Individual Report

Introduction

In the ever-evolving field of Natural Language Processing (NLP), the development of question-answering (QA) systems represents a significant leap forward in our ability to parse and interact with digital information. This project embarks on the journey of creating a QA system based on reading comprehension, drawing inspiration from the expansive insights presented in "Speech and Language Processing" by Daniel Jurafsky & James H. Martin (2023). Our objective is to craft a system capable of understanding and responding to questions posed in natural language, utilizing text passages to extract relevant answers. This endeavor involves an in-depth exploration of data patterns and the implementation of the BERT model, a groundbreaking development in NLP.

Individual Work

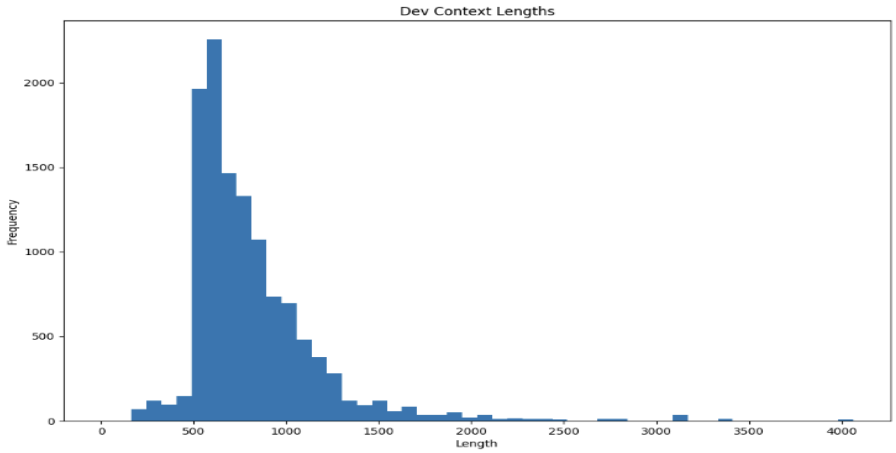
1. EDA(Exploratory Descriptive Analysis)
2. Construction of BERT model with LORA
3. Streamlit with the application of BERT model with LORA

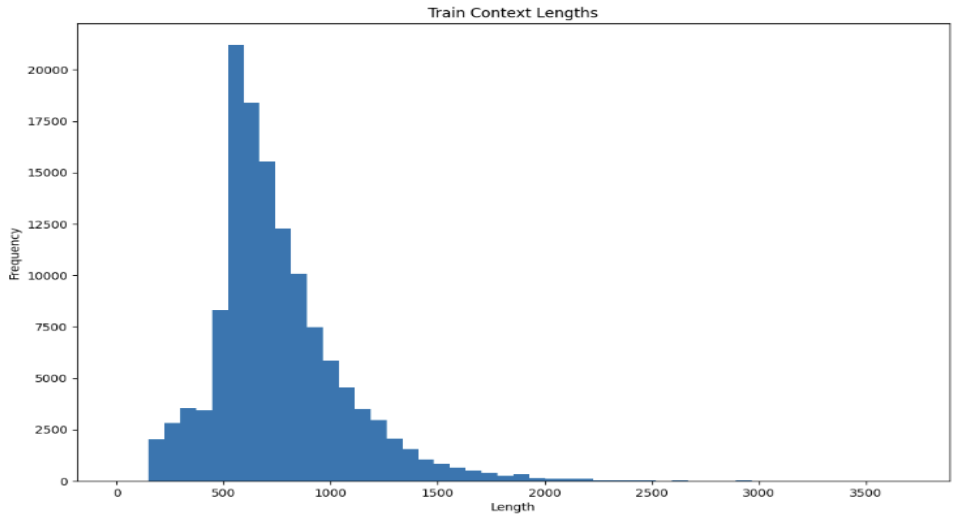
EDA

	SQuAD 1.1	SQuAD 2.0
Train		
Total examples	87,599	130,319
Negative examples	0	43,498
Total articles	442	442
Articles with negatives	0	285
Development		
Total examples	10,570	11,873
Negative examples	0	5,945
Total articles	48	35
Articles with negatives	0	35
Test		
Total examples	9,533	8,862
Negative examples	0	4,332
Total articles	46	28
Articles with negatives	0	28

Table-n: Dataset statistics of SQuAD 2.0, compared to the previous SQuAD 1.1. (Rajpurakar, et al. 2018)

