

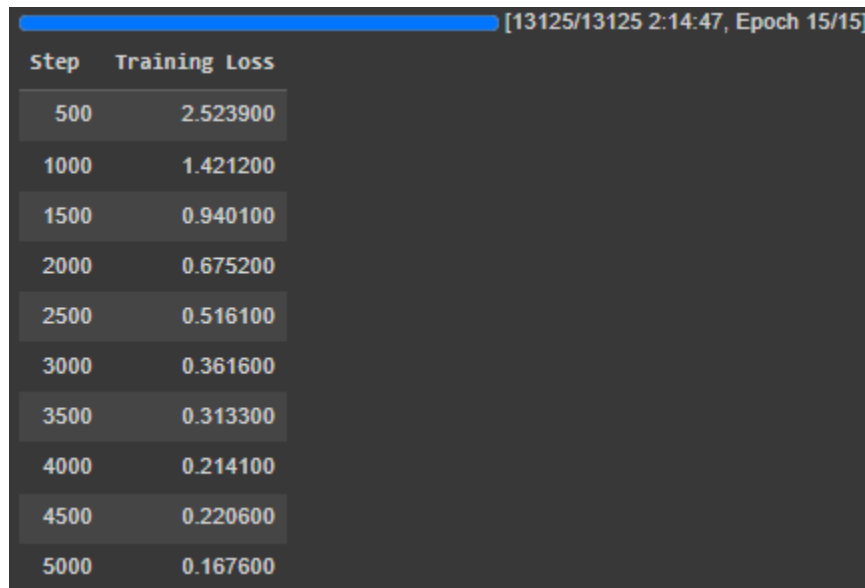
BertQA

Athreyan Mohana Krishnan Sangeetha

February 22, 2025

1 Training

The training for the model was done for 15 epochs, the training loss for the first five epochs are reported below.



A terminal window showing a blue progress bar at the top with the text "[13125/13125 2:14:47, Epoch 15/15]". Below the bar is a table with two columns: "Step" and "Training Loss". The table lists data for steps 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, and 5000, showing a decreasing trend in training loss.

Step	Training Loss
500	2.523900
1000	1.421200
1500	0.940100
2000	0.675200
2500	0.516100
3000	0.361600
3500	0.313300
4000	0.214100
4500	0.220600
5000	0.167600

Each step is a batch of 8. For a testing dataset size of 7000, there are totally 875 steps. So 4357 steps covers the first 5 epochs. The report includes more than the required number of steps.

2 Evaluation methods

For determining the exact match the target answer was converted to lowercase since the model we use produces uncased answers. We calculate f1 score for each prediction by defining precision as the number of common tokens divided by number of tokens in the predicted answer and the recall as the number of common tokens divided by the number of tokens in the target answer. It is worth noting that we do not consider the order of the tokens when calculating the common tokens.

```
[ ]: def compute_exact_match ( prediction , truth ):  
      return int( prediction.lower() == truth.lower() )  
  
def f1_score ( prediction , truth ):
```

```

pred_tokens = prediction . split ()
truth_tokens = truth . split ()

# if either the prediction or the truth is no - answer then f1 = 1 if they
→agree , 0 otherwise
if len( pred_tokens ) == 0 or len( truth_tokens ) == 0:
    return int ( pred_tokens == truth_tokens )

common_tokens = set( pred_tokens ) & set( truth_tokens )

# if there are no common tokens then f1 = 0
if len( common_tokens ) == 0:
    return 0

prec = len( common_tokens ) / len( pred_tokens )
rec = len( common_tokens ) / len( truth_tokens )

return 2 * ( prec * rec ) / ( prec + rec )

```

3 Results

3.1 Our Tuned Model

3.1.1 Examples of Mistakes

Wrong Answers Certain answers were plain wrong

```

Question: What year was the state tree selected?
Answer: 1949
Correct Answer: ['1908']
#####

```

```

Question: To whom did Chopin reveal in letters which parts of his work were about the singing student he was infatuated with?
Answer: konstancja gładkowska
Correct Answer: ['Tytus Woyciechowski']
#####

