Course Project : COVID-19 Named Entity Recognition

Nov. 28th, 2020

Group 18 (jim)

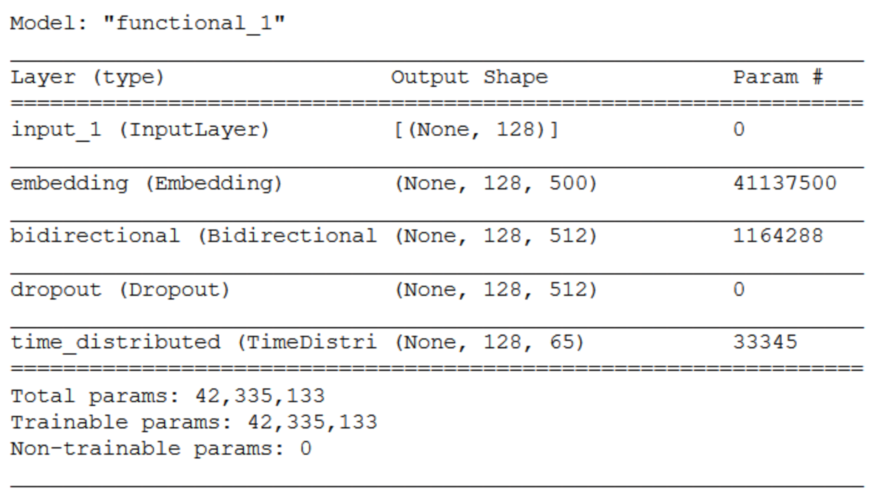
Ching-Yu Lin

clinah@connect.ust.hk

Model

The model contained three main components, including an embedding layer, a bidirectional GRU layer, and a TimeDistributed layer with the optimizer Adam.

- The embedding layer had the input dimension of the vocabulary size of training tokens and the output dimension of 500.
- The bidirectional GRU layer had a hidden size of 512 (sentence length maximum x 2).
- The TimeDistributed layer classified the results into 65 classes (the number of different tag types).



```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128)]             0
_________________________________________________________________
embedding (Embedding)        (None, 128, 500)          41137500
_________________________________________________________________
bidirectional (Bidirectional (None, 128, 512)          1164288
_________________________________________________________________
dropout (Dropout)            (None, 128, 512)          0
_________________________________________________________________
time_distributed (TimeDistri (None, 128, 65)           33345
=================================================================
Total params: 42,335,133
Trainable params: 42,335,133
Non-trainable params: 0
_________________________________________________________________
```

Results & Discussions

- The model obtained the accuracy of 0.9049 on the validation set based on the calculation performed by the “evaluate” function in evaluate.py.
- The model slightly overfitted on the training data. However, it did not affect the performance of the model on the validation set.
    - The accuracy of the model on the training set was 0.9673 based on the calculation of Kera’s evaluate function between the training tokens and training tags.
    - The accuracy of the model on the validation set was 0.9129 based on the calculation of Kera’s evaluate function. The result was slightly better than the models which did not overfit on the training set (from around 0.88, 0.89 to 0.91).
        - The reason for this phenomenon might be related to the tolerance of the large embedding layer since the model had a relatively large embedding layer, and this was also where the overfitting came from.
        - The accuracy rate is slightly different here compared to the result from the “evaluate” function from evaluate.py because, in the “evaluate” function, it removed the padding tokens before the accuracy calculation.

- With the larger embedding layer, observable improvements on the existing model turned out to be possible.
    - The benchmark model throughout my testing is a model with a comparable smaller embedding layer, a bidirectional LSTM layer, and a TimeDistributed layer. The size of this benchmark model's embedding layer is 100.
        - Initially, the size of the benchmark model's embedding layer was 50. After several testing, size 50 and size 100 returned a similar performance.
    - After enlarging the size of the embedding layer to 500, the performance on the validation set accuracy improved from 0.88-0.89 to 0.90-0.91. Although the improvement was small, it was still significant while not changing the main structure of the model.
- MLP model would destroy the word sequence information (the surrounding words), which resulted in an undesirable outcome. The accuracy rate on the validation set was only 0.7548. While for RNN like models, they could easily obtain at least 0.80 accuracy.
- The TimeDistributed layer was critical to classify the output from an output node into a finite and known number of classes (the tags).
    - Without the TimeDistributed layer, the MLP model reported a tragic result with only 0.05 to 0.06 accuracy, which was significantly lower than the naïve 'O' guessing model (0.7562 accuracy).
- Adding another RNN layer (SimpleRNN, LSTM, or GRU) did not help to improve the performance. With only one RNN layer, the model was more successful.
- Directly increasing the size of the hidden part of RNN like layers did not help enhance the models. (The benchmark model's LSTM layer was of size 128.)
- Among the three optimizers, SGD, Adam, and RMSprop, Adam was more suitable for the task. It outperformed the other two optimizers.
- Among the three kinds of RNN elements, SimpleRNN, LSTM, and GRU, GRU was the best one. It gave a slightly better and more stable result than LSTM. SimpleRNN was far behind the other two RNN structures.

Activities & Acknowledgements

- After implementing the MLP model, we obtained 0.7548 accuracy on the validation set. Even with the increased epoch number (5 to 20), there were still no observable improvements.
- We discussed the issues with DING, Shanhe, and he told us that his group had a bidirectional LSTM model with an embedding layer. The accuracy rate of their model was around 0.89. Afterward, we turned to RNN models and set a benchmark model for further testing.
    - The benchmark model consisted of an embedding layer of size 100, a bidirectional LSTM layer with hidden size 128, and a TimeDistributed layer with 65 classes with optimizer Adam.
- We subsequently performed the testing on RNN structure types (SimpleRNN, LSTM, and GRU), optimizers (SGD, Adam, and RMSprop), multiple RNN layers, the RNN hidden size, and the embedding layer size.
    - The notion to increase and somehow abuse the embedding layer size was from a discussion question raised in the Q&A tutorial session.
- We had an idea to apply the sequence-to-sequence model to this task since it could be deemed as a vocab_size(tokens) <-> vocab_size(tag_types) translation problem. However, we did not have time.