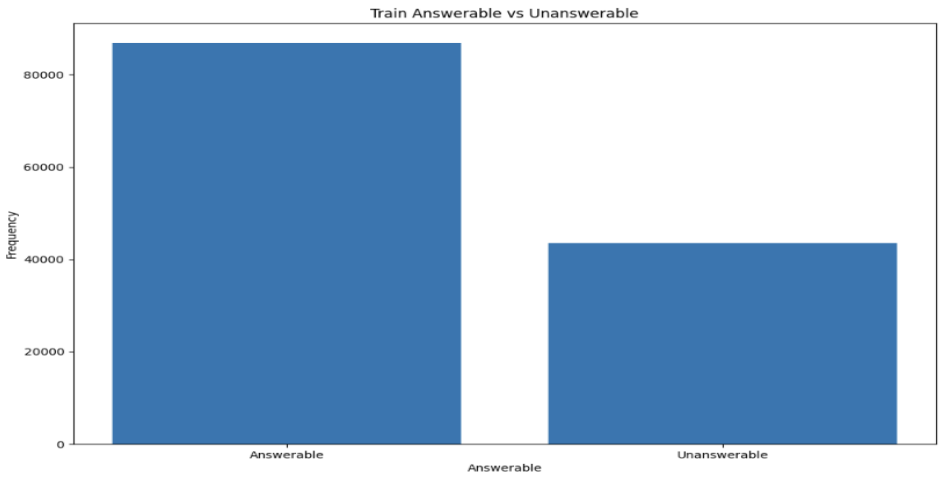
The context length analysis of the train and test set is shown below, whereas it is shown that the frequency of the words are more than 2000 in the train and test set.

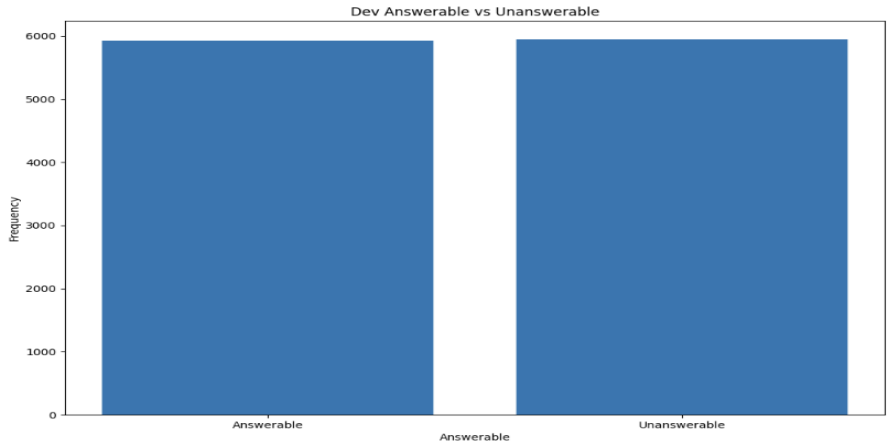




Answerable vs unanswerable:

According to the below image, the answerable questions are more in train than the unanswerable questions.

