

Dependence environment

For all training notebook, they are designed to run on Colab with an instance that have A100 GPU, you should also run them on there. (Set the GPU to premium in Colab, then you may able to get an instance that have A100 GPU, depending on what Google is willing to assign to you according to their GPU availability.)

For others, our code is designed with a laptop RTX 3060 GPU and should run well on GPU that support CUDA, the code should also able to run on CPU, however, we haven't test that.

Below is the package and python we had used:

Python 3.7.9

Cuda 11.3, may need to change accord to GPU

datasets 2.10.1

transformers 4.26.0.dev0

pandas 1.3.5

numpy 1.21.6

matplotlib 3.5.3

seaborn 0.12.2

sklearn 0.0.post1

torch 1.12.1+cu113

tqdm 4.64.1

Overall

Our project has four parts. For what folder contains what please see below table:

answerable_classifier	contains code for train classifier by using train set and evaluate it using dev set.
train_bert_base_qa	contains code for train QA answering model using train set and evaluate it using dev-set.
combined	contains code for evaluate the overall model that use both two models above to give answer.
Read_txt_and_answer	The required application on predicting answer from a given span inside a document with a question (question need to input by user when running the program).

Run the project

Firstly, we talk about the conversion of both train and dev data set from official website. First download train and dev set from the official site provided from the project guide which are in .json form. Any of the implementations of our project requires those .json dataset be

converted to respective .csv file first before loading any of the programs. As our implementations is designed to load data from .csv.

To do this, you have to run the respective python programs with the respective dataset .json placed at the same root, to generate the respective .csv file, for details please below table:

Folder	To run programs	Use which program use to convert ,json to .csv	.json needed
answerable_classifier	train_classifier_colab.ipynb	convert_json_to_csv_for_classifier_train.py	train-v2.0.json
	dev_classifier_accuracy.ipynb	convert_json_to_csv_for_classifier_dev	dev-v2.0.json
train_bert_base_qa	train_bert_qa_colab.ipynb	convert_json_to_csv_train_set_no_impossible_q.py	train-v2.0.json
	predict.ipynb	convert_json_to_csv_dev_set.py	dev-v2.0.json
combined	predict_with_classifier.ipynb	convert_json_to_csv_dev_set_for_combined_use.py	dev-v2.0.json
Read_txt_and_answer	read_doc_and_answer.py	N/A	N/A

The .csv output names are mentioned at below section.

What each file does:

Folder	To run programs	Use
answerable_classifier	train_classifier_colab.ipynb	To train classifier at Colab
	dev_classifier_accuracy.ipynb	To evaluate classifier on dev set to get accuracy
train_bert_base_qa	train_bert_qa_colab.ipynb	To train QA model at Colab
	predict.ipynb	To generate predicted answers on QA model
combined	predict_with_classifier.ipynb	To generate predicted answers on combined model
Read_txt_and_answer	read_doc_and_answer.py	Our application that can read the context from .txt and answer user question inputs at the program.

Train BERT QA model

Go to folder train_bert_base_qa, ensure the train-v2.0-no-imposs-q.csv generated from above section is in here, then run first two section of code in train_bert_qa_colab.ipynb at colab, the first section of code will return what GPU you are using, ensure it is an A100, then second section will ask you for permission on google drive, enable it and then upload the generated

train-v2.0-no-imposs-q.csv to your google drive, find the path of it in Colab and change this code line from

```
“SQuAD = pd.read_csv("/content/gdrive/MyDrive/bertqa/train-v2.0-no-imposs-q.csv",  
dtype=str, keep_default_na = False)”
```

To

```
“SQuAD = pd.read_csv("<your path>/train-v2.0-no-imposs-q.csv", dtype=str,  
keep_default_na = False)”
```

And this code line from

```
“model.save_pretrained('/content/gdrive/MyDrive/bertqa/bert_qa_pt_'+ str(epoch))”
```

To

```
“model.save_pretrained('<your path>/bert_qa_pt_'+ str(epoch))”
```

Then save change, and run all to train.

For each epoch of training, a checkpoint will be saved at <your path>, with name “bert_qa_pt_n” where n is 0 to 3 for 4 epoch.

You can also check the progress of training and the average loss printed out after each epoch.

