SQuAD 2.0 Project Report

George Washington University

Columbian College of Arts & Sciences

DATS 6312 Natural Language Processing

Final Project - Group 7

Members: Cody Yu, Pon Swarnalaya Ravichandran, Ei Tanaka

Date: December 11, 2023

Table of Contents

1. Introduction

In the dynamic realm of Natural Language Processing (NLP), the advent of question-answering (QA) systems marks a significant stride in our ability to interact with and process digital information. This project is dedicated to developing a reading comprehension-based QA system inspired by the comprehensive insights in "Speech and Language Processing" by Daniel Jurafsky & James H. Martin (2023). We aim to create a system that can interpret and respond to questions posed in natural language, drawing answers from provided text passages.

1.1 Question Answering Task - Extractive QA

In the Natural Language Processing (NLP) field, extractive Question Answering (QA) is a pivotal task involving locating the answer to a question within a specified text passage. This task is inherently challenging, as it requires the system to comprehend the posed question and accurately extract the specific text portion that contains the answer. As detailed in Hugging Face's documentation and task library (n.d.), extractive QA demands the capability to sift through extensive text and pinpoint information that precisely responds to the query.

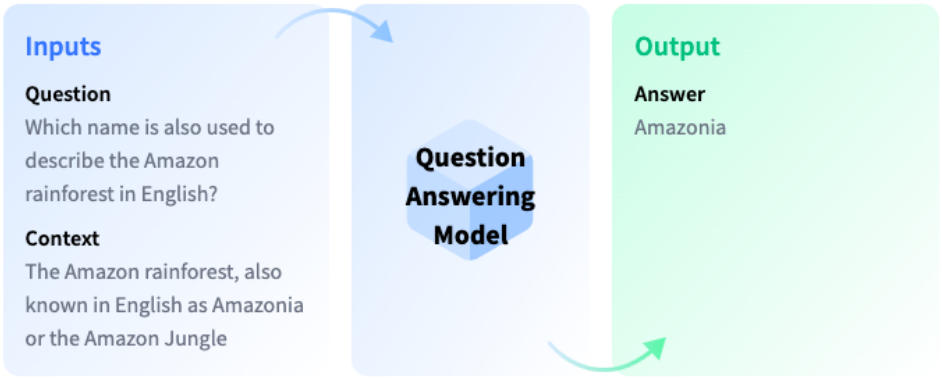


Fig-1: Question Answering Task (Hugging Face. 2001)

Jurafsky and Martin (2023), in their seminal work, "Speech and Language Processing," elucidate the complexities of extractive QA, highlighting the necessity for advanced NLP techniques and models. These models are crucial for understanding the context and semantics embedded in both the question and the passage, thus enabling the identification of the exact text span that answers the question. Extractive QA is particularly vital in scenarios necessitating factual answers directly sourced from the provided text, such as in academic research or specific information retrieval tasks.

In our project, we embrace the challenges of extractive QA by training our model on the SQuAD 2.0 dataset. This dataset, encompassing diverse questions and passages, provides a comprehensive framework for the system to learn from varying contexts and question types. The model is meticulously trained to parse the subtleties of language in questions and

passages, enhancing its ability to discern and extract the relevant answers accurately. This endeavor underscores the significance of sophisticated text processing and comprehension in NLP, laying the groundwork for more intelligent and adept information retrieval systems.

1.2 Dataset

SQuAD 2.0 is an innovative dataset designed to advance the field of machine reading comprehension, a key area in natural language understanding. This dataset extends its predecessor, SQuAD 1.1, by incorporating a significant new feature: 53,775 unanswerable questions crafted by crowdworkers. These questions are based on the same paragraphs used in SQuAD 1.1 but are specifically designed to be relevant to the paragraph content while simultaneously containing plausible yet incorrect answers. This design represents a substantial shift from the previous version, which only included answerable questions, ensuring a correct answer could be found within the provided text. Two such examples are shown in the blow.

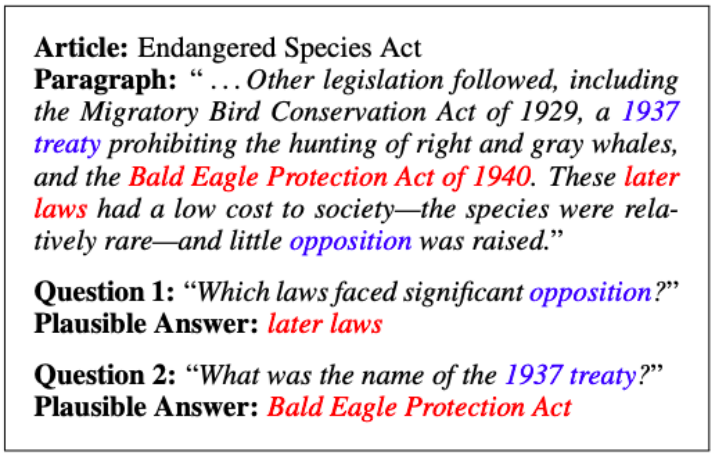


Fig-2: Two unanswerable questions written by crowdworkers and plausible (but incorrect) answers. Relevant keywords are shown in blue. (Rajpurakar, et al. 2018)

The primary objective behind the development of SQuAD 2.0 is to challenge and enhance the capabilities of machine learning models in discerning when a correct answer is not present in the given text. This represents a more complex task than the one posed by SQuAD 1.1, which required models to identify the text span most related to the question. SQuAD 2.0 aims to foster a deeper level of comprehension and critical analysis, pushing the boundaries of what machine learning models can achieve in understanding and interpreting the text.

The dataset has been rigorously tested and proven to be both challenging and high quality. When evaluated using state-of-the-art machine learning models, SQuAD 2.0 yields a significantly lower F1 score of 66.3% compared to the human benchmark of 89.5% F1 score. This gap highlights the enhanced difficulty of the dataset, as the same model architecture achieved an 85.8% F1 score on SQuAD 1.1, much closer to human performance. The unanswerable questions in SQuAD 2.0 have also been shown to be more challenging than those created by other methods, such as distant supervision or rule-based approaches.

SQuAD 2.0 has been made publicly available and is now the primary benchmark on the official SQuAD leaderboard. The introduction of this dataset marks a significant step forward in the pursuit of advanced reading comprehension systems. It is expected to drive the development of models that better understand the text and recognize the boundaries of their knowledge, a crucial aspect of proper language comprehension.

In creating the SQuAD 2.0 dataset, crowdworkers from the Daemo crowdsourcing platform were employed to generate unanswerable questions. Their task involved creating up to five questions per paragraph from articles in SQuAD 1.1, stipulating that these questions could not be answered based on the paragraph alone. Despite this, the questions had to contain plausible answers and reference entities within the paragraph. To guide the process, workers were shown corresponding questions from SQuAD 1.1 and encouraged to make the unanswerable questions resemble the answerable ones. Each worker dedicated approximately 7 minutes to each paragraph and received $10.50 per hour compensation.

