CPSC 8430-Deep learning
Homework-3

Harshini Gaddam
C11074076

Extractive question answering using BERT

Github link:
https://github.com/Harshini-Gaddam/DL_HW3_Harshini/blob/main/Bert_DL3_Harshini.ipynb

Task: Extractive question answering

In a given paragraph, the model must identify the answer to the question we posed. BERT transformers are used since BERT is a type of transformer that has been pre-trained on a large amount of text data and can be fine-tuned for different NLP tasks such as question-answering.

BERT (Bidirectional Encoder Representations from Transformers):

A particular type of neural network architecture called a transformer is made to handle data in sequences like sentences or paragraphs. It makes use of self-attention methods to enable the network to pay attention to various input sequence segments at various moments, allowing it to capture long-range dependencies between words.

Dataset:

Spokensquad dataset is used to train and test. The dataset has 37,111 question answer pairs as the training set and 5,351 question answer pairs as testing set.

Parameters:

Palmetto is used to launch jupyter notebook with the computational environment “cuda” with A100.
MAX_LENGTH = 250
MODEL_PATH = "bert-base-uncased"
Initial Learning rate=2e^-5
Learning rate: 1.62e^-05 (used in the recent optimizer step)
Epochs=10
Optimizer=Adam

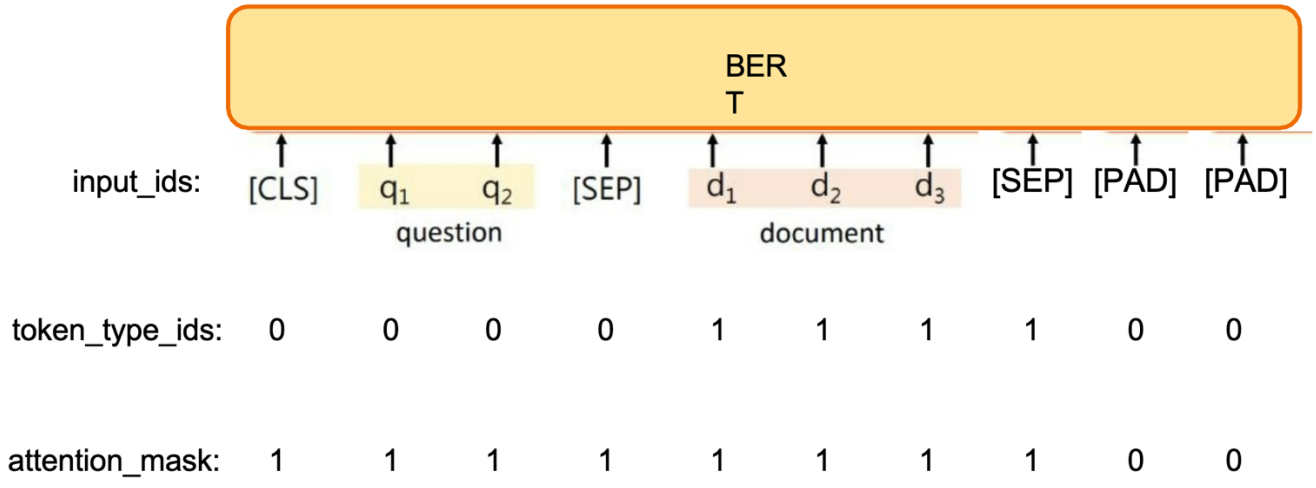
Data Processing:

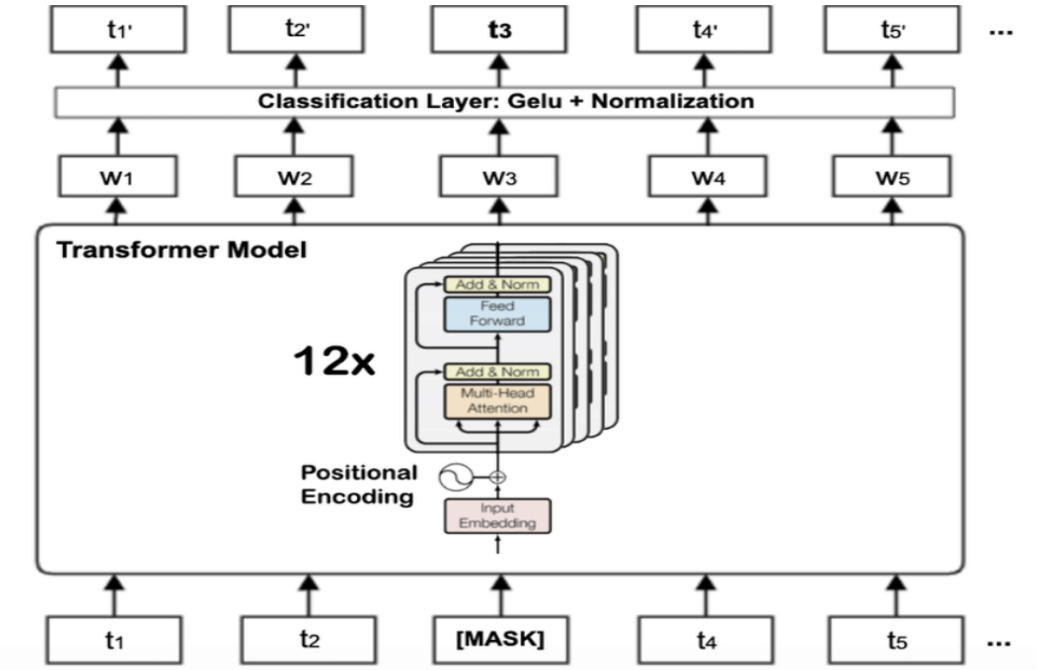
From the training dataset which has 37,111 question answer pair: questions, answers and contexts are imported to train_contexts, train_questions, train_answers. From the testing set, these are imported to validate into valid_contexts, valid_questions, valid_answers.

Data is tokenized, where Tokenization is the process of disassembling a string of text into smaller tokens, such words since Spokensquad dataset has long paragraphs. It enables the neural network to recognize the associations between words in the input text and comprehend its grammatical structure. During tokenization, the text is cleaned and preprocessed, next the text is split into individual words or subwords. After the text has been split into tokens, each token is assigned a unique integer ID that corresponds to its position in the tokenizer's vocabulary.
After tokenization, where Spokensquad text has been converted into integer_ids, where the position of each token in the tokenizer's vocabulary is represented by an individual integer ID that is given to each token in the text. The input_ids parameter is used to represent the input sequence to the BERT model. The BERT model takes this sequence of token IDs as input and generates a sequence of hidden representations for each token in the input sequence.

token_type_ids is an input parameter representing the segment IDs of tokens. token_type_ids are used to indicate which tokens belong to the first sentence and which tokens belong to the second sentence. BERT model with token_type_ids can differentiate between different segments of the input sequence and learn to encode the relationships between them.

The self-attention mechanism in the transformer layers of the model is controlled by an input parameter Attention_mask , that specifies which tokens in the input sequence it should focus on. The (batch size, sequence length) binary tensor that makes up the attention mask has elements that are either 0s or 1s. The corresponding token should be attended to if the value is 1, whereas the corresponding token should be disregarded if the value is 0. Each token attends to every other token in the sequence during the self-attention phase in the transformer layers of the BERT model, however the attention mask controls the attention to only the sequence's real tokens and not its padding tokens.