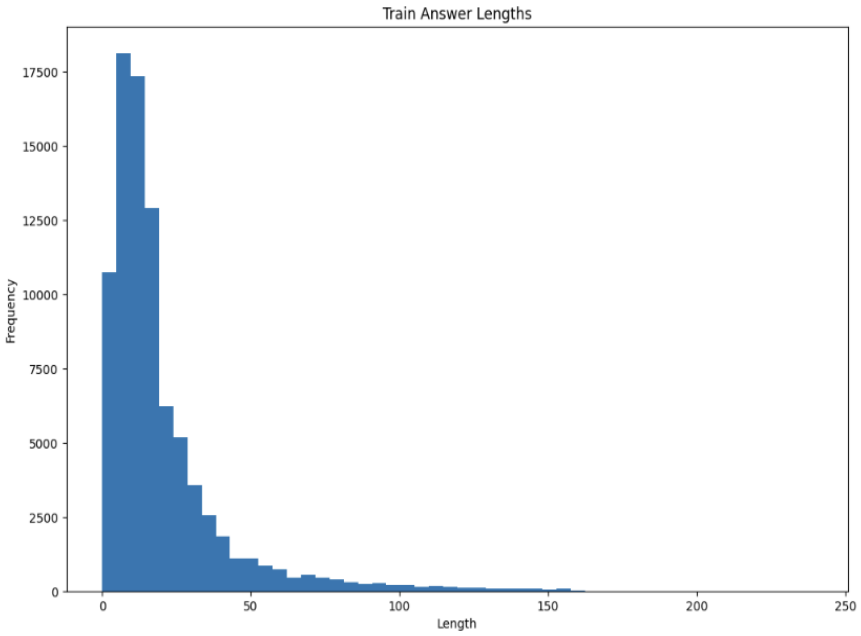




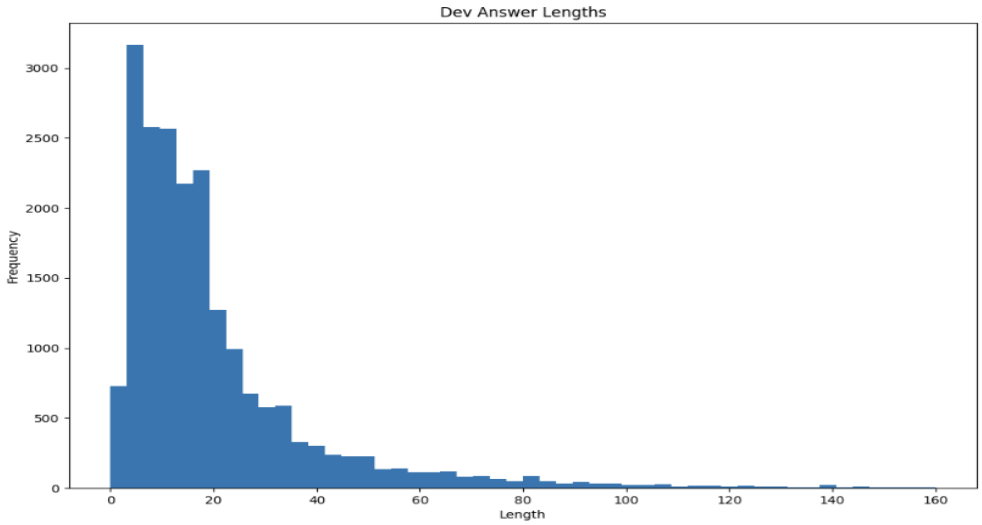
But, in means of dev(test set) it is seen that the data of answerable and unanswerable is balanced and the frequency is more than 5500.

The length of the answers were analyzed and the it is shown below:

The answer lengths of the train set shows that the frequency is high for the answers when the length is between 0 to 20.



The answer lengths of the test set shows that the frequency is high for the answers when the length is between 0 to 20.