Evaluate QA model without classifier on dev set

Download all 4 generated checkpoints (the “bert_qa_pt_n” folders with its content) to the folder train_bert_base_qa (then this folder as root). And ensure dev-v2.0.csv converted from above section is in same root directory. Then, open predict.ipynb, modified the line:

```
“model = BertForQuestionAnswering.from_pretrained('./bert_qa_pt_3/',  
local_files_only=True)”
```

To

```
“model = BertForQuestionAnswering.from_pretrained('./<name of model you want to test>',  
local_files_only=True)”
```

Then run all the code to get output-dev.json, which is the evaluation result on dev set. Run the code with each model so you get all output-dev.json for all 4 epoch. Then use evaluate-v2.0.py from official squad website to get performance on squad 2.0 dev set based on each output-dev.json. Remember which epoch is the best, in our case, it is epoch 3, so bert_qa_pt_3.

Train classifier

Go to folder `answerable_classifier`, ensure the `train-classifier-v2.0.csv` generated from above section is in here, then run first two section of code in `train_classifier_colab.ipynb` at colab, the first section of code will return what GPU you are using, ensure it is an A100, then second section will ask you for permission on google drive, enable it and then upload the generated `train-classifier-v2.0.csv` to your google drive, find the path of it in Colab and change this code line from

```
“trainset = pd.read_csv("/content/gdrive/MyDrive/classifier_bert/train-classifier-v2.0.csv")
```

To

```
“trainset = pd.read_csv("<your path>/train-classifier-v2.0.csv")”
```

And this code line from

```
“model.save_pretrained('/content/gdrive/MyDrive/classifier_bert/bert_qa_classifier_pt_'+str(i))”
```

To

```
“model.save_pretrained('<your path>/bert_qa_classifier_pt_'+ str(i))”
```

Then save change, and run all to train.

For each epoch of training, a checkpoint will be saved at `<your path>`, with name

“`bert_qa_classifier_pt_n`” where `n` is 0 to 3 for 4 epoch.

You can also check the progress of training and the average loss printed out after each epoch.

Evaluate classifier model on dev set

Download all 4 generated checkpoints (the “`bert_qa_classifier_pt_n`” folders with its content) to the folder `answerable_classifier` (then this folder as root). And ensure `dev-classifier-v2.0.csv` converted from above section is in same root directory. Then, open `dev_classifier.ipynb`, modified the line:

```
“model = BertForSequenceClassification.from_pretrained(  
    'bert_qa_classifier_pt_3/', ...”
```

To

```
“model = BertForSequenceClassification.from_pretrained(  
    '<your model name>/', ...”
```

Then run all the code, at the last section of the code will print out the accuracy of your model, accuracy in there times 100 is the percentage of accuracy. Test all model and remember which is the best, in our case it is “`bert_qa_classifier_pt_3`”.

Evaluate our whole system

Fast, copy the best QA model, and best classifier model from above section to folder “combined” (this folder as root). In our case it is “bert_qa_classifier_pt_3” and “bert_qa_pt_3”. Then ensure the dev-v2.0-combined-use.csv is generated from above section and is at root. Then open predict_with_classifier.ipynb. Modified this line from

```
“model_qa = BertForQuestionAnswering.from_pretrained('./bert_qa_pt_3/',  
local_files_only=True)”
```

To

```
“model_qa = BertForQuestionAnswering.from_pretrained('./<your QA model>',  
local_files_only=True)”
```

And this line from

```
“classifier_model = BertForSequenceClassification.from_pretrained(  
    './bert_qa_classifier_pt_3/', ...”
```

To

```
“classifier_model = BertForSequenceClassification.from_pretrained(  
    './<your classifier model>', ...”
```

Save change and run all code. It will generate the prediction result of our system and save at output-dev.json, which can then use with the mentioned evaluate-v2.0.py from squad official site to get eval.json for the performance on squad 2.0 dev set.