Questions from workers who wrote fewer than 25 questions per article were discarded to maintain quality. This approach helped filter out contributions from those who struggled with the task. The dataset was then divided into training, development, and test splits, following the same article partition as SQuAD 1.1, combining new and existing data. In the development and test sets, articles without unanswerable questions were removed, leading to an approximately equal ratio of answerable and unanswerable questions. However, The training data had about twice as many answerable questions as unanswerable ones.

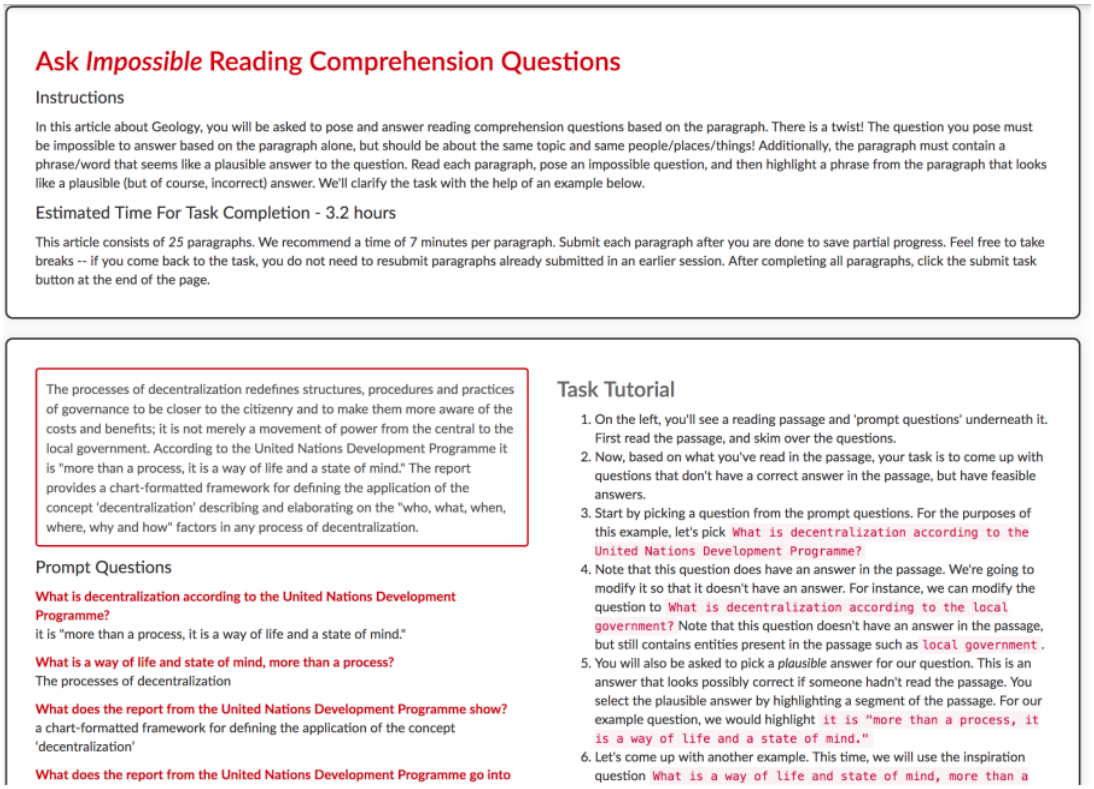


Fig-3: The instructions are shown to crowdworkers at the beginning of each question writing task. (Rajpurakar, et al. 2018)

For dataset validation, additional crowdworkers were hired to answer all questions in the development and test sets of SQuAD 2.0. They were presented with entire articles and associated questions, which were a mix of answerable and unanswerable types. Workers had to highlight the answer in the paragraph or mark the question as unanswerable, with an average time of one minute spent per question. Multiple answers were collected for each question to ensure accuracy and reduce response variance. The final answer was determined through a majority vote, with a preference for shorter answers and a tendency to resolve ties by opting to provide an answer. On average, 4.8 responses were gathered per question. This method contrasts with the evaluation of SQuAD 1.1, where only a single human's performance was assessed, likely leading to an underestimation of human accuracy in the earlier dataset.

