CPSC 8430 : Deep Learning
Homework 3 Report
Submitted by: Shwetha Sivakumar
CUID: C20140199

GitHub link for HW3:
https://github.com/shwethasivakumar/Deep-Learning/blob/main/HW3/HW3_extractive_question_answering.ipynb

We will be working on the "Extractive question answering" task as part of Homework 3. An extractive question-answer task entails determining and obtaining the response to the specified query from the dataset or document that was supplied. Our tool of choice will be BERT, namely the bert-base-uncased model in the extractive question-answering task using the Spoken-SQuAD dataset.

Dataset Description:

The dataset used in this assignment is SpokenSQuAD a document in spoken format; the questions are in text format, while each answer is a span of words in the document. To create the spoken dataset from the SQuAD dataset, the following steps were followed: First, the articles were converted into spoken form using the Google text-to-speech software. Then, the respective automatic speech recognition transcriptions were obtained using CMU Sphinx. Throughout this dataset or document, the questions were in text format. For the training set of SpokenSQuAD, the SQuAD training set is used, and for the testing set, the SQuAD development set is used. In all, the dataset consisted of 37,111 question-and-answer pairs as the training set and 5,351 question-and-answer pairs as the testing set.

BERT (bert-based-uncased):

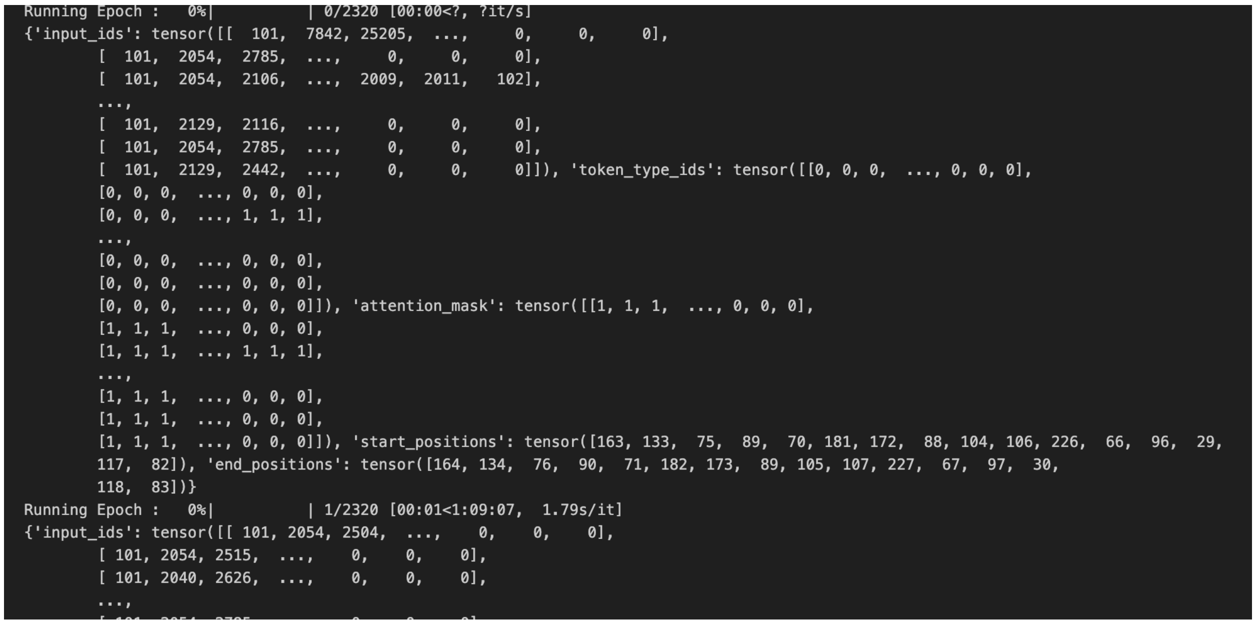
For our task, the BERT model will be used for training and testing. BERT is based on a transformer architecture used in different NLP tasks to capture the contextualized representations of bidirectional text. Pre-training of BERT refers to the fact that the BERT model is trained on a large dataset comprising unlabeled text data. The BERT model becomes capable of predicting the answers present within a sentence based on the terms surrounding the sentence in question. It uses an MLM, in which a pre-determined portion of the words in each input sentence are randomly replaced with a unique token denoted by [MASK]. The model is then trained to predict the replaced terms in light of the context of the remaining terms in the input sentence. Simultaneously in training, BERT incorporates the NSP. In this task, it is trained to predict whether a given pair of sentences is consecutive in the original text or randomly sampled from different parts of the corpus. This helps BERT understand the required context better. After pre-training on the unlabeled text, BERT can be fine-tuned for extractive question-answering tasks. In the case of the Spoken-SQuAD dataset, BERT is fine-tuned for question-answer tasks by feeding it passages of text, with questions related to the text, along with their answers. The pre-trained BERT model parameters were fine-tuned better to best suit the extractive question-answering task.