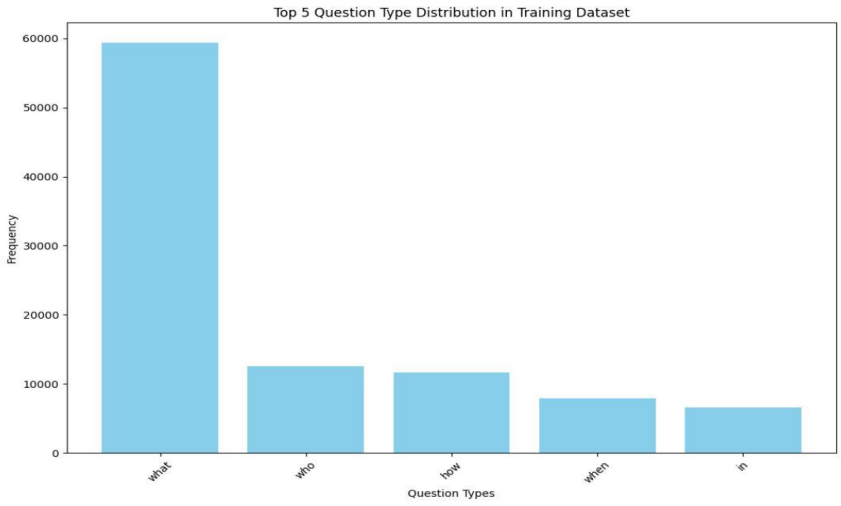


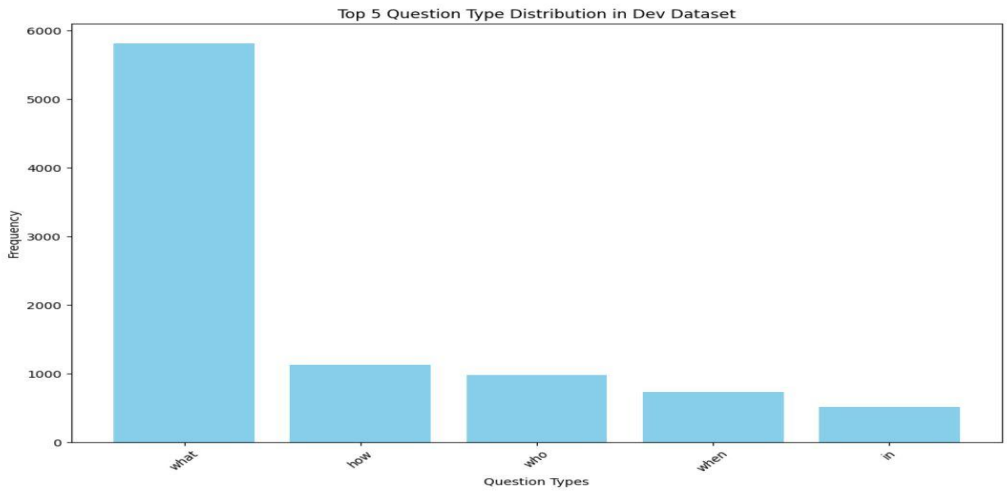
AVERAGE CONTEXT-QUESTION AND ANSWER POSITION ANALYSIS:

The answer context-question similarity in both the sets, answer answer position analysis as well.

```
Average Context-Question Similarity in Training Dataset (GPU): 0.6867929904435697
Average Context-Question Similarity in dev_dataset (GPU): 0.6735267256822767
Average Answer Position in Training Dataset: 41.98904981775197
Average Answer Position in Dev Dataset: 41.24052655725903
```

QUESTION TYPE DISTRIBUTION IN TRAIN AND DEV SETS:





Model Description

BERT Model

BERT, an acronym for Bidirectional Encoder Representations from Transformers, is a revolutionary model in the field of natural language processing (NLP). Initially introduced in Google AI's paper titled “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, BERT has set new benchmarks in various NLP tasks. It represents a significant stride in NLP, marking one of the most notable advancements in this area in recent times.

Developed by Google, BERT is a technique based on the transformer architecture, primarily focusing on pre-training for NLP applications. It was devised by Jacob Devlin and his team at Google in 2018. By 2019, BERT had been integrated into Google's search engine, and by the end of 2020, it was instrumental in processing nearly all English language queries.

At its essence, BERT employs a transformer-based language model. It features a series of encoder layers and self-attention mechanisms. The structure of BERT closely mirrors that of the original transformer model introduced by Vaswani et al. in 2017.

BERT utilizes the Transformer, an innovative attention mechanism adept at understanding the contextual relationships among words (or sub-words) within a text. The standard Transformer model comprises two key components: an encoder that interprets the text input and a decoder that generates predictions for a given task. However, BERT is exclusively focused on creating a language model, so it only employs the encoder part of the Transformer. The intricate functionalities of the Transformer are elaborated in a research paper by Google.

Unlike traditional models that process text in a unidirectional manner (either from left-to-right or right-to-left), the Transformer encoder employed in BERT analyzes the entire word sequence simultaneously. This approach categorizes it as bidirectional, although it might be more apt to describe it as non-directional. This unique feature empowers the model to comprehend the context of each word by considering its position relative to all surrounding words, both to its left and right.

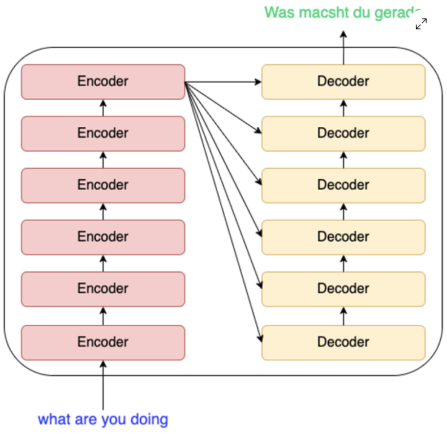


Fig-6: Encoder-Decoder Architecture

In the BERT framework, there are two main steps: **pre-training and fine-tuning.** BERT pre-training uses two training strategies:

Masked Language Modeling(MLM)

Before feeding the whole word sequences into BERT, [mask] tokens are attached with 15% of the words in each sequence. The goal of this part is to attempt to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. The prediction of the world requires three parts:

- Adding a classification layer on top of the encoder output.
- Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
- Calculating the probability of each word in the vocabulary with softmax.