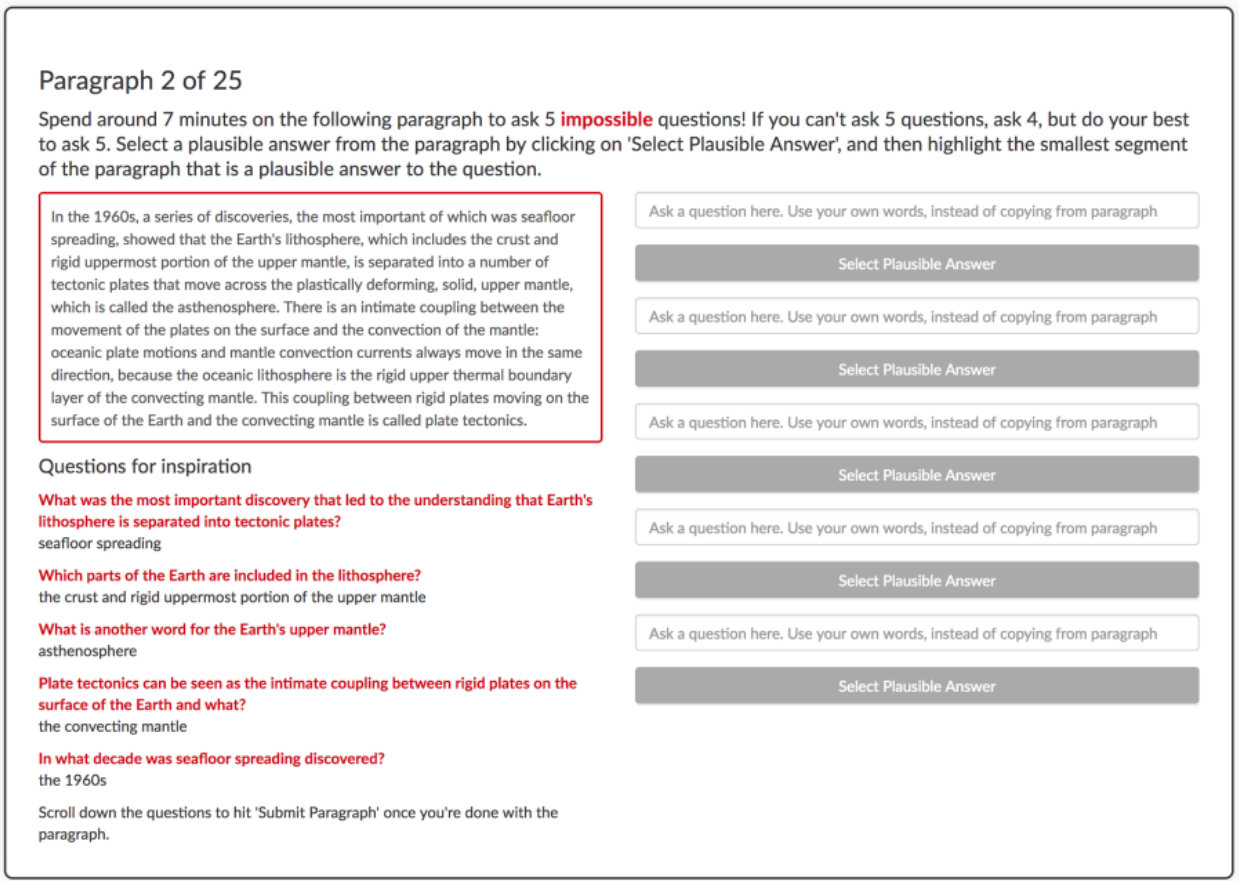


Fig-4: The interface crowdworkers used to write unanswerable questions and annotate plausible answers. (Rajpurakar, et al. 2018)

2. EDA

Table 1 presents a comparative analysis of dataset statistics for SQuAD 1.1 and SQuAD 2.0, highlighting the evolution of the QA datasets over time. The SQuAD 2.0 dataset has expanded significantly, offering 130,319 training examples compared to 87,599 in SQuAD 1.1. Notably, SQuAD 2.0 introduces the concept of 'negative examples'—questions for which the provided text does not contain an answer—with a substantial 43,498 such instances in the training set, compared to none in SQuAD 1.1. This development echoes the total number of articles with negatives, which stands at 285 for training, whereas SQuAD 1.1 did not incorporate this feature. The development set in SQuAD 2.0 also presents an increase, with 11,873 total examples and 5,945 negative examples, as opposed to 10,570 total examples and no negative examples in SQuAD 1.1. The number of articles containing negatives in the development set is 35 for SQuAD 2.0, a new addition to the dataset structure.

For the test set, SQuAD 2.0 offers 8,862 total examples, which is fewer than SQuAD 1.1's 9,533 and includes 4,332 negative examples, introducing a new layer of complexity to the dataset. The total articles are fewer in SQuAD 2.0, standing at 28 compared to 46 in SQuAD 1.1, yet all contain negatives, unlike SQuAD 1.1.

This evolution in dataset composition from SQuAD 1.1 to SQuAD 2.0 indicates a shift towards more challenging and nuanced question-answering tasks. This reflects a trend in NLP research that aims to create models capable of understanding when no answer is available within a given text.

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

2.1 Context Length Analysis

We delved into the context length distribution for both the train and development sets of the SQuAD dataset. These distributions are critical as they inform us about the variability and typical lengths of passages that our model will encounter.

Fig. 5 illustrates the context length distribution in the training set. The histogram shows a right-skewed distribution, indicating that most contexts contain fewer than 500 tokens, sharply declining as token length increases. A long tail extending towards the 3500 token mark suggests that while more protracted contexts are less frequent, they are still a significant consideration for model input design.

Similarly, Fig. 6 depicts the context length distribution in the development set. This distribution exhibits a right skew, with most context lengths falling below 500 tokens. The tail for the development set is slightly shorter, with fewer instances extending beyond the 3000 token length compared to the training set.

Both distributions suggest that a substantial portion of the contexts are concise enough to fit within the typical input length limitations of transformer-based models like ELECTRA. However, more extended contexts must be addressed by implementing strategies like truncation or sliding window techniques to ensure that the model can effectively handle inputs of varying lengths without losing critical information.

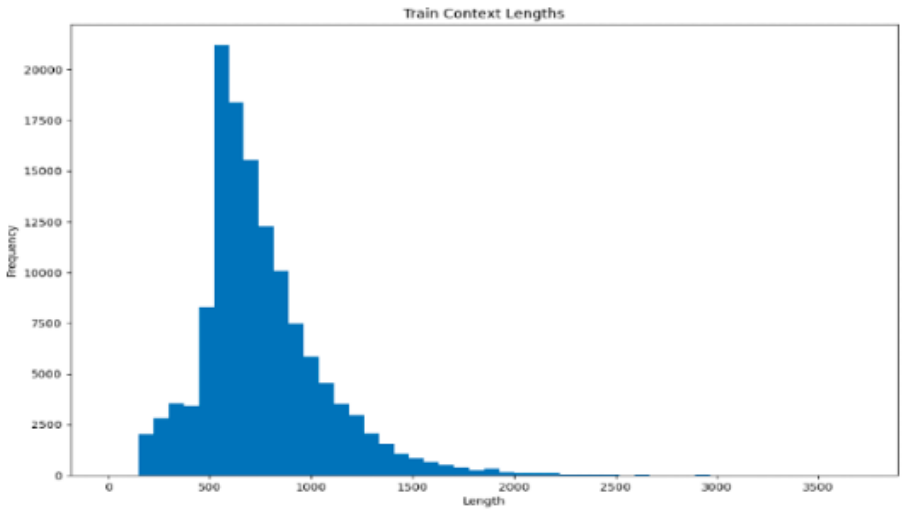


Fig-5: The Context length distribution in the train set

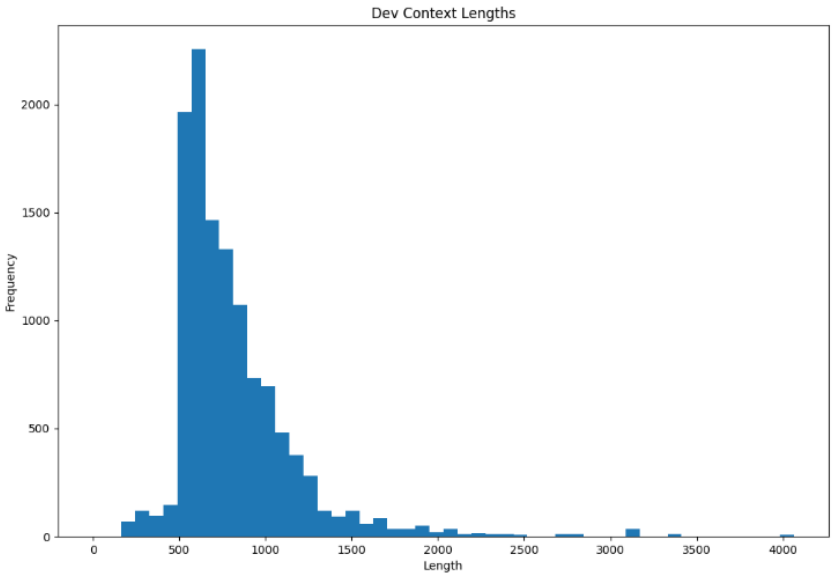


Fig-6: The Context length distribution in the dev set

2.2 Question Length Analysis

The Question Length Analysis provides an in-depth look into the distribution of question lengths across the SQuAD dataset. This analysis is instrumental in understanding the typical question structures and guiding data preparation for model training.