Parameters:

The following are the parameters of the computational environment that were used in tuning the BERT model: Palmetto was used to start a Jupyter Notebook with the settings of the computational environment set to "cuda" with V100 hardware. Some of the important parameters are MAX_LENGTH being set at 250 characters and the chosen model being "bert-base-uncased." Most importantly, the original learning rate was set to 2e^-5. During the process of optimization, the learning rate changes, having a value of 1.62e^-05 at the last optimizer step. Training runs over a total of 10 epochs, and the optimizer used is Adam.

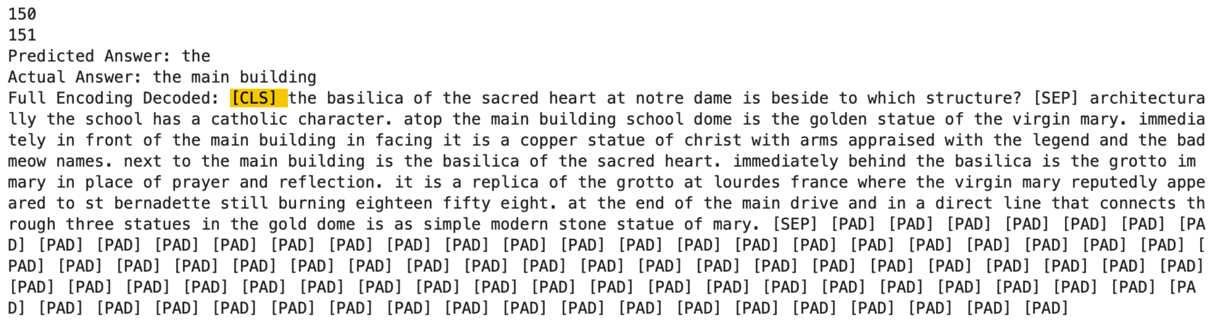
Preprocessing the data:

The major steps concerned with the processing of data in this project involve loading training and validation datasets. This is done to fetch context, questions, and answers. Upon loading the data, tokenization of SpokenSQuAD is performed to prepare the text to feed into the model. It means the text has to be broken down into smaller tokens, usually words or sub-words, through which the neural network views word relationships and grammatical structure. The token_type_ids in BERT is to understand where one sequence ends and where another begins. Once cleaned and preprocessed, the text is tokenized, breaking it down into individual pieces or tokens, each token is assigned an integer ID that corresponds to its position in the format of the tokenizer. These resulting tokens are contained in the `input_ids`, which becomes the sequence that will be fed into the BERT model. But the `token_type_ids` is used to represent segment IDs of tokens to identify which of the tokens come from the first or second sentence. Therefore, this segmentation allows the model to determine parts of the input sequence easily so that the model can encode the relationship between the two segments efficiently. The `attention_mask` is an input parameter controlling the self-attention mechanism of transformer layers inside the BERT model, indicating the tokens it should pay more attention to in the course of its process. This binary tensor ensures that the model attends to real tokens in the sequence while ignoring padding tokens. Each of these steps in the processing of data makes sure the input data is correctly tokenized, segmented, and masked for input into the BERT model during training and validation, hence learning the SpokenSQuAD spoken content effectively.



```
Running Epoch :   0%|          | 0/2320 [00:00<?, ?it/s]
{'input_ids': tensor([[  101,  7842, 25205,  ...,     0,     0,     0],
        [  101,  2054,  2785,  ...,     0,     0,     0],
        [  101,  2054,  2106,  ...,  2009,  2011,   102],
        ...,
        [  101,  2129,  2116,  ...,     0,     0,     0],
        [  101,  2054,  2785,  ...,     0,     0,     0],
        [  101,  2129,  2442,  ...,     0,     0,     0]]), 'token_type_ids': tensor([[0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 1, 1, 1],
        ...,
        [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,  ..., 1, 1, 1],
        ...,
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0]]), 'start_positions': tensor([163, 133,  75,  89,  70, 181, 172,  88, 104, 106, 226,  66,  96,  29,
        117,  82]), 'end_positions': tensor([164, 134,  76,  90,  71, 182, 173,  89, 105, 107, 227,  67,  97,  30,
        118,  83])}
Running Epoch :   0%|          | 1/2320 [00:01<1:09:07,  1.79s/it]
{'input_ids': tensor([[ 101, 2054, 2504,  ...,    0,    0,    0],
        [ 101, 2054, 2515,  ...,    0,    0,    0],
        [ 101, 2040, 2626,  ...,    0,    0,    0],
        ...,
```

