasks, Reimers, N. & Gurevych, L. [1] find that it is the choice of word embedding that tends to dominate most other hyperparameter or LSTM architecture choices in terms of the impact on prediction performance. The authors also find that a conditional-random-field (CRF) classifier is preferred for NER tasks. Other design choices, for example additional feature representations, tagging schemes, the number of LSTM layers and the number of recurrent units only had a minor impact for the evaluated tasks.

We therefore choose to use the popular Bi-LSTM CRF architecture as our baseline model. We allow for further modifications such as different embedding combinations and attention mechanisms, as well as additional hidden layers in the Bi-LSTM network. Figure 2 illustrates the architecture of our BiLSTM-CRF classifier and attention mechanism for application to this NER task.

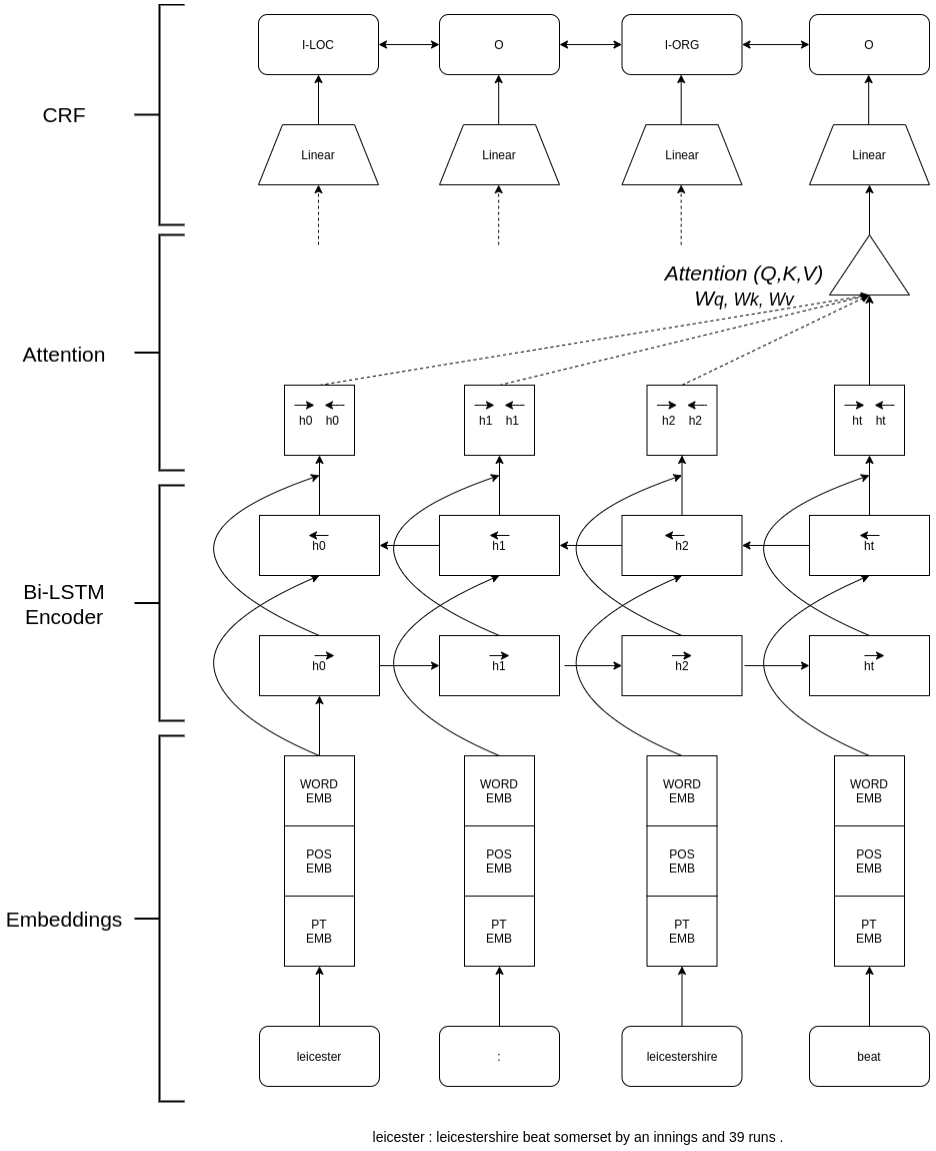

Figure 2 - The NER model architecture adopted for this assignment

Each lower-case word in the input set of sentences is first mapped to a word embedding. We choose the pre-trained GloVe embedding as described in the previous section. This word embedding is then concatenated with zero/any/all of the additional embeddings (POS and dependency parse tree (PT)) before being passed as input to the Bi-LSTM model.

Within the Bi-LSTM model, one LSTM network runs from the beginning of the sentence to the end, while the other runs in reverse. Given that there is little additional benefit to be had in modifying the standard Bi-LSTM architecture [1], we only consider different values for one dimension of the Bi-LSTM hyperparameter space: the number of layers; and we limit our search to a choice between just 1 layer or 2 layers.

The output of both LSTM networks are then used as input for the attention mechanism. We only consider adding two self-attention mechanisms in the evaluation process to limit the model configuration search space, which gives us three options for attention in each model:

1. No attention
2. Dot Product Attention

$$Attention(Q, K, V) = softmax(QK^T)V$$

3. Scaled Dot Product Attention

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Dot-product attention is one of the most commonly used attention functions and so including this in our evaluation seems a natural choice. Following Vaswani et al. [4], we also consider scaled dot product attention where we scale the dot products by $\sqrt{d_k}$ to counteract the effects of small gradients on the softmax function for large values of d_k.