Fig. 7 depicts the question length distribution within the development set. The histogram shows a normal distribution, peaking around 50 tokens before tapering off. This suggests that most questions in the development set are moderately concise, which is beneficial for processing with NLP models that may have limitations on input size.

Fig. 8, illustrating the question length distribution in the training set, presents a starkly different picture. The histogram here is highly skewed, with an overwhelming majority of questions being very short, clustering within a narrow range at the lower end of the token count. This could imply a heavy concentration of succinct questions requiring less contextual information for the models to locate answers.

The disparity in distribution shapes between the training and development sets could impact model generalization. If a model is trained primarily on shorter questions but evaluated on a set with more variable lengths, its performance might not be optimal. Therefore, aligning the training and development data distributions is crucial to ensuring the model can effectively handle a wide range of question lengths.

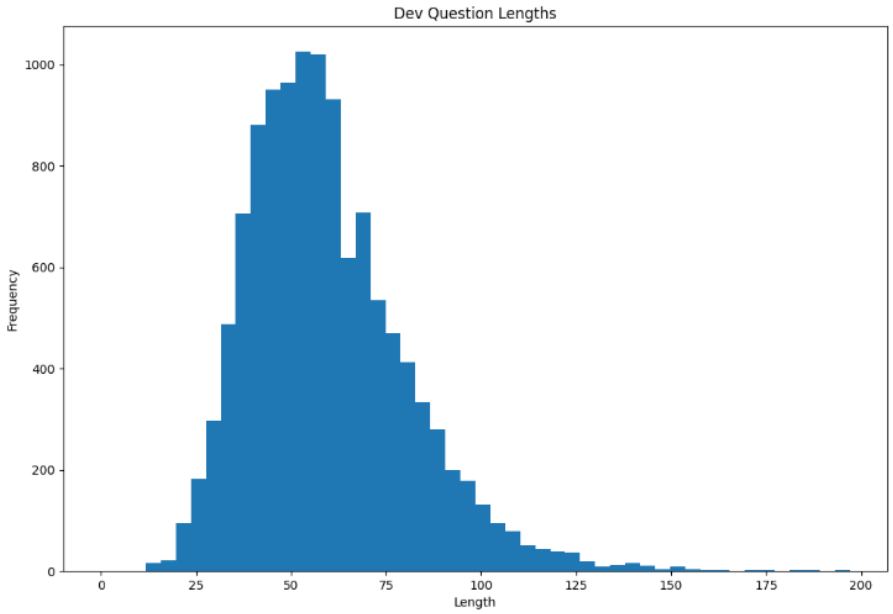


Fig-7: The Question Length Distribution in the Dev Set

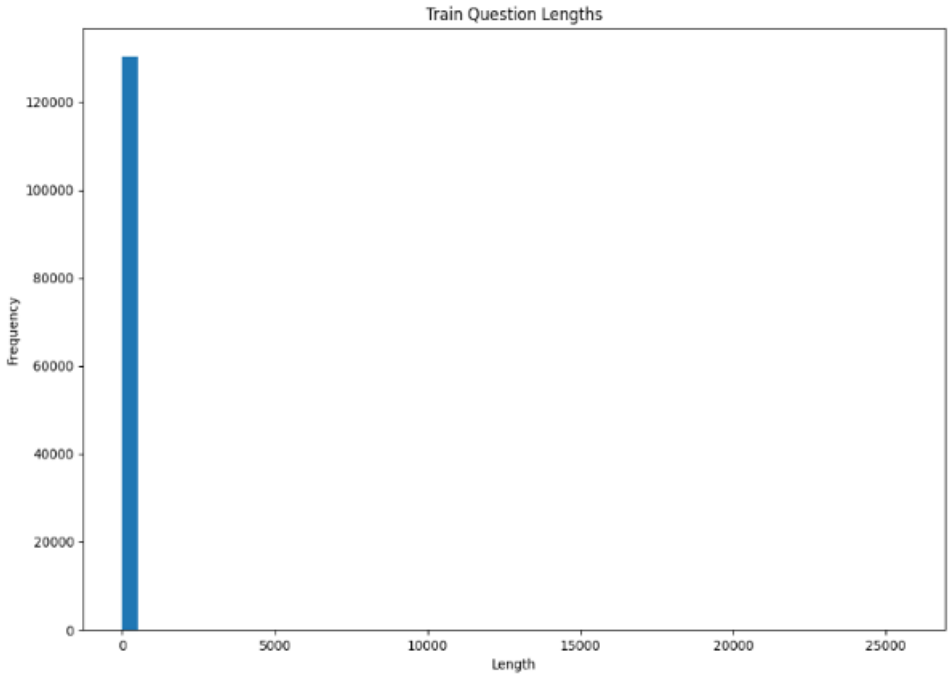


Fig-8: Question Length Distribution in the Training Set

2.3 Answer Length Analysis

This analysis involves studying the number of words, characters, or other relevant metrics that make up the answers provided for a given set of questions. The goal is to gain insights into the characteristics of answers and understand how they vary in length across different contexts.
An answer length analysis plot typically involves visualizing the distribution of answer lengths in a dataset.