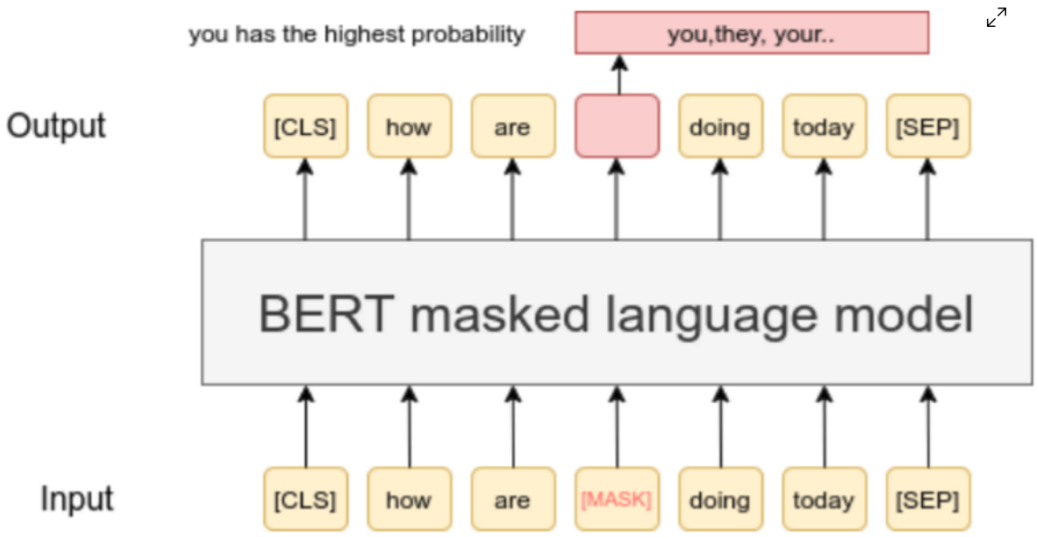


Fig-7: Masked Language Modeling

Next Sentence Prediction (NSP)
In half of the cases (50%), the second sentence is actually the next sentence that follows the first one in the original text. BERT tries to recognize that these two sentences are connected in a logical sequence. In the other half of the cases, the second sentence is just a random sentence taken from somewhere else, which probably doesn't have a logical connection with the first sentence. BERT learns to identify that these sentences don't naturally follow each other.

To assist the model in differentiating the two sentences during training, the input is prepared as follows before it's fed into the model:

- A special token, [CLS], is placed at the start of the first sentence, and another special token, [SEP], is added at the conclusion of each sentence.

- Each token is given an additional marker, known as a sentence embedding, which identifies whether it belongs to the first sentence (Sentence A) or the second sentence (Sentence B). These sentence embeddings are akin to token embeddings but are limited to a choice between two options.

- Additionally, each token is tagged with a positional embedding. This tagging indicates the specific position of the token within the sequence. The theory and application of these positional embeddings are detailed in the Transformer paper.

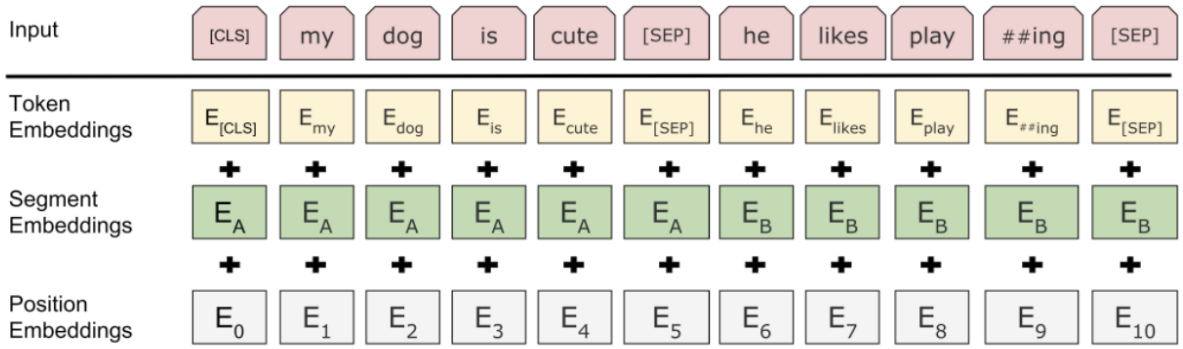


Fig-8: BERT input representation

To determine whether the second sentence is definitely related to the first, the model carries out these actions:

- The complete input sequence is processed by the Transformer model.
- The output from the [CLS] token is converted into a vector with a shape of 2×1. This is achieved by employing a basic classification layer, which utilizes learned matrices of weights and biases.
- The probability of the sequence being the next one (IsNextSequence) is computed using the softmax function.

