HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

MSC BIG DATA TECHNOLOGY

6000H - Natural Language Processing

Covid19 NER

Wong Chin Hang 20123808
Le Berrigaud Lohan 20753689
Fan Vincent Ka Chun 20754774
TONG Ka Kiu Cathy 20671611

1. Your team name, names & student ids of team members, your rank and scores on the leaderboard, and the display name of your kaggle account.

Team name: team	Names & student ids of team members: Wong Chin Hang 20123808 Le Berrigaud Lohan 20753689 Fan Vincent Ka Chun 20754774 TONG Ka Kiu Cathy 20671611
Rank 19 Score: 0.91619	Display name of your kaggle account: Cathy Tong, fancent, CHIN HANG WONG, LLB0611

2. What algorithms are you using in this project?

We have tested several models, including a base model using word2vec, BERT, BiLSTM+CRF and RoBERTa.

We have tried to use word2vec as lower dimension word embedding which makes use of a neural network to learn word associations from a large corpus of text. In the project, we use pre-trained word2vec that learns word associations from Google News copus with about 100 billion words. The base model uses the pre-trained word2vec as an embedding layer, with bidirectional LSTM layer and TimeDistributed layer. Bidirectional LSTM runs both forward and backward to preserve information from the past and future. The Time Distributed Layer allows applying one layer to every element of a sequence independently. However, we have encountered technical issues as the google Colab do not have enough memory. Our team tried to run it on our local computers without gpu acceleration but it still kept training after several hours. We finally decided not to use word2vec with the base model. The following models have satisfied results without using word2vec.

In the BiLSTM+CRF model, we have first converted input pickle files into dataframes. The data frame consists of 3 columns: sentence index column, word column and tag column. Each word in sentences and their corresponding tags are broken down and stored in word column and tag column respectively.

After loading data, we have used word2idx to convert words to corresponding integer ID and tag2idx to convert tag to integer ID. In order to feed into BI-LSTM layer later in the model, all input texts have to be of the same length. We have selected the padding size as the length of the longest sentence.

In the BiLSTM+CRF model, there are 4 layers. The first layer is an embedding layer which converts each word into a fixed length vector of defined size. The second layer is a Bidirectional LSTM layer (BI-LSTM). BI-LSTM layer produces vector representation for input words. It produces a vector representation of each word in a sentence in both forward and backward directions. Forward direction and backward direction in BI-LSTM layer aggregate both past and future information. The third layer is a time distributed dense layer which

keeps one-to-one relations for input and output. In the model, one word will be classified as one tag only. The final layer is a Conditional Random Field (CRF) layer. As there are dependencies between predictions in NER, the CRF layer includes context from previous labels. We have more comprehensive information for making better predictions.

Since BERT has made huge changes in the NLP scene since 2018, we decided to use a pretrained BERT to perform the NER task. However, we learned from the last project that using RoBERTa, which is an improved version of BERT, would very likely yield better results. So, we first used the BERT model to train a baseline model, then we used RoBERTa to swap places with the BERT model to compare the results. From our local metrics, we concluded that RoBERTa is indeed a better model. We hypothesized that it is because RoBERTa makes use of dynamic masking, full-sentence without NSP loss and large mini-batches. In dynamic masking, training data will be duplicated into 10 copies. 15% of words in the sequence are randomly chosen to mask in each copy. These words are replaced by mask tokens. As a result, we have 10 different mask results in the same sentence. The inputs for RoBERTa are packed with full sentences sampled contiguously from one or more documents. RoBERTa also makes use of large mini-batches that includes more training data and increases training time. The resulting prediction therefore performs better than the BERT model.

3. How do you conduct parameter tuning? Is there any difference between your local validation and online results?

We used an empirical approach for the choice of hyperparameters. We tried to fine tune hyperparameters by comparing validation accuracy among them. We did a manual grid search for each of the hyperparameters. The hyperparameters include learning rate, number of layers, number of nodes of each layer, epsilon of Adam, batch size, weight decay and dropout rate.

The local validation and online results are pretty similar which led us to think that we are not overfitting nor undefitting. Moreover, we saw that RoBERTa performed better than BERT both locally and online. Therefore, this concluded our hypothesis of RoBERTa outperforming BERT.

4. How to run your code? Which third-party libraries are you using?

We used Google Colab to run our code because it provides gpu acceleration and it is a convenient platform to share work among teammates. However, we used Kaggle notebook for our final submission as it allows unattended usage as our model takes up to 2 hours to finish training. To reproduce our results, simply run all cells in sequential order in the kaggle notebook (project2-final.ipynb).

We used several libraries for different purposes:
1. Numpy, pandas, Dataset and pickle are employed for general math processing.
2. Pytorch, keras, tensorflow(1.15.2 version) and sklearn are employed for building BERT, CRF and RoBERTa
3. Transformers are employed for tokenization and building BERT and RoBERTa models
4. Gensim provides word2vec pretrained package for lower dimension embedding.