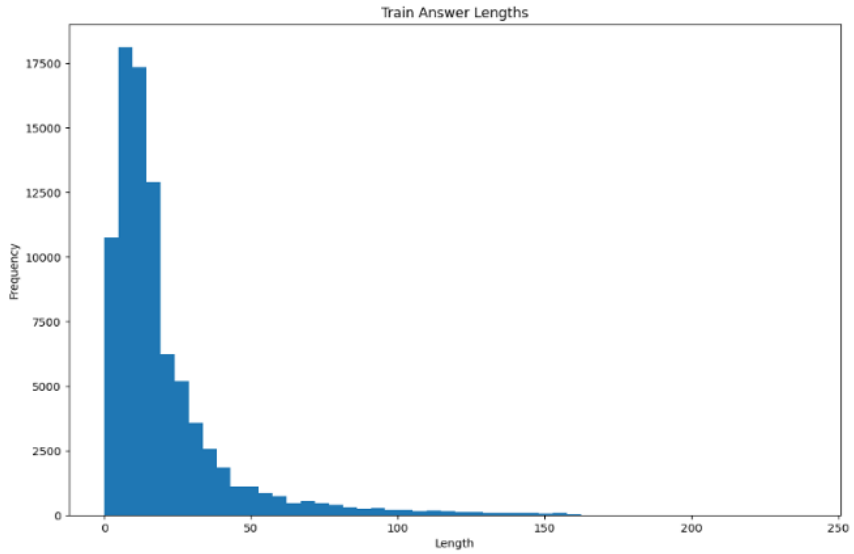


Fig-9: The Answer Length Distribution in the training set

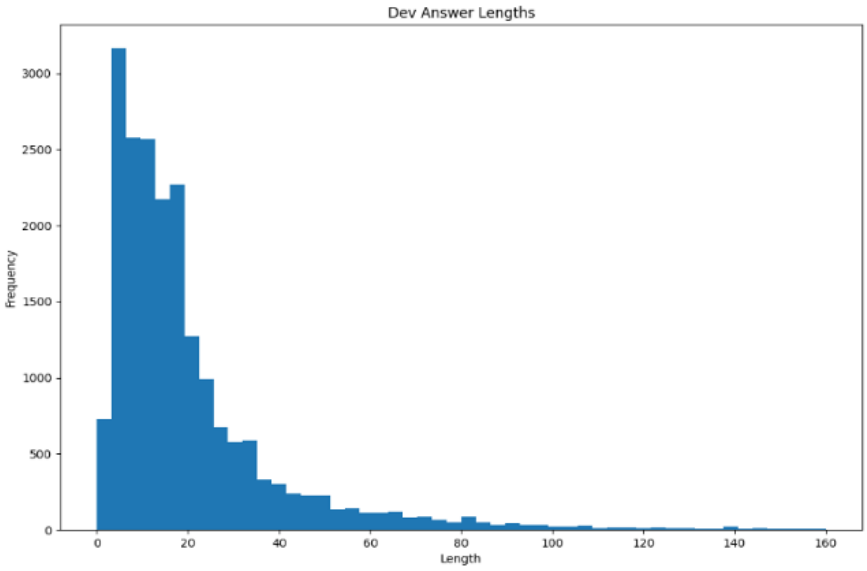


Fig-10: The Answer Length Distribution in the dev set

Typically, the SQuAD 2.0 Is different from the earlier SQuAD 1.1 because of this complexity and the inclusion of unanswerable questions in it. So let us draw some insight about the ans vs units.

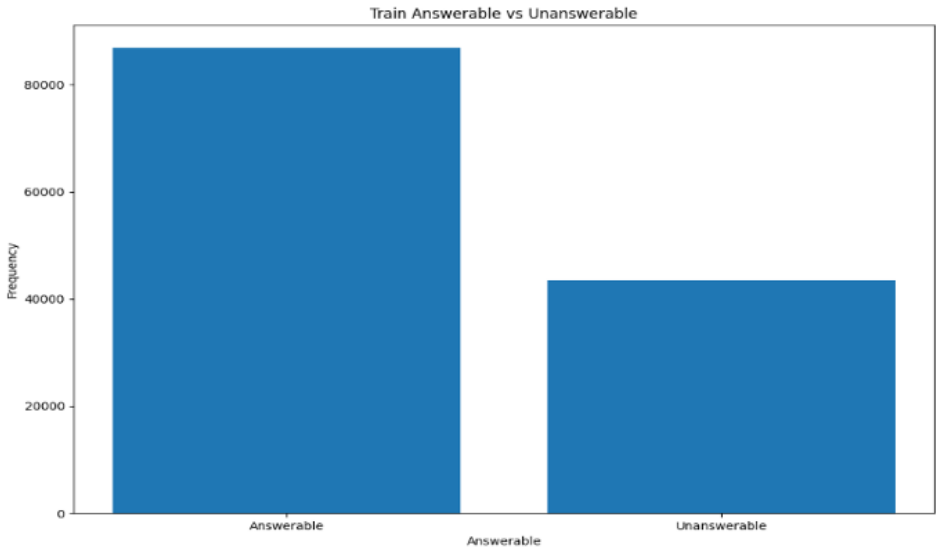


Fig-12: The number of Answerable vs. Unanswerable Questions in the Training set

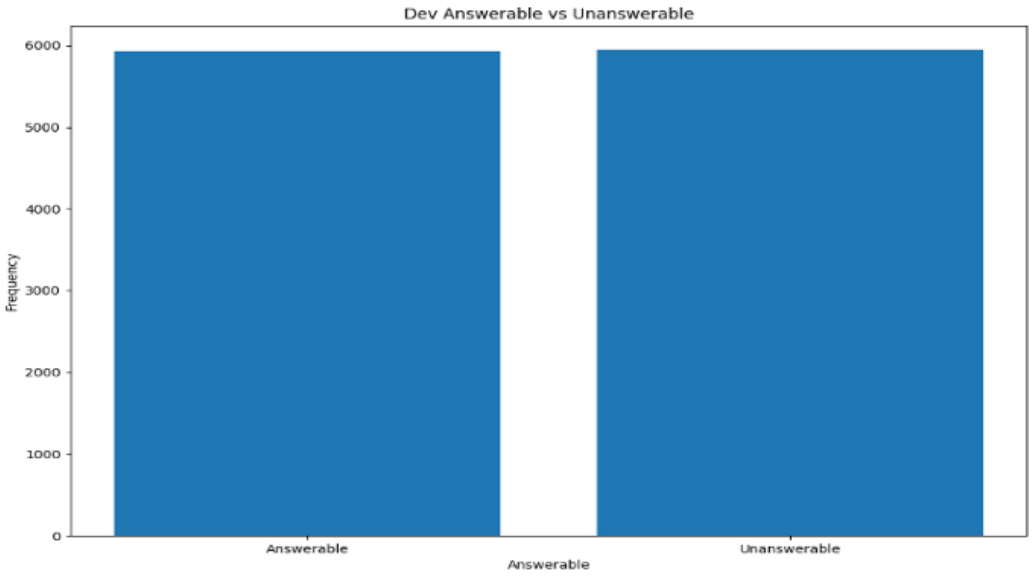


Fig-13: The number of Answerable vs. Unanswerable Questions in the dev set

The above plot representing the statistic between the answerable and unanswerable shows that the train set has more than 80000 answerable questions and more than 40000 unanswerable questions. In the test set, the answerable and unanswerable questions are equal, which makes the prediction more precise.

3. Model Description

3.1 ELECTRA Model

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) represents a significant shift in pre-training text encoders in Natural Language Processing. Introduced by Clark, Luong, Le, Manning, and others (2020), ELECTRA diverges from the traditional denoising autoencoder methods, such as those employed by BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019), which mask a portion of the input tokens and train the network to predict these masked tokens.

ELECTRA innovates on this concept by introducing replaced token detection as its pre-training task. Unlike the masked language modeling (MLM) approach, which learns from only a fraction of the tokens (typically 15%) in an input sequence, ELECTRA corrupts the input by replacing some tokens with samples from a generator model. The network is then trained as a discriminator to identify whether each token in the input is original or a replacement. This methodology addresses a mismatch in BERT—where the network encounters artificial [MASK] tokens during pre-training but not during fine-tuning for downstream tasks.

