A simple DistilBert for question and answer machine

Tran Le Phuong Lan

Electrical Engineering 3rd year student

tran.lephuonglan@edu.bme.hu

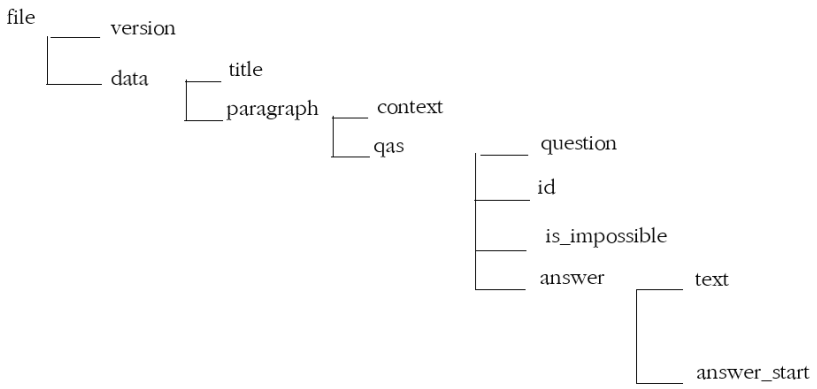
Abstract:

This project uses the free-provided dataset from https://rajpurkar.github.io/SQuAD-explorer . After importing the dataset into DataFrame object in Python, the next step is to define a pretrained-model DistilBertForQuestionAnswering and train the model on the downloaded dataset. DistilBert is used instead of BERT because the DistilBert is much smaller than BERT, however, it preserves 97% performance of normal BERT model.

1. Download and process the raw dataset:

By following the link in the abstract, we can see even though the dataset is provided with Training set v2.0, Dev set v2.0, Evaluation, i will only use the Training set v2.0.This is because there are more than 130000 samples in the Training set v2.0 which is big enough for fine tuning the pretrained model on this dataset in the small scale of my project.

The Training set v2.0 is downloaded as file with extension .json. After importing dataset into a Python object and examining it, the structure is described as:



File is dictionary type object with two items: version and data.

Data is a list of 442 objects. Each is dictionary type with two keywords: *title* and *paragraph*.

Paragraph is a list type object. The length of each *Paragraph* is different from each other. Objects of each *Paragraph* are dictionary type with two keywords: *context* and *qas*.

qas is a dictionary type object with four keywords: *question, id, is_impossible, answer*.

answer is dictionary type object with two keywords: *text* (which is the extracted answer from a '*context*' to a '*question*' of the '*context*), *answer_start* (*answer_start* indicate the index of the first character in the answer '*text*' in a '*context*').

Must put the downloaded dataset *.json* file in a folder 'data', the folder 'data' is created in the current working directory. Then, run the jupyter notebook file 'Data_Preparation_Project'.

After running the notebook file, there is created file 'final.tsv' saved in the folder 'data', which is the raw dataset has been transformed into a DataFrame object.

2. Process the samples in the DataFrame object into lists of appropriate inputs for the model: (refer to [1]) (this section is included in the notebook file 'BertModel_Project')

The preparation step: Before running the notebook file 'BertModel_Project', the 'final.tsv' must be uploaded into a folder 'data' which is created in the working directory of this notebook file.

The first step: Creating the labels for the model: by extracting the end and start tokens of an answer (in this case, the '*text*') in a '*context*'

Even though we already have the *answer_start*, however, it is not a valid label as for the model, the '*context*' and *'text*' must be firstly tokenized by the DistilBertTokenizerFast.

For example: *'context'* after tokenized {'hello', 'i' , 'am', 'lan', 'tran'}; *'text'* after tokenized {'lan', 'tran'}. The *answer_start label* must be the index of 'lan' in the 'context' list, which is 3. The *answer_end label* must be the index of 'tran' in the 'context' list, which is 4.

The *answer_start* is 3 as start label for the 'context': {'hello', 'i' , 'am', 'lan', 'tran'}. The *answer_end* is 4 as end label for the 'context': {'hello', 'i' , 'am', 'lan', 'tran'}.

This step is done for every sample in the DataFrame object.

The second step: Creating the appropriate input for the model. As the input of the DistilBertForQuestionAnswering contains the concatenation of '*context*' tokens and '*text*' tokens with special tokens [CLS], [SEP], the DistilBertTokenizerFast can take care of this task.

The third step: each input must have the same length of 512 tokens, no more, no less. The function: 'fit_max_length' makes sure length of each input to be 512-token maximum.

The fourth step: as a constrain of the model, every input must have the same length, in this case, the model requires the input length of 512. As a result, section 'PADDING' in the notebook file adds [PAD]s at the end of each input so that they are all 512 tokens.

The fifth step: creating segment mask for every input.

The sixth step: creating attention mask for every input.

The seventh step: split the dataset (the input, the start labels, the end labels, the attention masks, the segment masks) into 90% for training set and 10% for evaluation set

The eigth step: since the input for model must be torch.tensor type, we must convert both the training set and the evaluation set into torch.tensor type

3. Describe the model:

Instead of using the default model downloaded from the Hugging face, the model is wrapped into a class named 'ForQuestionAnsweringClassifier' so that we can configure some parameters like number layers, number of heads, dropout. With the configuration adjusted by us, the model can even become smaller for fitting training in free-provided GPU from GoogleColab. However, these parameters need choosing wisely, they will not affect the predicting accuracy of the model too much.

4. Training and Evaluation:

As mentioned in the section 3, there is limitation on the GPU memory for training and evaluating, we can not train the whole dataset (which is around 130000 samples) at a time. We should not only train the model on batches of the dataset but evaluate the model also on batches. From many failed experience with evaluation of the model on the evaluation set (which is always the runtime error, the GPU crashes due to not enough memory). This is the solution for evaluation I could think up in order to evaluate the model in the limited GPU.

5. Conclusion:

As the limit of time for the project, the built model still does not meet the high accurate predictions.

Nevertheless, the model could be improved more if the techniques learnt from the lab 7, lab 8 and lab 9 could be implemented in the model.

6. References:

[1] ChrisMcCormickAI, Fine tune Bert https://www.youtube.com/watch?v=x66kkDnbzi4

[2] ChrisMcCormickAI, applying SQUAD 1.0 dataset to BertForAnsweringQuestion already fine-tuned on SQUAD

[3] 'Question Answering with SQUAD 2.0' section from Hugging face/ transformer

[4] Jay Alammar, a blog about basic knowledge about fine tuning, input format, output format of BERT models