Why does BERT restrict the maximum input sequence length to 512?
The layers in BERT process the input sequence in parallel, and computational resources and memory grow linearly with the sequence length. A token size limit of 512 strikes a balance between the amount of computational resources deployed to capture long-range dependencies and the model performance in leveraging such dependencies effectively. Padding for the sentences is also done by the max_length parameter defined.



```
150
151
Predicted Answer: the
Actual Answer: the main building
Full Encoding Decoded: [CLS] the basilica of the sacred heart at notre dame is beside to which structure? [SEP] architectura
lly the school has a catholic character. atop the main building school dome is the golden statue of the virgin mary. immedia
tely in front of the main building in facing it is a copper statue of christ with arms appraised with the legend and the bad
meow names. next to the main building is the basilica of the sacred heart. immediately behind the basilica is the grotto im
mary in place of prayer and reflection. it is a replica of the grotto at lourdes france where the virgin mary reputedly appe
ared to st bernadette still burning eighteen fifty eight. at the end of the main drive and in a direct line that connects th
rough three statues in the gold dome is as simple modern stone statue of mary. [SEP] [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] [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]
```

The Hugging Face BERT-based uncased model uses lowercase text and does not distinguish cases in line with its design. Hence, it is quite suitable for tasks whose requirements do not rely on case sensitivity since it does not consider capitalization in text input.

Training and Testing Process:

The process of training and testing in fine-tuning the BERT model on the Spoken-SQuAD dataset involves a few important aspects. First, this model is initialized using class `BertModel` of the library `Transformers`, which is intended for natural language comprehension and has been pre-trained on massive datasets to learn the contextualized representations of words. Once the model has been initialized, a loop is performed over batches coming from `DataLoader`. Inside the loop, the gradients of the model parameters are zeroed in order not to accumulate from previous iterations before performing a forward pass. In the forward pass, input data is given to the BERT to obtain the predictions of the start and end positions of the answer spans. Afterwards, the focal loss function in order to find the loss between predicted and true start/end positions. Later on, by means of backpropagation, gradients are computed and used by the AdamW optimizer in order to update model parameters. Moreover, a learning rate scheduler is used, which might improve convergence and slightly better results based on runtime changes in the learning rate for training epochs. Model evaluation then takes place on the validation dataset by comparing model output, that is, predictions of answer spans, with true answer spans to compute accuracy metrics. The training and evaluation process also covers monitoring and displaying several important metrics, such as training accuracy, training loss, and validation accuracy, to track the performance and progress of model training. This holistic training pipeline will allow the BERT model to learn useful representations of spoken text and accurately predict the answer span given a question.

Learning rate adjustment:

A learning rate scheduler dynamically adjusts the optimizer's learning rate during training to improve the performance of the model. When using a learning rate scheduler, the `scheduler.get_last_lr()` returns the learning rates of recent optimizer steps, one can inspect how this may affect model performance. Initialize an exponential learning rate scheduler via the variable `scheduler = ExponentialLR(optim, gamma=0.9)`, where a gamma value of 0.9 reduces the learning exponentially decaying the rate by 10% at the end of each epoch. Then calling `scheduler.step()` at the end of each epoch's end will decrease the learning rate depending on the exponential decay schedule defined by gamma.