One of the critical advantages of ELECTRA's approach is its computational efficiency. Since the model learns from all input tokens rather than just a tiny subset, it demonstrates a more efficient use of computational resources. Clark et al. (2020) also note that ELECTRA, when trained on datasets like GLUE (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016), outperforms MLM-based methods like BERT and XLNet given the same model size, data, and compute. The approach is also notable for its scalability and effectiveness, with variants like ELECTRA-Small and ELECTRA-Large achieving high performance on benchmark tasks, showcasing compute and parameter efficiency compared to existing generative approaches in language representation learning.

3.1.1 Pre-trained Method

The methodology centers on replaced token detection, a novel approach Clark et al. (2020) introduced in their ELECTRA model. This technique utilizes two neural networks, a generator and a discriminator, based on the Transformer architecture. The generator, tasked with masked language modeling, randomly masks tokens in an input sequence and tries to predict their original identities. The discriminator, conversely, is trained to distinguish between authentic tokens and those replaced by the generator.

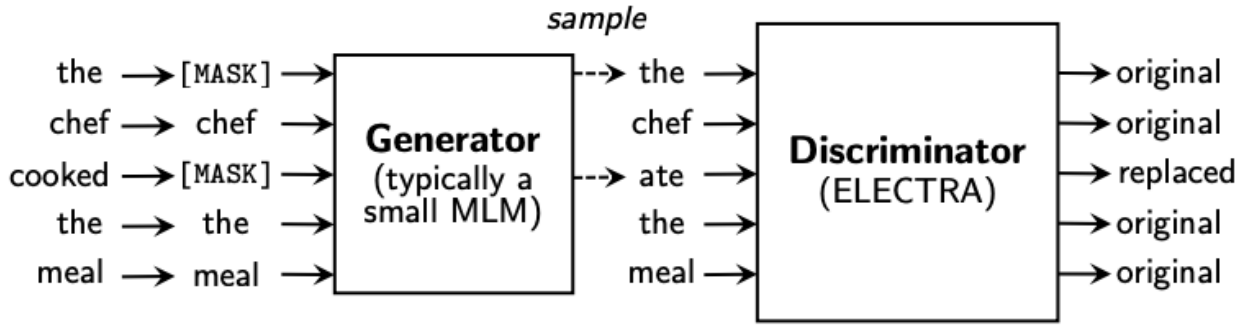


Fig-[n]: An overview of replaced token detection [Adapted from “ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS,” by K. Clark et al., 2020, Figure 2].

A vital element of this approach is creating a corrupted input sequence, where tokens masked by the generator are replaced with its predictions. The discriminator is trained on this sequence to identify tokens that align with the original input. This method diverges from Generative Adversarial Networks (GANs) in significant ways. Notably, tokens accurately predicted by the generator are considered "real" by the discriminator. Additionally, the generator is trained using maximum likelihood, as adversarial training is challenging due to backpropagation complexities associated with sampling.

$$p_G(x_t|\boldsymbol{x}) = \exp\left(e(x_t)^T h_G(\boldsymbol{x})_t\right) / \sum_{x'} \exp\left(e(x')^T h_G(\boldsymbol{x})_t\right)$$

Equation 1: The generator outputs the probability of generating the particular token x_t with the softmax layer for a given portion t. Where e denotes the embedding, x denotes a sequence of input, and h denotes a contextualized vector representation.
By K. Clark et al., 2020, Equation 1.

$$D(\boldsymbol{x}, t) = \text{sigmoid}(w^T h_D(\boldsymbol{x})_t)$$

Equation 2: The discriminator predicts whether the token x_t is “real” for a given portion t.
By K. Clark et al., 2020, Equation 2.

Equation 3 describes the process of generating training data for the pre-training task. It consists of two primary operations involving the generator G and the creation of a corrupted version of the input sequence for the discriminator D to analyze.

Firstly, a set of indices m is randomly selected from the uniform distribution spanning the length of the input sequence, where i ranges from 1 to k, and k represents the number of tokens to be masked, typically 15% of the total tokens in the sequence. These indices correspond to the positions of the tokens in the input sequence x that will be masked.

Once the positions to be masked are determined, the generator G predicts replacements $\hat{x}_{i,}$ for the actual tokens at these positions. The generator's predictions are based on probabilities $P_G(x_i|\, x^{masked})$ calculated given the masked input x^{masked}. The masked input is obtained by replacing the selected tokens in the original input sequence x with a [MASK] token, a process denoted as $x^{masked} = REPLACE(x, m, [MASK])$.

Finally, the corrupted sequence $x^{corrupt}$ is produced by replacing the masked tokens in the original sequence x with the tokens predicted by the generator $\hat{x}$. The discriminator D uses $x^{corrupt}$ to learn to differentiate between the 'real' tokens from the input sequence and the 'fake' tokens generated by

$$m_i \sim \text{unif}\{1, n\} \text{ for } i = 1 \text{ to } k \qquad \boldsymbol{x}^{\text{masked}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \texttt{[MASK]})$$
$$\hat{x}_i \sim p_G(x_i|\boldsymbol{x}^{\text{masked}}) \text{ for } i \in \boldsymbol{m} \qquad \boldsymbol{x}^{\text{corrupt}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{x}})$$

Equation 3: The process of generating the training data.
By K. Clark et al., 2020, Equation 3

The goal is to minimize a combined loss function over a substantial corpus, considering both the generator's masked language modeling loss and the discriminator's loss. Crucially, the discriminator's loss is not back propagated through the generator. Post-pre-training, the focus shifts to fine-tuning the discriminator for downstream tasks.

$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i|\boldsymbol{x}^{\text{masked}})\right)$$
$$\mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\mathbb{1}(x_t^{\text{corrupt}} = x_t)\log D(\boldsymbol{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t)\log(1 - D(\boldsymbol{x}^{\text{corrupt}}, t))\right)$$

Equation 4: Loss functions of Generator G and Discriminator D.
By K. Clark et al., 2020, Equation 4

$$\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D)$$

Equation 5: The combined loss to minimize.
By K. Clark et al., 2020, Equation 5

3.2 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.

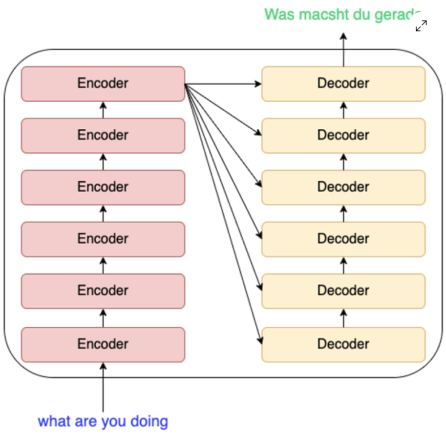


Fig-6: Encoder-Decoder Architecture

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.