Run the system with the simple application

Copy two best model mentioned, to the folder “read_txt_and_answer” (this folder as root). Modified the content inside “context.txt” to the context input you want. The context should be saved in the first line of the .txt. then open “read_doc_and_answer.py” and modified this line from:

```
“model_qa = BertForQuestionAnswering.from_pretrained('./bert_qa_pt_3/',  
local_files_only=True)”
```

To

```
“model_qa = BertForQuestionAnswering.from_pretrained('./<QA model>',  
local_files_only=True)”
```

And

This line from

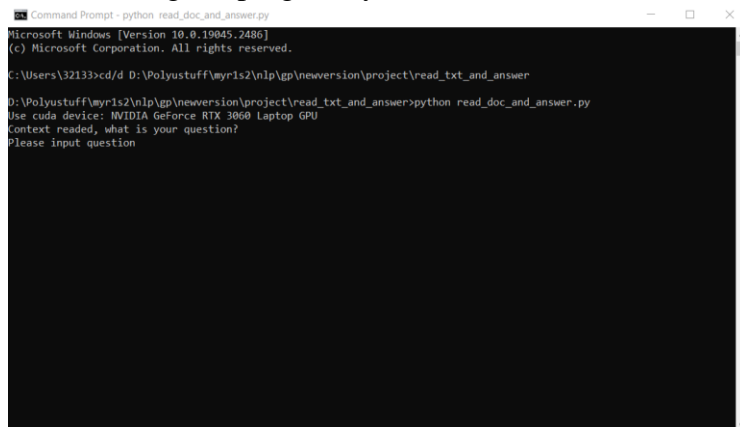
```
“classifier_model = BertForSequenceClassification.from_pretrained(  
    './bert_qa_classifier_pt_3/',”
```

To

```
“classifier_model = BertForSequenceClassification.from_pretrained(
    './<classifier model>/',”
```

Then save and run the program. If the model is not found/cannot be loaded, the program will print it cannot find model. In this case, you should ensure model is here , then close program and rerun it.

After running the program, you will see:



```
Command Prompt - python read_doc_and_answer.py
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\32133>cd/d D:\Polyustuff\myrls2\nlp\gp\newversion\project\read_txt_and_answer

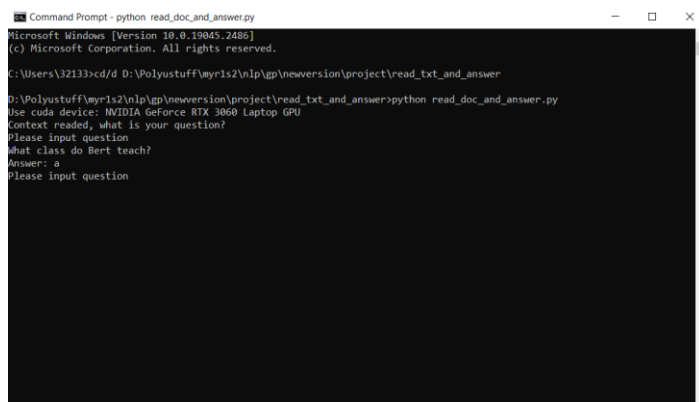
D:\Polyustuff\myrls2\nlp\gp\newversion\project\read_txt_and_answer>python read_doc_and_answer.py
Use cuda device: NVIDIA GeForce RTX 3060 Laptop GPU
Context readed, what is your question?
Please input question
```

Said the context is read and you can input question. In this example, the context is:

“Bert is a teacher, he teaches class a from Monday to Friday every week. Bert is an experienced teacher with 10 years of teaching experience.”

If I ask “What class do Bert teach?”

It will reply “a” which is correct:



```
Command Prompt - python read_doc_and_answer.py
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\32133>cd/d D:\Polyustuff\myrls2\nlp\gp\newversion\project\read_txt_and_answer

D:\Polyustuff\myrls2\nlp\gp\newversion\project\read_txt_and_answer>python read_doc_and_answer.py
Use cuda device: NVIDIA GeForce RTX 3060 Laptop GPU
Context readed, what is your question?
Please input question
What class do Bert teach?
Answer: a
Please input question
```

And you can also ask question indefinitely. When it meets a question that is not answerable, it will print “Answer:” Which is nothing. As our system out put “” nothing when meet a unanswerable question. Below is an example:

If I ask “When did Bert born?”, it replies “Answer:”

```
Command Prompt - python read_doc_and_answer.py
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\32133>cd/d D:\Polystuff\myr1s2\nlp\gp\newversion\project\read_txt_and_answer

D:\Polystuff\myr1s2\nlp\gp\newversion\project\read_txt_and_answer>python read_doc_and_answer.py
Use cuda device: NVIDIA GeForce RTX 3060 Laptop GPU
Context readed, what is your question?
Please input question
What class do Bert teach?
Answer: a
Please input question
When did Bert born?
Answer:
Please input question
```

To close the program, just close the command prompt (terminal in macOS).