In fine-tuning part, BERT can be used for different NLP tasks, the only thing need to do is to add small layer to the core model:

Sentiment analysis tasks like classifying positive or negative sentiment can be implemented by adding a classification layer on top of BERT's output for the [CLS] token, similar to what is done for next sentence prediction.

In question-answering datasets like SQuAD, the system must identify the answer span in a context paragraph for an input question. BERT-based QA models can be trained by learning extra vectors to predict the start and end tokens of the answer in the paragraph.

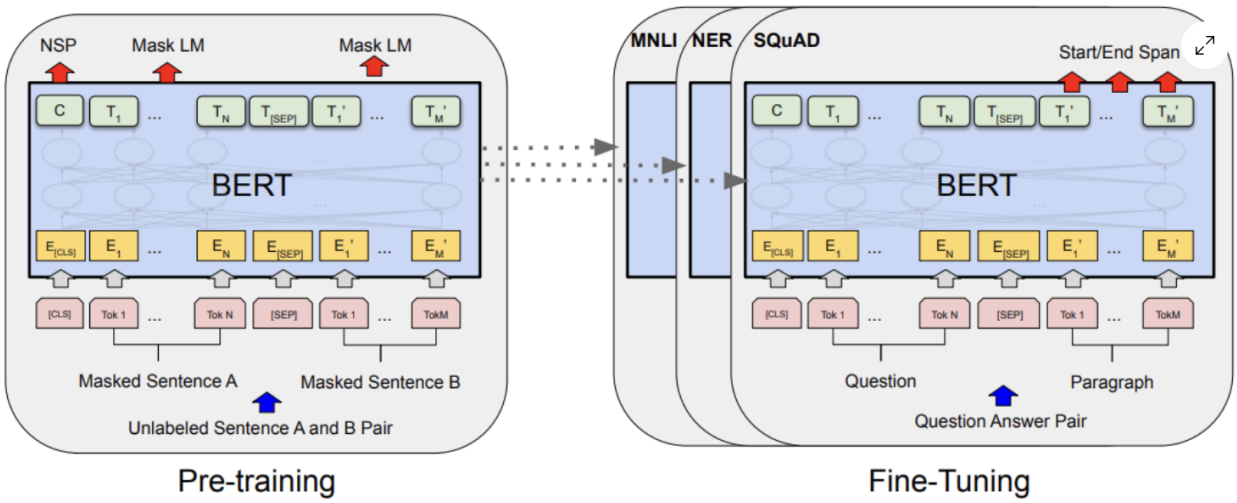


Fig-9: Fine-Tuning BERT

LORA(Low-Rank Adaptation)

LORA was developed to address the high parameter count required for fine-tuning large language models from scratch. It is based on the insight that fine-tuning a pre-trained model does not necessitate updating every weight. Rather, LORA shows it is possible to learn a more compact, task-specific representation of each layer's weights. By learning these lower-dimensional adaptations, LORA provides a way to fine-tune with substantially fewer parameters.

Consider a fully-connected layer with d input units and n output units. The weight matrix W for this layer is of dimensions d x l. For an input x, the layer output Y is calculated as Y = WX.

When fine-tuning with LORA, the weight matrix W is fixed and two additional matrices A and B are introduced, giving:
Y = WX + A*BX

For example, if d=600 and k=3000, W has 600 x 3000 = 1,800,000 weights. LORA's innovation is that matrices A and B can be low rank. If A has dimensions 800 x r and B has dimensions r x 3200, then by adjusting r we can reduce the total parameters.

Setting r=1 gives:
A: 600 x 1
B: 1 x 3000
Total weights = (600 x 1) + (1 x 3000) = 3600

This allows fine-tuning with far fewer parameters than updating W directly.

