Project Report

1. Overview

The cutting edge techniques are so prevalent everywhere and contemporary children are fluent in manipulating devices since they have been exposed to devices when they are growing up. Variety educational applications give children a chance to learn by themselves. With the advent of the COVID-19, many children have to stay at home and need contactless education. Most educational applications are based on non-language operations. For example, “OSMO” is one of the kids educational applications being able to interact with users by using image recognition. If applications have the ability to communicate with users, the effect of learning would improve significantly. There are definite demands on applications handling natural language processing. To make educational applications having semantically natural conversations with children, language models should be trained. We proposed building a language model of an educational application for children like virtual teaching assistants.

Question-Answering, which is a computer science discipline within the fields of information retrieval and natural language processing [wiki], enables us to have more semantic conversations between devices and humans. By using Question-Answering techniques in Natural Language Processing, children can question things and get answers by virtual teaching assistants and keep expanding their knowledge without losing their interests.

To aid children to understand knowledge on science topics like Space, we will collect information from ESA-Space for Kids . This site has information about scientific topics. The dataset we will be using should be structured with text, questions and answers. We will create questions and answers from the text.

2. Dataset description and characteristics

Dataset is csv format and answers are extracted from exactly the same words or sentences from text. We will find a ‘start_char_index’ and ‘end_char_indet’ through code (note: not generative or paraphrasing answers). There are a total of 150 samples for the training set with 19 topics. We used 20 samples for a test set. Test set has the same text as the train set but has different question and answer pairs. The below Table 1 shows the number of tokens at each set.

	Training Set (Text, Question)	Test Set(Text, Question)
Total Tokens	44607	5570
Unique Tokens	1265	699
Mean Tokens of Questions	10	11
Mean Tokens of Text	287	268

Table 1. The number of tokens at training and test sets

3. NLP models and techniques utilized

BERT, which stands for Bidirectional Encoder Representations from Transformers. By concurrently conditioning on both left and right context in all layers, BERT is aimed to pre-train deep bidirectional representations from unlabeled text, in contrast to recent language representation models.

BERT Tokenizer [Preprocess]

Tokenizer is the main preprocessing phase, which splits text into tokens based on rules. The tokens are converted into numbers and then, tensors, which are the expected model inputs. In detail, texts are converted into a sequence of tokens, and creates a numerical representation of the tokens, and assembles them into tensors. The tokenizer returns three important values:

- **input_ids** are the indices corresponding to each token in the sentence
- **attention_mask** indicates whether a token should be attended to or not
- **token_type_ids** identifies which sequence a token belongs to when there is more than one sequence. The first sequence, the "context" used for the question, has all its tokens represented by a 0, whereas the second sequence, corresponding to the "question", has all its tokens represented by a 1.

Diverse pre-trained BERT models

`'Bert-large-uncased-whole-word-masking-finetuned-squad'`

We used pre-trained tokenizer with 'bert-large-uncased-whole-word-masking-fine tuned-squad' dataset SQuAD dataset, which is a reading comprehension dataset consisting of questions posed by crowdworkers on a set of Wikipedia articles. The texts are lowercase and tokenized using WordPiece and a vocabulary size of 30,000. This model was trained with Whole Word Masking, which means that all of the tokens corresponding to a word are masked at once.

`'Distilbert-base-uncased-distilled-squad'`

DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT's performances. This model is fine-tuned using knowledge distillation on SQuAD v1.1. [Distilbert]

`'Bert-base-cased-squad2'`

The model distinguishes lowercase and uppercase. Difference between BERT base and BERT large is on the number of encoder layers. BERT base model has 12 encoder layers stacked on top of each other whereas BERT large has 24 layers of encoders stacked on top of each other. [bert-base vs bert-large]

`'bert-medium-squad2-distilled'`

This model is distilled from its teacher model 'bert-large-uncased-whole-word-masking-squad2' and uses haystack's distillation feature for training. This model has an overall f1-score of 72.76. Some of the hyperparameters are as follows: batch_size = 6, epochs = 2, learning rate = 3e-5, temperature = 5

ReTraining-FineTune

There are variables for fine tuning; learning rate, optimizer, batch size.

Optimizer

Adam (Adaptive Moment Estimation) combines RMSProp and Momentum and computes adaptive learning rates. We can obtain a group of parameters from `'param_optimizer = list(model.named_parameters())'` and use these parameters to optimize. The 'betas' option is coefficients used for computing running averages of gradient and its square. The ‘lr’ option stands for learning rate and the ‘eps’ option is a term added to the denominator to improve numerical stability. [Pytorch Adam Ref] **AdamW** implements Adam algorithm with weight decay fix as introduced in Decoupled Weight Decay Regularization.[Hugging Face]

Evaluation

Loss Function

The loss from start and end logits are weighted equally in the loss function.

Evaluation Method

The below is extracted from the Paper : Question and Answering on SQuAD 2.0

To evaluate our models we use the **standard SQuAD performance metrics**: Exact Match (EM) score and F1 score. For our project, we focus on the EM and F1 scores with respect to the dev set.