Having applied the attention mechanism, the model then maps the output through a dense layer onto the tag space. For each token in the sentence, we then have a probability distribution for the possible tags. The tag with the highest probability is selected and the CRF maximizes the tag probability of the complete sentence.

4 Evaluation

4.1 Evaluation setup

We adopt two approaches to evaluation and model optimisation. The first is a supervised, directed approach where we selectively test a number of model configurations and embedding combinations based on our prior beliefs about what we expect to perform best. The second approach to evaluation is a full ablation study, where we exhaustively search the full model configuration space that we allow for (described in Section 3).

In the following section we present evaluation results for the top models selected in our supervised, directed search as well as the summary results of the full ablation study.

4.2 Evaluation result

The training data is composed of 3000 sample sentences with their associated NER tags. We train all our models on this set of data before proceeding with the evaluation on the validation data set. The validation data is composed of 700 sample sentences and their associated NER tags.

For the selective model search, Figure 3 shows the best performing model configuration alongside that of the baseline model configuration. The full summary of results for the selective model search are shown in Figure 4 below, where the 7.X labelling corresponds to the section of the accompanying Python Jupyter Notebook where these models are implemented.

The best performing model configuration on the validation set from the selective search is the single layer Bi-LSTM with scale-dot-product attention and just the word embedding. However, on the test set submission to Kaggle our two layer Bi-LSTM with scale-dot-product and all feature embeddings (model 7.7) performs the best.

Models	F1
Baseline	0.968369507676019
Best other model	0.9675754367390154

Figure 3 - Best other model from the selective search is model 7.4

Selective Search - Models	F1
7.1 Baseline	0.9684
7.2 two layers + dot product attention + word embedding + pt + pos features	0.9655
7.3 one layer + dot product + only word embedding	0.9644
7.4 one layer + scale dot product + only word embedding	0.9676
7.5 two layers + scale dot product + only word embedding	0.9665
7.6 two layers + scale dot product + word embedding + pos feature	0.9656
7.7 two layers + scale dot product + word embedding + pt + pos features	0.9631
7.8 two layers + dot production attention + word embedding + pos feature	0.9631
7.9 two layers + dot production attention + only word embedding	0.9669
7.10 one layer + dot production attention + word embedding + pt + pos features	0.9672
7.11 one layer + dot production attention + word embedding + pos feature	0.9666
7.12 one layer + scale dot production attention + word embedding + pt + pos features	0.9623
7.13 one layer + scale dot production attention + word embedding + pos feature	0.9661

Figure 4 - Performance of all the models defined for the selective search

The results for the full ablation study are even more encouraging than the Kaggle result. In Figure 5 below we see one of the model configurations in the search performs better than the baseline model performance.

Models	F1
Baseline	0.9678
Best other model	0.9684

Figure 5 - Best other model from the exhaustive ablation study is model 21

Figure 6 shows the full summary of the ablation study which covers a search space of 24 model configurations in total. The table is arranged in a horizontal tree style i.e. the first 7 models are of type “No Attention” like the baseline with different embeddings and layers; the next 7 are of type “Dot Product Attention” with different embeddings and layers etc.

Model	Attention	Layers	Embedding	F1
Baseline model	No Attention	1 layer BiLSTM	Word Embedding	0.9678
Model 1			+ PoS embedding	0.9636
Model 2			+ Parse Tree embedding	0.9676
Model 3			+ ALL	0.9648
Model 4		2 layer BiLSTM	Word Embedding	0.9668
Model 5			+ PoS embedding	0.966
Model 6			+ Parse Tree embedding	0.9627
Model 7			+ ALL	0.9661
Model 9	Dot Product	1 layer BiLSTM	Word Embedding	0.9659
Model 10			+ PoS embedding	0.9657
Model 11			+ Parse Tree embedding	0.9644
Model 12			+ ALL	0.9632
Model 13		2 layer BiLSTM	Word Embedding	0.9674
Model 14			+ PoS embedding	0.9623
Model 15			+ Parse Tree embedding	0.9641
Model 16			+ ALL	0.9596
Model 17	Scale Dot Product	1 layer BiLSTM	Word Embedding	0.9673
Model 18			+ PoS embedding	0.964
Model 19			+ Parse Tree embedding	0.9652
Model 20			+ ALL	0.9643
Model 21		2 layer BiLSTM	Word Embedding	0.9684
Model 22			+ PoS embedding	0.9653
Model 23			+ Parse Tree embedding	0.9651
Model 24			+ ALL	0.9628

Figure 6 - Performance of all the models defined for the exhaustive ablation study

The best performing model from the full ablation study is Model 21, which is Scale Dot Product attention; 2 layer Bi-LSTM with just the word embedding achieving a total F1 score of 0.9684 above the baseline of 0.9678.

References

1. Reimers, N. & Gurevych, L.: Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks, arXiv:1707.06799v2 [cs.CL], (2017)
2. Konkol, M. & Konopík, M.: Named Entity Recognition for Highly Inflectional Languages: Effects of Various Lemmatization and Stemming Approaches. 10.1007/978-3-319-10816-2_33. (2014)
3. Honnibal M, Johnson M. An improved non-monotonic transition system for dependency parsing. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; 2015 Sep 17-21; Lisbon, Portugal. Stroudsburg. 2015. pp. 1373–1378.
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I., Attention Is All You Need, 31st Conference on Neural Information Processing Systems (2017)