Input_ids, token_type_ids, attention_mask, start_positions, end_positions of a paragraph that is tokenized can be seen below.

```
{'input_ids': tensor([[ 101, 2004, 4941,  ...,    0,    0,    0],
        [ 101, 2054, 2001,  ...,    0,    0,    0],
        [ 101, 2129, 2116,  ...,    0,    0,    0],
        ...,
        [ 101, 2129, 2116,  ...,    0,    0,    0],
        [ 101, 2129, 2020,  ...,    0,    0,    0],
        [ 101, 2079, 3767,  ...,    0,    0,    0]]), 'token_type_ids': tensor([[0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        ...,
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0]]), 'attention_mask': tensor([[1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        ...,
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0]]), 'start_positions': tensor([ 83, 116, 118, 117, 135, 125,  69,  85,   1,  77, 128,
103,  43,  65,
         53, 100]), 'end_positions': tensor([ 84, 117, 119, 118, 136, 126,  70,  86,   2,  78, 129, 104,  44,  66,
         54, 101])}
```

Maximum input sequence length of BERT is restricted to 512, why?
Bert layers operates on the input sequence in parallel and the amount of memory and computation required increases with the length of the input sequence. The 512 limit strikes a balance between the computational resources required to capture long-range dependencies and the model's ability to do so.

The sentences are padded according to the max_length set.

```
[CLS] which nfl team represented the afc at super bowl 50? [SEP] super bowl fifty was an american football game to det
ermine the champion of the national football league nfl for the twenty fifteen season. the american football conferenc
e a f c c champion denver broncos defeated the national football conference n f c c champion carolina panthers twenty
four to ten to earn their third super bowl title. the game was played on february seventh twenty sixteen and levis sta
dium in the san francisco bay area santa clara california. as this was the fiftieth super bowl the league emphasized t
he golden anniversary with various goldsteins initiatives as well as temporarily suspending the tradition of naming ea
ch super bowl game with roman numerals under which they gain would have been known as super bowl l sell that the logo
could prominently featured the arabic numerals fifty. [SEP] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PA
D] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PA
D] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PA
D]
```

MAX_LENGTH = 250
MODEL_PATH = "bert-base-uncased"

Hugging Face BERT-based uncased model uses lower case text and leaves out case details in its instruction. As a result, it is suited for tasks where case information is not crucial because it is not sensitive to capitalization in text inputs.

Learning rate adjustment:

Learning rate scheduler is used to adjust the learning rate of the optimizer during the training process to improve the performance of the model.

scheduler.get_last_lr() is used to obtain the learning rate(s) used in the most recent optimizer step(s) of a training process that employs a learning rate scheduler and to analyze the impact of the learning rate scheduler on the performance of the model.

scheduler = ExponentialLR(optim, gamma=0.9) is used to create an Exponential learning rate scheduler. Gamma=0.9 means that the learning rate will be multiplied by 0.9 (i.e reduced by 10%) at the end of each epoch. scheduler.step() at the end of each epoch causes the learning rate to be adjusted according to the exponential decay schedule specified by the gamma parameter.

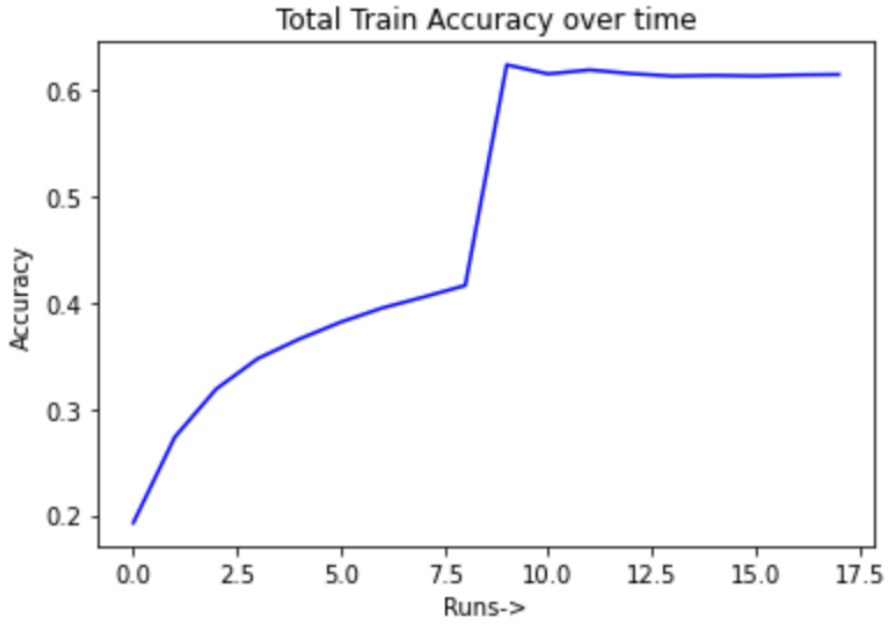
Accuracy of the model:

The accuracy of a BERT model refers to how well it performs on extractive question answering task.
Accuracy = (Number of correctly classified samples) / N
The number of samples that the model correctly classified divided by the total number of samples in the validation set.
Accuracy of the model is calculated using the formula,

Total_accuracy = (sum(acc)/len(acc))

Where sum(acc) is the total sum of samples model correctly predicted and len(acc) is the total length of validation set.

Accuracy is plotted over time

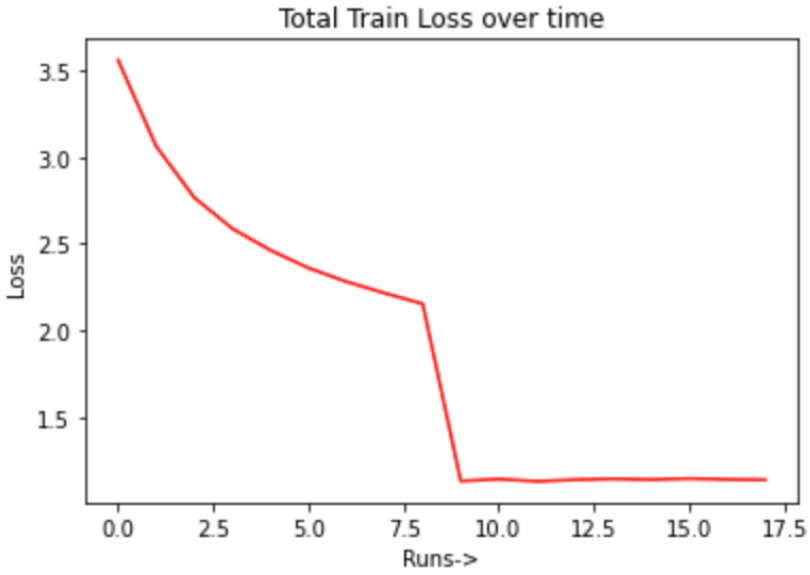


From the above plot, it can be inferred that total_accuracy is increasing over time as the number of epochs are being increased. This says that the performance of the model is good and the model is improving its ability to accurately predict the outcomes of the task.

Training loss:

loss = focal_loss_fn(out_start, out_end, start_positions, end_positions,1) is used to compute the training loss for the model. The model's accuracy in identifying answer spans is improved by the focal loss function's attempt to assign greater weight to the minority class. out_start and out_end are the output logits of the BERT model for the start and end positions of the answer span. start_positions and end_positions are the true start and end positions of the answer span in the input text.

Training loss is plotted over time.



It is seen that the training_loss is getting decreased which indicates that the model is gradually improving its performance on the training data and is becoming more accurate in predicting the target variable.

WER (Word error rate):

WER is the metric used to assess the efficacy of language modeling systems in natural language processing.
WER is calculated by comparing the number of errors between the recognized text and the reference text, and dividing that by the total number of words in the reference text.

wer_score = wer.compute(predictions=pred_answers, references=true_answers)
The above line of code is used to calculate wer_score and it is around 5-10% usually.

Conclusion:

Using Bert, spokensquad dataset is trained and validated.