• **Exact Match**: A binary measure of whether the system output matches the ground truth answer exactly.

• **F1**: Harmonic mean of precision and recall, where precision = (true positives) / (true positives + false positives) and recall = true positives / (false negatives + true positives). F1 score = (2×prediction× recall) / (precision + recall).

(Note: In SQuAD, **precision measures to what extent the predicted span is contained in the ground truth span, and recall measures to what extent the ground truth span is contained in the predicted span**.)

BLEU Paper: Evaluating Question Answering Evaluation

BLEU stands for BiLingual Evaluation Understudy. A BLEU score is a quality metric assigned to a text which has been translated by a Machine Translation engine. BLEU scores a candidate by computing the number of n-grams in the candidate that also appear in a reference.

BERTScore

BERTScore is an automatic evaluation metric for text generation that computes a similarity score for each token in the candidate sentence with each token in the reference sentence. It leverages the pre-trained contextual embeddings from BERT models and matches words in candidate and reference sentences by cosine similarity.

Additional Interface - Pipelines

The pipelines offer an excellent and simple method for using models for inference. Pipelines are classes that encapsulate the majority of the library's sophisticated logic. Pipelines are made of a tokenizer mapping raw textual input to the token and a model to make predictions from the inputs and some post-processing for enhancing the model’s output.

GPT-3

GPT stands for Generative pre-trained Transformer, which is an autoregressive language model that uses deep learning to produce human-like text. Given an initial text as a prompt, it will produce text that continues the prompt. [Wiki]. Researchers at OpenAI described the development of GPT-3 as third-generation "state-of-the-art language model". Instead of fine-tuning the model, we decided to experiment prompt engineering, which is another approach to improve pre-trained models. Prompt Engineering adds informative context into the prompt, helping GPT-3 better understand the scenario and background of the questions. We compared the result using prompt engineering to that without prompt engineering, and we uses two approaches to evaluate the result. The first one is BLEU score, the second one is GPT-3 similarity embedding with cosine similarity.

4. Experiments conducted

[Yoonjung Choi]

4.1 Experiment on BertQuestionAnswering Models for fine-tuning

Training workflow:

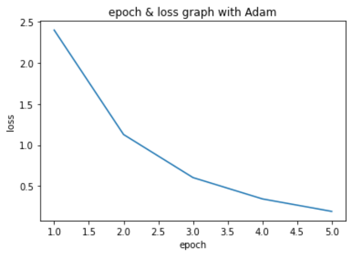
1. Set GPU environment.
2. Prepare data input and output.
3. Load model
4. Forward pass by feeding input data through the model with model.train()
5. Backward pass to update parameters with optimizer.step()
6. Log variables(steps, loss) for monitoring progress
7. Save fine-tuned model

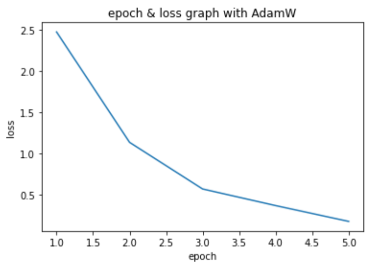
Evaluation workflow:

1. Prepare data input and output
2. Load fine-tuned model
3. Forward pass by feeding input data through the model with model.eval()
4. Log results
5. Save and compare result

● **'bert-large-uncased-whole-word-masking-finetuned-squad'**

This Bert Question Answering model with uncased squad set shows the decrease in training loss of the below plots. Based on initial experiments, We modified a few values for fine-tuning like batch size, learning rate, number of epochs. We used batch size 5 because we faced the issue "CUDA out of memory", used learning late 1e-5 is a very small number to prevent overshooting, and used 5 epochs.





Adam: y = [2.3990, 1.1283, 0.6045, 0.3451, 0.1929] | AdamW: y = [2.4794, 1.1356, 0.5683, 0.3663, 0.1738]

● **'distilbert-base-uncased-distilled-squad'**

When fine tuning on this model, training is very fast and finished quickly rather than the Bert Large model. However, this model cannot respond to some questions.

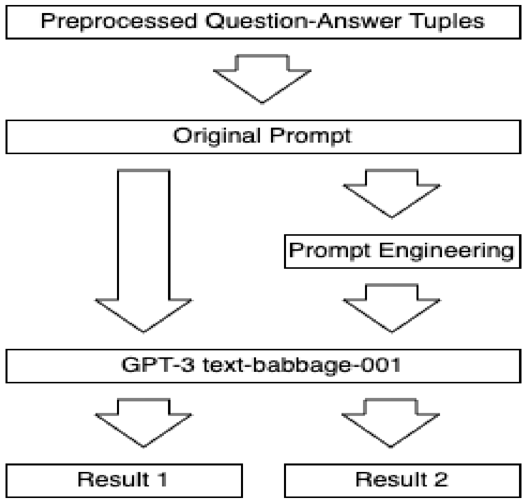
[Jiayao Li]

4.2 Experiment on GPT-3 with Prompt Engineering

We used two of GPT-3's endpoints, text completion and embedding for result generation and evaluation:

- **text-babbage-001**
- **text-similarity-babbage-001**

Diagram of Workflow:



The different results from original and engineered prompts are then each compared against the human-labeled result, by using both BLEU and similarity embedding. Both evaluation approaches reveal an improvement of accuracy by promt engineering.