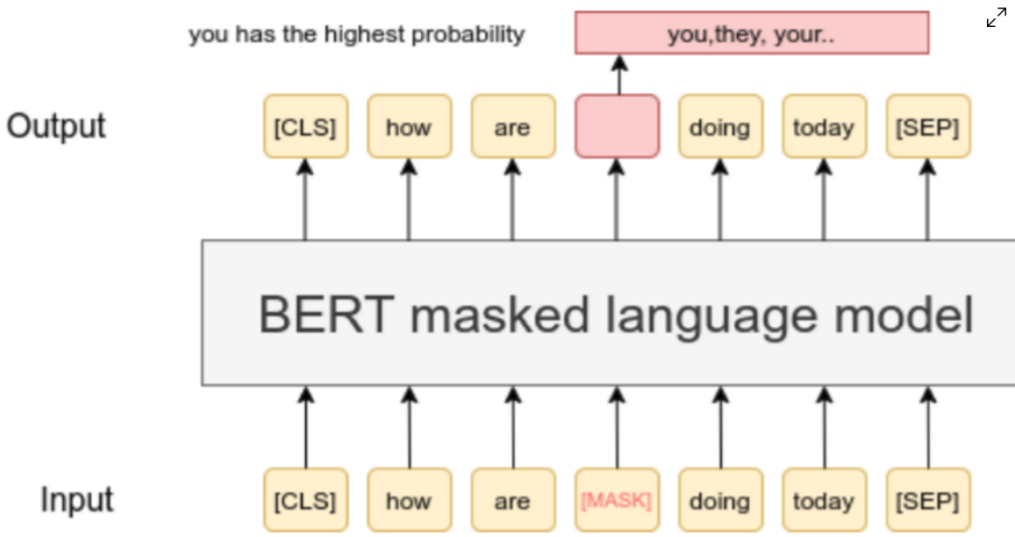


Fig-7: Masked Language Modeling

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.

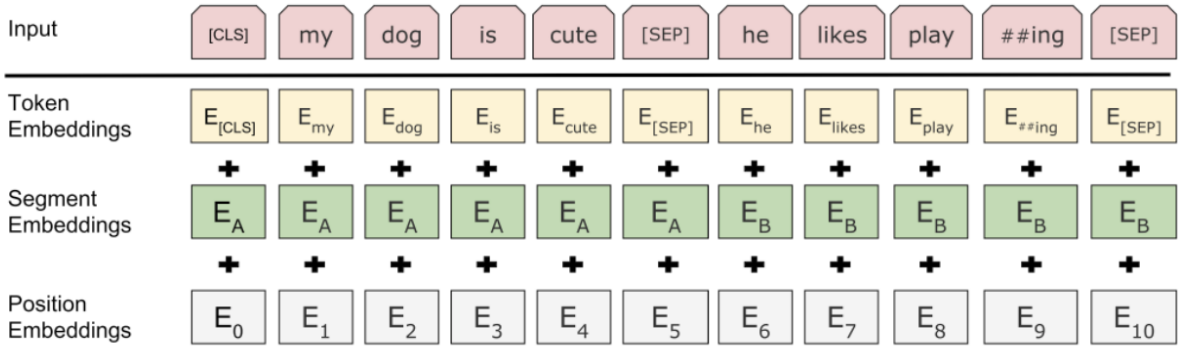


Fig-8: BERT input representation

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

- The Transformer model processes the complete input sequence.
- 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.

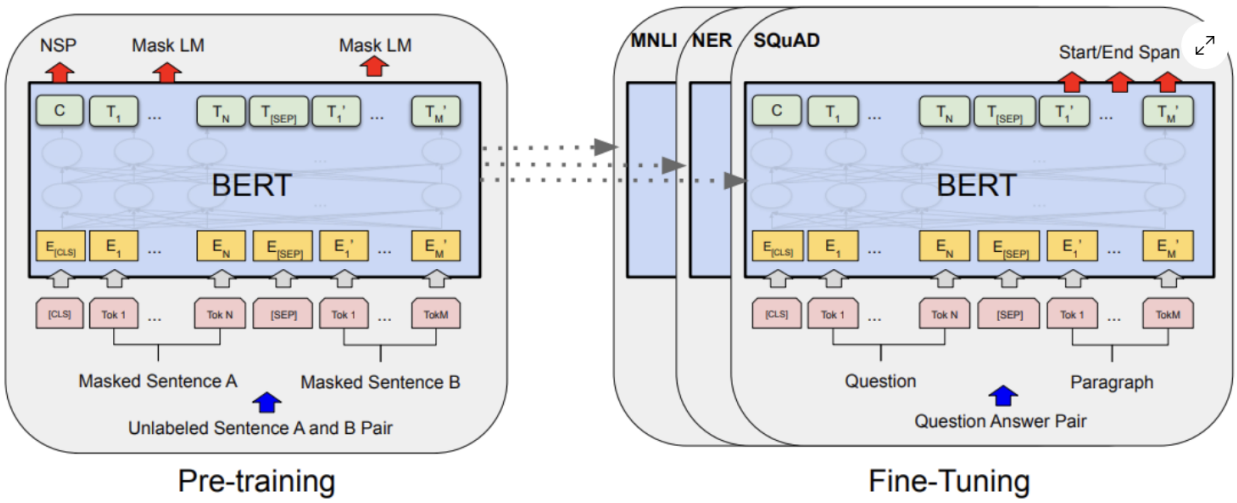


Fig-9: Fine-Tuning BERT

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.

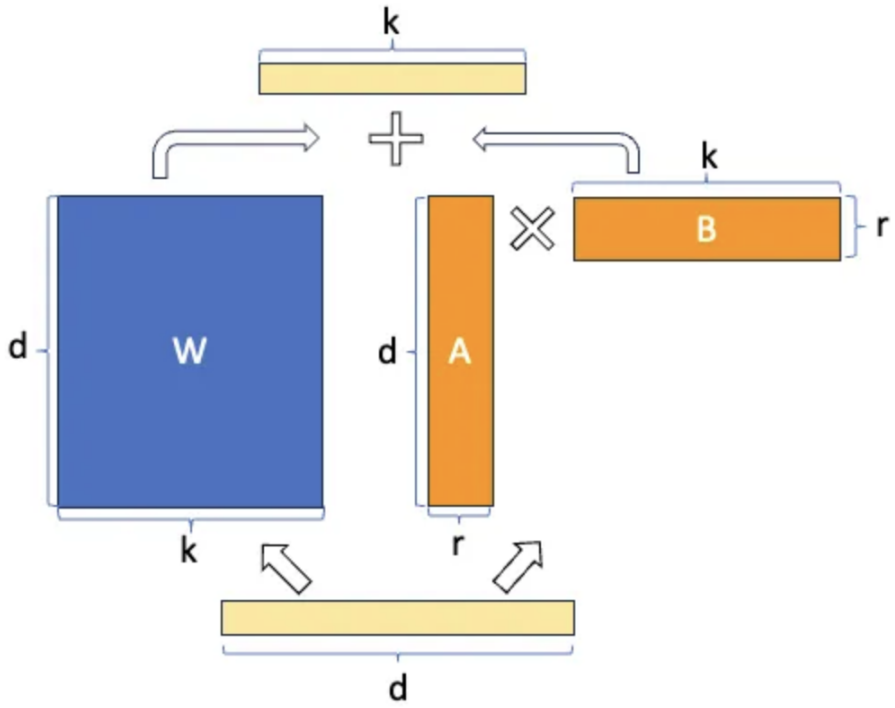


Fig-10: LORA

4. Experiment Setup

4.1 Data

In our research, we address the SQuAD 2.0 challenge, utilizing the designated datasets for training, development, and testing. Adhering to the distribution specified in the provided documentation, our dataset is partitioned into 130,319 training examples, 11,873 development examples, and 8,862 test examples.

