Recurrent Neural Models for Sequence Labeling

Natural Language Processing assignment 1

Amir Hajiabadi Molin Zhang Jialing Li

Academic year: 2021/2022

Abstract:

In this assignment we are asked to predict the POS tag for each word, given a corpus of documents. We use four different new models, implemented with Tensorflow, based on Keras, and tested the performances with different input data strategies.

Data preparation

The dataset consists of 199 documents, which are split into training, validation, and test set. We decided to split all the documents into sentences, since the lengths of the documents vary too much with respect to the lengths of the sentences.

Data preprocessing

Firstly, building vocabulary based on the given dataset words and tags. Secondly, defining a function for loading glove embedding, one is to check differences between pre-trained embedding model vocabulary and dataset specific vocabulary in order to highlight out-of-vocabulary terms. The other is to build an embedding matrix of a specific dataset given a pre-trained word embedding model. Thirdly, adding embeddings to the vocabulary, so to obtain vocabulary V2,OOV2,V3,OOV3,V4. Finally, Sequence embedding with padding.

Models

Baseline model—SimpleRNN

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 2900, 300)         4923000
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 2900, 128)         54912
_________________________________________________________________
time_distributed (TimeDistri (None, 2900, 46)          5934
=================================================================
Total params: 4,983,846
Trainable params: 60,846
Non-trainable params: 4,923,000
```

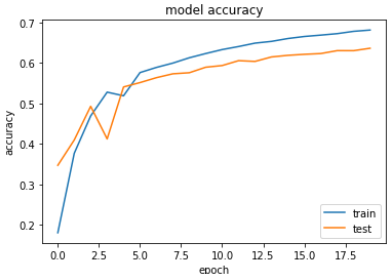

Figure 1. Baseline model structure (left) and training process (right)

We built a baseline model using keras with Embedding, SimpleRNN, and Dense layer. The Embedding layer is not trainable. Each iteration of the loop takes as input a vector representing the next word in the sequence, and the output activation of the last iteration in simpleRNN layer. The final classification result is produced through the dense layer.

New models

Next, we deformed the baseline model, using LSTM and GRU to prevent gradient disappearance with long short-term memory, and a bidirectional model with two directional information. All models were trained for 20 epochs. The optimizer here we used was Adam and the learning rate was 0.01.

- Model1--Bidirectional LSTM
- Model2--Bidirectional LSTM + Dense
- Model3--2 Bidirectional LSTM + Dense
- Model4--Bidirectional LSTM + GRU

The results of the accuracy of the model are as follows:

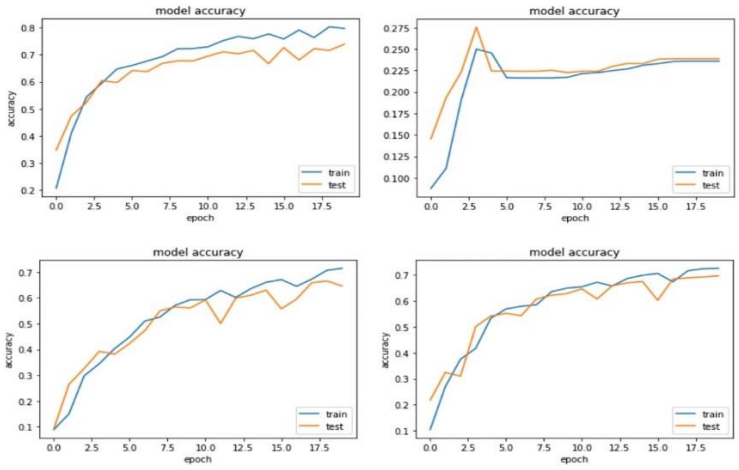

Figure 2. Accuracy during training of Model1 (top-left), Model2 (top-right), Model3 (bottom-left), Model4 (bottom-right).

Model1		**Model2**	
Loss	0.12383577972650528	Loss	0.4322543144226074
Accuracy	0.7416214942932129	Accuracy	0.2343084067106247
Model3		**Model4**	
Loss	0.17342257499694824	Loss	0.14391665160655975
Accuracy	0.643500566482544	Accuracy	0.6974040865898132

Error Analysis

Model 1 with Bidirectional LSTM, which got an accuracy of 0.74 on the validation set, exhibiting the greatest performance in our studies via the comparison of several models. Additionally, we checked the relevant confusion matrix and observed that some categories had classification results above 95%, while several had zero precision and recall. The final mean f1 in test set was 0.65.

Conclusion

Overall, based on the classification report, we came to the conclusion that Model 1 with Bidirectional LSTM is the best model for the task, with a final mean f1 0.65 in the test set, although performance might still be improved. Future experimentation options are multiple. First of all, the existing models still have space for a better performance, as can be observed from the training curve, and Model 1, 3, and 4 have not yet reached the overfitting point (declining of accuracy in validation set). Second, we only tried the 300 dimensions with GloVe embedding, which took a long time to train and only performed general (not reaching overfitting point), Reducing the dimension could improve performance as an option if considering efficiency. Finally, the confusion matrix distribution reveals that the unbalanced categories affect the performance, especially in some categories f1 are 0. This also demonstrates that in the future, balanced sampling tricks or augmentation approaches may be applied to boost performance.