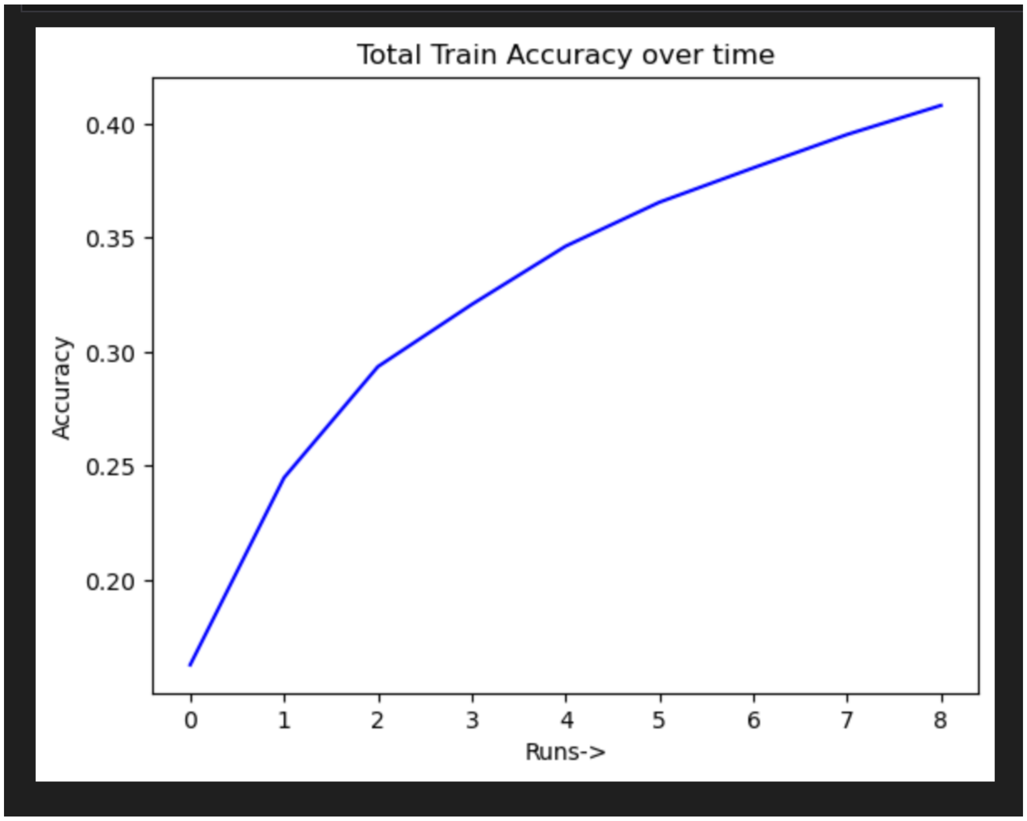
Doc Stride:

The common usage of doc_stride is for a step size when sliding the window over a document, especially with long documents, and those in Spoken-SQuAD. It determines how much each window is overlapped with the next one. If this was the case in the code, where doc_stride was set to max_paragraph then this would infer that the gap between successive windows is precisely the size of the longest paragraph in the dataset. This way, one can ensure that at least a single window must have overlapped above any part of the document during feature extraction.