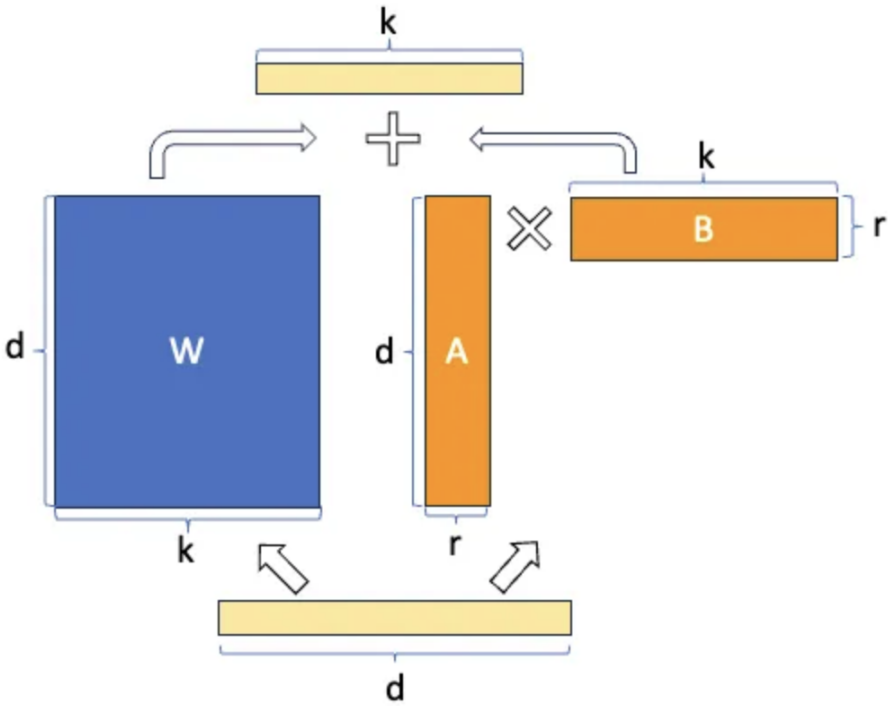


Fig-10: LORA

Experiment Setup

Evaluation Method

Our model's performance is assessed using the Exact Match (EM) and F1 metrics. The F1 score is given greater weight as it is the primary criterion for judgment in the competition. Throughout the training phase, we also monitored the negative log-likelihood to indicate model fit to the data, which helped us identify potential overfitting issues.

BERT with LORA Model

In this experiment setup, the BERT model is meticulously fine-tuned for the SQuAD 2.0 challenge, leveraging a structured approach based on HuggingFace's PyTorch implementation. This approach is tailored for optimizing BERT's performance in question answering tasks.

The fine-tuning process is defined by specific hyperparameters that include the maximum sequence length of inputs and other crucial settings as follows:

- The maximum sequence length: 512
- Learning Rate: 5e-5

- Batch Size: 16
- Number of Epochs: 3 (The training duration was 2 hours)
- LoRA Parameters:
- r: 16
- lora_alpha: 32
- lora_dropout: 0.05

The tokenizer for this model is initialized using the bert-base-uncased configuration, with the addition of an optional LoRA configuration for enhanced adaptation. The training parameters such as learning rate, batch size, and number of epochs are carefully selected for effective training.

The BERT model is initialized specifically for question answering, incorporating an AdamW optimizer with the predefined learning rate. The training function orchestrates the training loop, focusing on efficiency and includes evaluation steps to monitor model performance, particularly using the F1 score as the key metric. Losses during training are tracked to ensure the model is fitting appropriately to the data and to identify any potential overfitting.

A fine-tuning step is conducted using Hugging Face's Trainer API, streamlining the training process and encompassing model saving functionalities. A prediction function is also established for running inference on new questions and contexts, employing the fine-tuned model to accurately generate answers.

Results:

• All params : 109,484,548
• Trainable Parameters: 54% 591,362
• Training Loss(epoch 1 -3): 2.6399 1.6731 1.4301
• Valid Loss(epoch 1 -3): 1.7555 1.3669 1.2309
• F1 Score: 0.50989

The integration of BERT with LoRA showcased a balance between powerful language processing capabilities and computational efficiency. Our experimental results, including training and validation losses and a promising F1 score, demonstrate the effectiveness of our model in understanding and responding to complex questions.

This endeavor not only highlights the evolving landscape of NLP but also opens up possibilities for more sophisticated and efficient question-answering systems. It exemplifies the potential of combining advanced models like BERT with innovative techniques like LoRA, setting a precedent for future research and applications in the realm of natural language understanding.