```

```

Question: Who taught directly or indirectly to all other Buddhas?
Answer:
Correct Answer: ['Gautama Buddha']
#####

```

Extra tokens Certain answers had extra tokens, the model couldn't exactly isolate which part of the output text contained the answer. Sometimes it was as small as a difference between plural and singular

```

Question: People communicate with dogs by voice commands, body language or posture and what else?
Answer: vocalization , hand signals and body posture .
Correct Answer: ['hand signals']
#####

```

```

Question: What group took tents and quilts to Wenchuan county?
Answer: the red cross society of china flew 557 tents and 2 , 500 quilts valued at 788
Correct Answer: ['Red Cross Society of China']
#####

```

```

Question: What was eliminated from the Hollywood round in Season seven?
Answer: group
Correct Answer: ['groups']
#####

```

Improper formatting Space formatting was one a big issue, commonly commas and hyphens do not have spaces on both sides of them. The model introduces spaces between tokens if one of the tokens doesn't have '#'s denoting a continuation of the previous token.

```

Question: About how many people speak Patois French in St. Barts?
Answer: 500 - 700
Correct Answer: ['500-700']
#####
Question: How many people are estimated to have died as a result of the creation of the Congo-Ocean Railroad?
Answer: 14 , 000
Correct Answer: ['14,000']
#####

```

```

Question: What was the commutation fee to avoid being conscripted during the American Civil War?
Answer: a $ 300
Correct Answer: ['$300']
#####

```

```

Question: What three composers did Chopin take inspiration from?
Answer: j . s . bach , mozart and schubert
Correct Answer: J. S. Bach, Mozart and Schubert
No Exact Match!
#####

```

Different script and languages The model could not answer non English script or when a different language was written in English script

```

Question: Who designed Chopin's tombstone?
Answer: clesinger
Correct Answer: Clésinger.
No Exact Match!
#####

```

```

Question: Which schools of Zen likes the use of meditation on the koan for spiritual breakthroughs?
Answer: rinzai ( [UNK] [UNK] 宗 ) and soto ( [UNK] [UNK] 宗 )
Correct Answer: Rinzai

```

3.1.2 Final Metrics

The model did not initially perform very well due to the absence of detokenization and case issues. After using a function(`tokenizer.convert_tokens_to_string()`) that detokenized a list of tokens into an output string and allowing the matching to be case insensitive the model started performing better.

```
Total Count: 1000
Exact Match Percentage: 45.4 %
Average F1 Score: 0.6067432780400943
Median F1 Score: 0.7826086956521738
Median EM: 0
```

3.2 Pretuned Model

The pretuned model still performed way better than this model likely due to the model being cased which may allow it to better understand context, availability of a bigger dataset for training and possibly minor architectural changes on the output head.

3.2.1 Examples of Mistakes

Wrong Answers Sometimes the model gives wrong answer, both the prediction and the truth don't make sense in the second image.

```
Question: How many albums does Kanye have on the "500 Greatest Albums of All Time" list?
Answer: 32 million
Correct Answer: 3
```

```
Question: Which album was darker in tone from her previous work?
Answer: 4
Correct Answer: Beyoncé
```

Extra Tokens The same issues of extra tokens ruin the exact match and f1 score

```
Question: Who is still looking for compensation and justice?
Answer: the many families
Correct Answer: many families
```

```
Question: Where did Chopin prefer to play for people?
Answer: Paris apartment
Correct Answer: apartment
```

Missing Tokens Sometimes certain tokens are missing in such a way that the answer makes sense but it's not an exact match

```
Question: What was the result of that lecture?
Answer: he was promptly hired to the faculty and taught there for twenty years
Correct Answer: Wieman's lecture was so brilliant that he was promptly hired to the faculty and taught there for twenty years
```

```
Question: In which century were antibiotics first introduced?
Answer: 20th
Correct Answer: 20th century
```