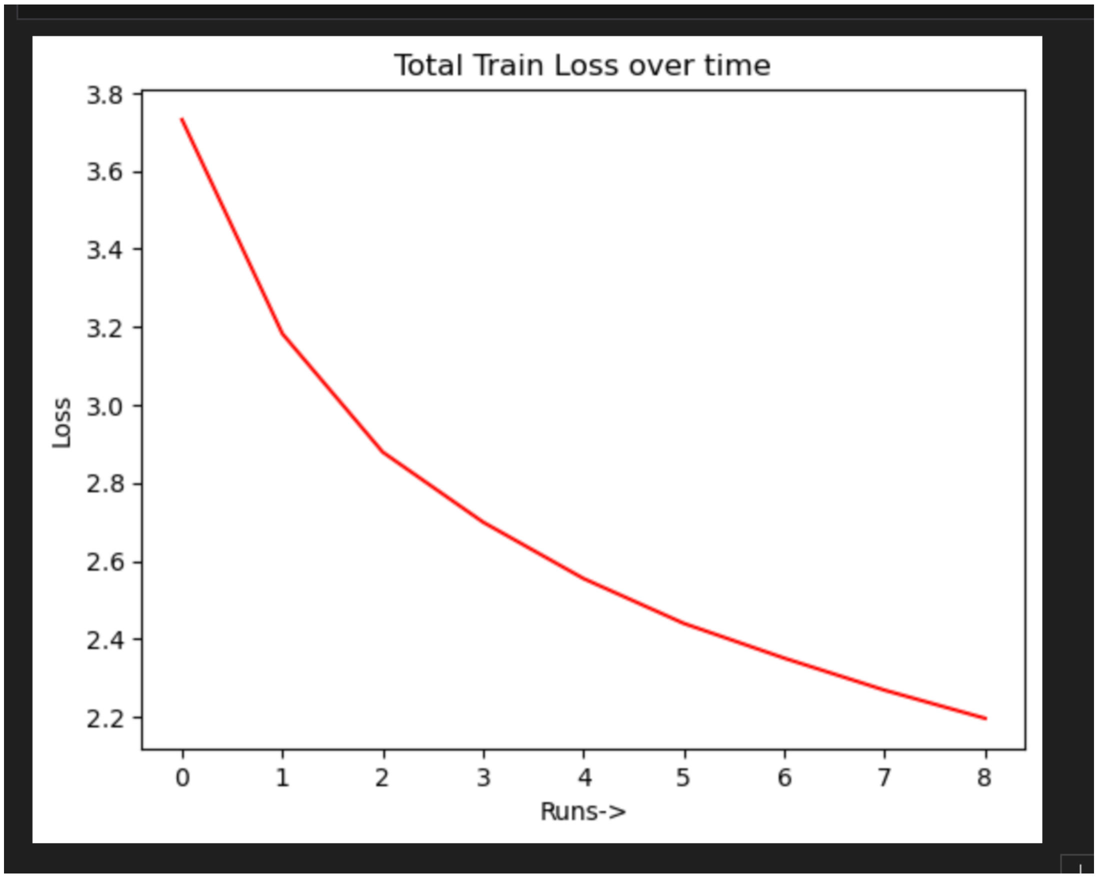
What if the answer is near the boundary of windows or across windows?
Suppose an answer crosses two different windows, or falls near the edges of any of the windows. In this regard, the boundary, and the use of overlapping windows become very important. This technique ensures that the window that pops up alongside another one also covers the area where the entire answer can be captured. Even if an answer is longer than a single window, the use of overlapping windows can provide much accuracy in extraction. This is very helpful when the question is fairly complex or the answer is distributed across several paragraphs.

Evaluation process:

Evaluation of a bert-base-uncased model for extractive question answering is determined by its accuracy. The accuracy of the model is given by the formula: Total accuracy equals correctly predicted samples' sum divided by the total length of the validation set. Other metrics of evaluation including F1 score, precision, and recall were also calculated.
The below chart represents accuracy over time.



As shown in the chart, overall accuracy increases gradually with an increase in the number of epochs. This indicates that the model is improving over time and its capability to predict correctness in the outcome of the tasks is also increasing.
The focal loss function is used for calculating the training loss of the model. This function optimizes the accuracy of the model by determining the answer present as a span of words in the document by multiplying the importance of the minority class. The output logits of the BERT model have start_positions and end_positions displaying the actual start and end positions of the span of the answer within the text given as input.
The training loss plotted with time is shown in the below chart.



Observations show that there is a downtrend in the training loss, which depicts that the model is gradually improving on the training data, and it is becoming more and more accurate in target variable prediction.

WER is the number of errors divided by the total words. The formula for the word error rate, in notation, is Substitution + Insertions + deletions. A substitution typically occurs when a word gets replaced. An insertion is something that has been added which shouldn't be added, and deletion is something that has been left out of the context completely. In real time, there will be noise; if noise increases, WER increases.

GitHub link where WER calculated is calculated: https://github.com/shwethasivakumar/Deep-Learning/blob/main/HW3/HW3_extractive_question_answering_WER-2.ipynb

Therefore, extractive question answering has been performed using the Spoken-SQuAD dataset with a BERT-based uncased model, which was trained and evaluated. The model effectively loads the data through data processing in the form of tokenization and encoding. During the training phase, the system optimizes iteratively using focal loss, and hence the learning rate scheduling has been used to improve performance over epochs. Evaluation metrics such as accuracy, F1 score, precision, and recall give insight into the efficiency. Also, the code executes test procedures that assess the performance of the model on unseen data.