[Rohith Puvvala Subramanyam]

4.3 Experiment on pre-trained BertQuestionAnswering models without FineTuning

We used multiple BERT models, to evaluate the outcome of the model and compare the F1-score of each model by implementing along with the pipeline. The list of models used is as follows

- **bert-base-cased-squad2**
- **bert-medium-squad2-distilled**

When NLP is implemented on a text or voice call, the entire data set is converted into strings, and the prime string then goes through several phases (the process called processing pipeline.) Depending on voice tone or sentence length, it employs trained pipelines to supervise your input data and recreate the entire string. The component returns to the main string after each pipeline. then moves on to the following elements. The components, their models, and training all affect the capabilities and efficacy.

5. Insights

[Yoonjung Choi]

5.1 Insights from BertQuestionAnswering experiment

From the initial experiment, We realized that QuestionAnsweringModel is a factoid model to extract start and end span from the texts and needed to use a tokenizer and model from the same dataset. We fixed some issues on the dataset where there could be typo or whitespace in question and answers. The training loss with 2 epochs quickly reduced from 2.5549 and to 0.9266. When we take into account that we did an experiment within only 2 epochs, which means there is a possibility of overshooting in case of feeding more data. Thus, we should take a look into the loss to be reduced stably. Also, we found that prediction is short words rather than expressing one sentence. From experiments with changing parameters on model, We learned how to train with optimizer, batch size, epoch and as well as converting raw data into question answering formats. For evaluation, we should declare model.eval() not to update via backpropagation. We compared two different Bert models before and after fine tuning, and evaluated them using BLEU and BERTScore's F1 Average. The result shows the table below. The DistillBert model was really fast trained rather than the Bert Large model, but it cannot answer some questions.

Model	BLEU	BERTScore (F1 Average)
bert-large-uncased-whole-word-masking-finetuned-squad	20.2	0.89
distilbert-base-uncased-distilled-squad	21	0.80

[Jiayao Li]

5.2 Insights from GPT-3

GPT-3 is a generative pre-trained model that is capable of giving answers to questions without context. It performs well on general questions. However, in our case, some of the questions are context-relevant. As a result, GPT-3 is not able to generate answers as expected for those questions. The common approach to overcome this issue is through fine-tuning, but we decided to experiment the prompt engineering method as an alternative to fine-tuning. It appears that answers generated by engineered prompt tend to have a higher similarity to human-labeled answers. We also noticed that BLEU is not best suited for the measurement of generative Q&A model. It labeled some of the highly similar answer tuples with a extremely low score. However, it still shows an improvement of accuracy by using prompt engineering.

Model	BLEU	Mean Embedding Similarity
GPT-3 text-babbage-001	0.59	0.840
GPT-3 text-babbage-001 + prompt engineering	6.47	0.864

[Rohith Puvvala Subramanyam]

5.3 Insights from multiple BERT models

As a result, without making significant task-specific architecture alterations, the pre-trained BERT model may be improved with just one additional output layer to produce cutting-edge models for a variety of tasks, including question answering and language inference. Answers generated by each model for the question which is randomly selected from the dataset as follows

Model	Question	Answer	F1-score
bert-base-cased-squad2	Which is our nearest star?	The sun	0.876
bert-medium-squad2-distilled	Which is our nearest star?	The sun	0.947

6. Conclusions

We explored diverse BERT pre-trained models and learned that each model has specific features, how it works as a code level, and evaluation matrix of models. We also experimented GPT-3 with prompt engineering and similarity embedding. Unlike the factoid Q&A approaches usch as BERT, GPT-3 model makes it more generative or sentences without or with context.

7. Proposed next steps

QuestionAnswering models need to improve with more datasets. For comprehensive functionality, we can combine summarization and question answering for Educational Virtual Assistant Application. In addition, Rasa AI can be applied for interactive conversation.

8. Contribution

Yoonjung Choi	Creation of Dataset 150 samples and Test set 20 samples Experiment on Bert/GPT Writing on Project Proposal,Project Updates, Project Report(Overview/Data Description/ NLP technology/Experiment/Insights/Conclusions/ NextSteps)/PPT
Rohith Puvvala Subramanyam	Research Bert models, Experiment of multiple BERT Writing on Project Report(NLP technology/ Experiment/ Insights of multiple BERT models) / PPT
Jiayao Li	Experiment of GPT-3 with Prompt Engineering and Similarity Embedding Writing on Project Report (NLP technology/ Experiment/ Insights) / PPT