Punctuation absorbed in the token The correct answer sometimes includes periods and quotes as part of the answer which doesn't seem correct.

```
Question: What was the name of the tour featuring both Beyoncé and Jay Z?  
Answer: On the Run Tour  
Correct Answer: On the Run Tour.  
No Exact Match!  
F1 Score: 0.75
```

```
Question: Who designed Chopin's tombstone?  
Answer: Clésinger  
Correct Answer: Clésinger.
```

```
Question: In turn, Chinese supporters have accused Western media of being what in their coverage?  
Answer: biased  
Correct Answer: biased.
```

```
Question: What was the top single off the album "Watch the Throne"?  
Answer: Niggas in Paris  
Correct Answer: "Niggas in Paris"
```

3.2.2 Improvements compared to our tuned model

This model not only improves on the case sensitivity but the formatting with regards to numbers is also improved.

```
Question: What length is the course of study at the Notre Dame School of Architecture?  
Answer: five-year  
Correct Answer: five-year  
Exact Match!  
F1 Score: 1.0  
#####  
Question: How many school buildings collapsed in the province?  
Answer: 7,000  
Correct Answer: 7,000  
Exact Match!  
F1 Score: 1.0
```

3.2.3 Final Metrics

```
Total Count: 1000  
Exact Match Percentage: 77.10000000000001 %  
Average F1 Score: 0.8927121639197584  
Median F1 Score: 1.0  
Median EM: 1
```

4 Code

```
[ ]: # !pip install datasets  
     # !pip install accelerate -U  
  
import pickle  
from transformers import BertForQuestionAnswering
```

```

from transformers import BertTokenizer
from transformers import TrainingArguments
from transformers import Trainer
from datasets import Dataset
import pandas as pd
import numpy as np
from transformers import pipeline

from google.colab import drive
import google.colab.auth
google.colab.auth.authenticate_user()
drive.mount('/content/drive')

drive_path = '/content/drive/My Drive/ece60146/HW10/'
with open (drive_path + 'dataset/train_dict.pkl', 'rb') as f:
    train_dict = pickle . load (f)
with open (drive_path + 'dataset/test_dict.pkl', 'rb') as f:
    test_dict = pickle.load (f)

with open (drive_path + 'dataset/eval_dict.pkl', 'rb') as f:
    eval_dict = pickle.load (f)

with open (drive_path + 'dataset/train_data_processed.pkl', 'rb') as f:
    train_processed = pickle . load (f)

with open (drive_path + 'dataset/test_data_processed.pkl', 'rb') as f:
    test_processed = pickle . load (f)

with open (drive_path + 'dataset/eval_data_processed.pkl','rb') as f:
    eval_processed = pickle . load (f)

print( train_dict . keys () )
print( test_dict . keys () )
print( eval_dict . keys () )

print( train_processed . keys () )
print( test_processed . keys () )
print( eval_processed . keys () )

model_name = 'bert-base-uncased'
model = BertForQuestionAnswering. from_pretrained (model_name )

# model = BertForQuestionAnswering. from_pretrained (drive_path + 'model/
↳ bert_tuned_15e.pt')

print( model . _modules )

```

```

training_args = TrainingArguments (
output_dir =drive_path + 'results', # output directory
use_mps_device = False ,
num_train_epochs =15 , # total number of training epochs , change this as you
    ↳need
per_device_train_batch_size=8 , # batch size per device during training ,change
    ↳this as you need
per_device_eval_batch_size=8 , # batch size for evaluation , change this as you
    ↳need
weight_decay =0.01 , # strength of weight decay
logging_dir = drive_path + 'logs', # directory for storing logs
)

train_dataset = Dataset . from_pandas ( pd . DataFrame (train_processed ))
eval_dataset = Dataset . from_pandas ( pd . DataFrame (eval_processed ))
test_dataset = Dataset . from_pandas ( pd . DataFrame (test_processed ))

trainer = Trainer (
model=model , # theinstantiated Transformersmodel to be fine - tuned
args = training_args , # training arguments , defined above
train_dataset = train_dataset , # training dataset
eval_dataset = eval_dataset # evaluation dataset
)
trainer . train ()

trainer.save_model(drive_path + 'model/bert_tuned_15e.pt')

def compute_exact_match ( prediction , truth ):
    return int( prediction.lower() == truth.lower() )

def f1_score ( prediction , truth ):
    pred_tokens = prediction . split ()
    truth_tokens = truth . split ()

    # if either the prediction or the truth is no - answer then f1 = 1 if they
    ↳agree , 0 otherwise
    if len( pred_tokens ) == 0 or len( truth_tokens ) == 0:
        return int ( pred_tokens == truth_tokens )

    common_tokens = set( pred_tokens ) & set( truth_tokens )

    # if there are no common tokens then f1 = 0
    if len( common_tokens ) == 0:
        return 0

    prec = len( common_tokens ) / len( pred_tokens )

