Inter IIT High Prep

BERT

`BERT stands for Bidirectional Encoder Representations from Transformers. It is a transformer-based machine learning technique for natural language processing pre-trained developed by Google.`

I used various methods to approach the problem.

Main Approach

1. After loading the data, a new column is made in the dataframe named ‘answers’ which contains the dictionary with the ‘start index’ and the ‘text answer’
2. train_answers, train_contexts and train_questions are made which are pandas.Series() for the respective columns.
3. A function is developed to add columns ‘answer_start’ and ‘answer_end’ to ‘train_answers’
4. All the rows which don’t have and ‘end_idx’ or have ‘answer_end’ equal to ‘None’ are dropped.
5. A tokenizer is made from a pretrained model `'distilbert-base-uncased'` of the `DistilBertTokenizerFast` Class.
6. train_answers and train_questions are encoded i.e., they are embedded.
7. Then the questions and answers are tokenized and the start and end positions are added to them
8. train_dataset is made on the train_encodings using the SquadDataset with the help of PyTorch.
9. A pretrained model `"distilbert-base-uncased"` is used for training the model on the train_dataset with the help of PyTorch.
10. After this the whole dataset is divided into batch sizes of sixteen i.e. a dataset of around 50000 values is subdivided into 16 batches which would be trained in 3 epochs.
11. After the training our model is ready
12. This model gives us an accuracy of around 65-70 %.

Approach 1

1. `Pre-process the question and paragraphs to create a list of tokens (individual words)`
2. `Use a natural language processing model (BERT) to generate embeddings for the tokens`

3. Compare the embeddings using a similarity measure (cosine similarity) to find the most similar paragraph
4. Use a threshold to determine if the most similar paragraph is "sufficiently" similar to the question
5. Return the paragraph if it is above the threshold
6. Return None if the paragraph is not sufficiently similar to the question

Approach 2

Apart from this rather than using the preprocessing technique, I imported some of the BERT models like BertForQuestionAnswering, BertTokenizer and BERT Model.
In BertForQuestionAnswering there is a pre-trained model, bert-large-uncased-whole-word-masking-finetuned-squad which when used, once the correct paragraph has been detected, gives the exact answer of the question.

1. All the unique paragraphs and their themes are made into a dictionary and appended to a list.
2. Paragraphs are filtered based on the required theme.
3. Then the paragraphs and question are transformed into vectors using TfidfVectorizer
4. After this the similarity between the paragraphs and the question is noted.
5. The one with the highest similarity is selected as the required paragraph.
6. Then the question and answer pairs are encoded using tokenizer i.e. they are embedded.
7. Then these ids are converted into tokens
8. Then the start and end scores are calculated on the tensors of these tokens to find the one with the maximum scores
9. These scores are converted into the index of the tokens. And all the tokens which are present in the given range are then joined and the final output is given.

Approach 3

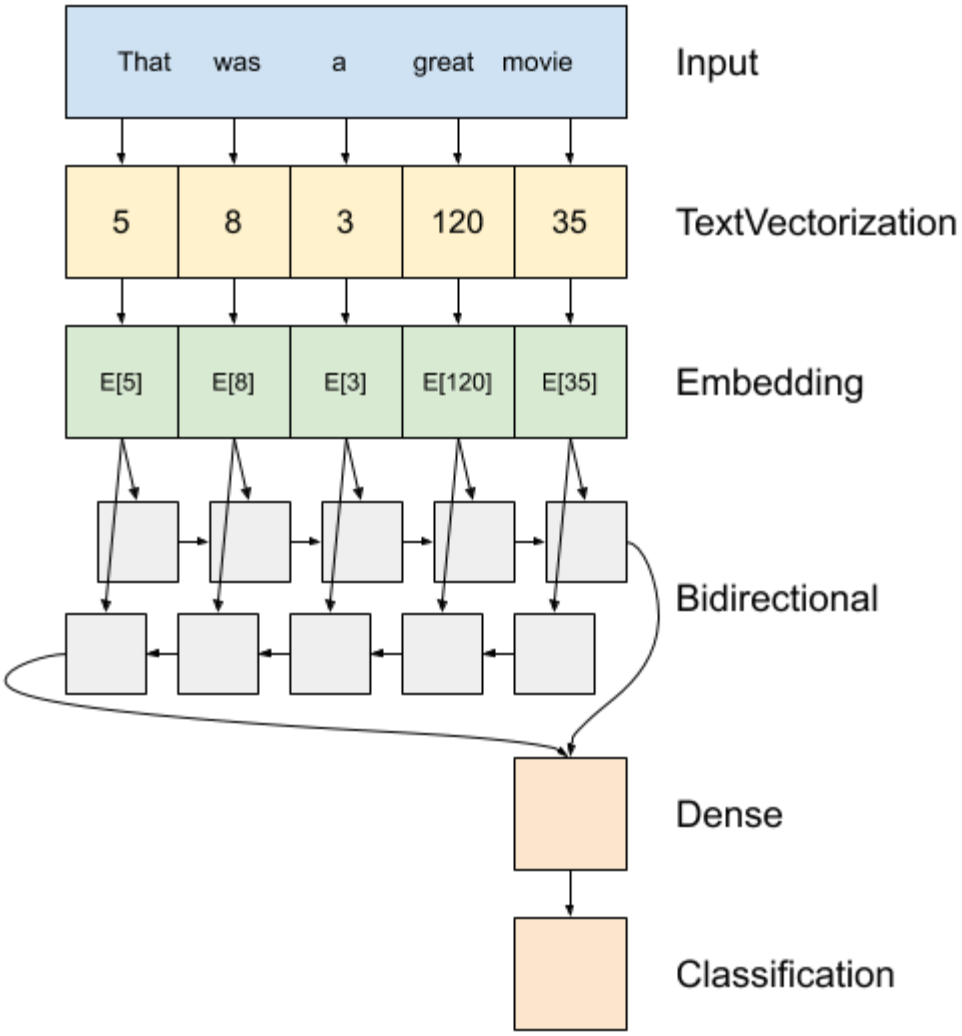
1. Imported Autotokenizer and AutoModel from transformers
2. Made token using the pre-trained model for the given paragraphs

3. Made ‘input_ids’ tokens and ‘attention_mask’ tokens.
   ‘Attention_mask’ tokens are the ones which have been padded upon so that the model does not attend to them
4. Then these tokens are converted into tensors.
5. After this the mean pooled sum operation is applied on the tensors after the attention mask has been applied on the main tensors.
6. After that cosine similarity is calculated for the question and the paragraphs.
7. The one with the highest similarity value is the required paragraph.

Shown below is the general working of the BERT technique. When an input of a sentence is given, we first divide the sentence into separate tokens. These tokens are then further vectorized. Which means they are given a numerical value and a direction and vector operations can be performed on them. After this embedding is done which completes the process of vectorization. Then these embeddings are used to further find the similarity between any two sentences. Here I have used cosine similarity.

$$sim(u, v) = \frac{u \cdot v}{\|u\|\|v\|} = \frac{\sum_{i=1}^{n} a_n b_n}{\sqrt{\sum_{i=1}^{n} u_n^2}\sqrt{\sum_{i=1}^{n} v_n^2}}$$

The larger the value of this similarity, the more probable it is that the answer would be found in thar paragraph


That was a great movie
Input
5 8 3 120 35
TextVectorization
E[5] E[8] E[3] E[120] E[35]
Embedding
Bidirectional
Dense
Classification

s