```

```

rec = len( common_tokens ) / len( truth_tokens )

return 2 * ( prec * rec ) / ( prec + rec )

print(compute_exact_match('hello','Hello'))

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

x = trainer . predict ( test_dataset )
start_pos , end_pos = x. predictions
start_pos = np . argmax (start_pos , axis =1)
end_pos = np . argmax ( end_pos , axis =1)

trained_exact_match_count = 0
total_count = 0
trained_f1_score_sum = 0
trained_f1_score_list = []
for k , (i , j) in enumerate ( zip(start_pos , end_pos)):
    tokens = tokenizer . convert_ids_to_tokens (test_processed ['input_ids'] [k])
    answer = tokenizer.convert_tokens_to_string(tokens[i:j+1])
    correct_answer = test_dict ['answers'] [k] ['text'] [0]

    print('Question: ', test_dict ['question'] [k])
    print('Answer:', answer)
    print('Correct Answer: ', correct_answer)

    if compute_exact_match( answer, correct_answer):
        trained_exact_match_count += 1
        print("Exact Match!")
    else:
        print("No Exact Match!")
    total_count += 1
    trained_f1_score_sum += f1_score( answer.lower(), correct_answer.lower())
    trained_f1_score_list.append(f1_score( answer.lower(), correct_answer.
→lower()))
    print('F1 Score: ', f1_score( answer.lower(), correct_answer.lower()))
    print('#####')

    # break

print('Total Count: ', total_count)
print('Exact Match Percentage: ', trained_exact_match_count /
→total_count*100,"%")
print('Average F1 Score: ', trained_f1_score_sum / total_count)
print('Median F1 Score: ' , np.median(trained_f1_score_list))
if trained_exact_match_count > 500:
    print('Median EM:', 1)

```



```

elif trained_exact_match_count == 500:
    print('Median EM:', 0.5)
else:
    print('Median EM:', 0)

question_answerer = pipeline ("question-answering",
    ↪model='distilbert-base-cased-distilled-squad')
pretrained_exact_match_count = 0
total_count = 0
pretrained_f1_score_sum = 0
pretrained_f1_score_list = []
for i in range(len ( test_dict ['question'])) ):
    result = question_answerer( question = test_dict ['question'][i], context =
    ↪test_dict ['context'][i])
    answer = result['answer']
    correct_answer = test_dict ['answers'][i]['text'][0]

    print('Question: ', test_dict ['question'][i])
    print('Answer: ', answer)
    print('Correct Answer: ', correct_answer)

    if compute_exact_match( answer, correct_answer):
        pretrained_exact_match_count += 1
        print("Exact Match!")
    else:
        print("No Exact Match!")
    total_count += 1
    f1_score_val = f1_score( answer.lower(), correct_answer.lower())
    pretrained_f1_score_sum += f1_score_val
    print('F1 Score: ', f1_score_val)
    pretrained_f1_score_list.append(f1_score_val)
    print('#####')

print('Total Count: ', total_count)
print('Exact Match Percentage: ', pretrained_exact_match_count /
    ↪total_count*100,"%")
print('Average F1 Score: ', pretrained_f1_score_sum / total_count)
print('Median F1 Score:' , np.median(pretrained_f1_score_list))
if pretrained_exact_match_count > 500:
    print('Median EM:', 1)
elif pretrained_exact_match_count == 500:
    print('Median EM:', 0.5)
else:
    print('Median EM:', 0)

```