Two preprocessing functions are coded to handle training and validation data. These functions tokenize the inputs, create sliding windows to manage sequence length, pad inputs to the maximum length, and calculate answers' start and end positions within the context. Special attention is paid to handling overflow tokens and offset mapping for precise answer localization.

4.2 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.

4.3 ELECTRA Model

The ELECTRA model is fine-tuned for the SQuAD 2.0 challenge in this experiment setup using a robust and structured approach based on HuggingFace's PyTorch implementation of BERT for question answering.

Key hyperparameters are specified, including the maximum sequence length of inputs, stride for tokenization, and the number of best predictions to generate like the following:

- The maximum sequence length: 384
- Stride: 128
- The number of predictions to generate: 20
- The maximum length of answer: 30

The tokenizer is initialized with the ELECTRA base discriminator checkpoint from Google, and training parameters like learning rate, batch size, and number of epochs are set.

- Learning rate: 5e-5
- Batch size: 32
- Number of epochs: 3 (It took one and a half hours to train.)

The ELECTRA model is initialized for question answering, with an AdamW optimizer set with the defined learning rate. A training function orchestrates the training loop, leveraging the Accelerator for efficient multi-device training, and incorporates evaluation steps to monitor model performance using the F1 and Exact Match metrics. The losses, including negative log-likelihood, are tracked to assess the model's fit to the data and detect overfitting.

A fine-tuning step utilizes Hugging Face's Trainer API, encapsulating the training process within a streamlined API call that includes model saving. A prediction function is established to run inference on new questions and contexts, utilizing the trained model to generate answers.

4.4 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 fits appropriately with the data and identifies 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 inference on new questions and contexts, employing the fine-tuned model to generate answers accurately.

5. Results

The results of our project are instrumental in understanding the capabilities of the ELECTRA and BERT with LoRA models in the context of the SQuAD 2.0 dataset. The primary focus was on evaluating the performance of these models in terms of Exact Match (EM) and F1 scores on the development set.

Models	Exact Match - Dev	F1 Score - Dev
ELECTRA (Fine-tuned)	50.07	50.07
BERT with LORA	50.98	50.98

Table-[n]:

5.1 ELECTRA Model Performance

The ELECTRA model, fine-tuned for the SQuAD 2.0 challenge, showcased promising results. In terms of the Exact Match (EM) score, the model achieved a score of 50.07, indicating that in 50.07% of the cases, the model's answer matched the human answer. Similarly, the model's F1 score, which measures the average overlap between the predicted answers and the ground truth, was also 50.07. This indicates a balanced performance in terms of both precision and recall.

5.2 BERT with LoRA Model Performance

The BERT model enhanced with LoRA (Low-Rank Adaptation) technology, tailored for the SQuAD 2.0 dataset, showed a slightly better performance than the ELECTRA model. The Exact Match score for the BERT with the LoRA model was 50.98, demonstrating a marginally higher accuracy in perfectly matching the human answers. Correspondingly, the F1 score for this model was also 50.98, illustrating its effectiveness in finding relevant information in the context paragraphs.

5.3 Comparative Analysis

Upon comparing both models, the BERT with LoRA configuration exhibited a slight edge over the ELECTRA model. While the difference in performance metrics was not substantial, it indicates the potential benefits of using LoRA for fine-tuning BERT models in complex question-answering tasks like those presented in SQuAD 2.0. The enhanced performance of BERT with LoRA could be attributed to its ability to adapt large language models more efficiently, focusing on the most relevant parameters for the task.

5.4 Discussion

The results achieved in this project underscore the challenges inherent in the SQuAD 2.0 dataset, particularly given the inclusion of unanswerable questions. Both models demonstrated considerable prowess in navigating these complexities, though there remains a notable gap compared to human performance. This gap highlights the ongoing challenges in the field of NLP, particularly in developing models that can accurately interpret and respond to nuanced human language.

Further, the comparative results suggest that while both models are effective for the task, applying LoRA in fine-tuning BERT models offers a slight advantage. This could be explored in future work to understand how low-rank adaptations impact model performance across a broader range of NLP tasks.

5.5 Limitations and Future Work

While the results are promising, there are limitations to this study. The scope of the project was confined to two models and one dataset. Future work could explore applying these models to other datasets or compare them with advanced NLP models. Additionally, investigating the reasons behind the performance differences and exploring further optimizations in model architecture and training strategies could yield even better results.

6. Summary and Conclusions

In conclusion, this project on SQuAD 2.0 demonstrated the capabilities of ELECTRA and BERT with LoRA models in tackling complex question-answering tasks. Our findings revealed a slightly superior performance of BERT with LoRA over ELECTRA, highlighting the potential of fine-tuning techniques in enhancing model efficacy. The project underscores the challenges and opportunities in natural language processing, particularly in developing models that closely mimic human comprehension and reasoning. While there remains a gap between current model performance and human benchmarks, our study contributes to the evolving landscape of NLP, offering insights and directions for future advancements in the field.

References

[1] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training.

[2] Clark, K., Luong, M.-T., Brain, G., Le Google Brain, Q., & Manning, C. (2020). ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS. https://arxiv.org/pdf/2003.10555.pdf

[3] ELECTRA. (n.d.). Huggingface. co. https://huggingface.co/docs/transformers/model_doc/electra

[4] Elias, G. (2020, December 29). Tutorial: How to pre-train ELECTRA for Spanish from Scratch. Skim AI. https://skimai.com/how-to-train-electra-language-model-for-spanish/

[5] Hu, Edward J., et al. “Lora: Low-rank adaptation of large language models.” arXiv preprint arXiv:2106.09685 (2021).

[6] Jurafsky, D., & Martin, J. H. (2023). Question answering and information retrieval. In Speech and Language Processing (3rd ed. draft). [Jan 7, 2023 draft]. https://web.stanford.edu/~jurafsky/slp3/

[7] Morzkowski, K., Sun, L., & Hoovestol, P. (n.d.). SQuAD 2.0 Project Report. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15785042 pdf

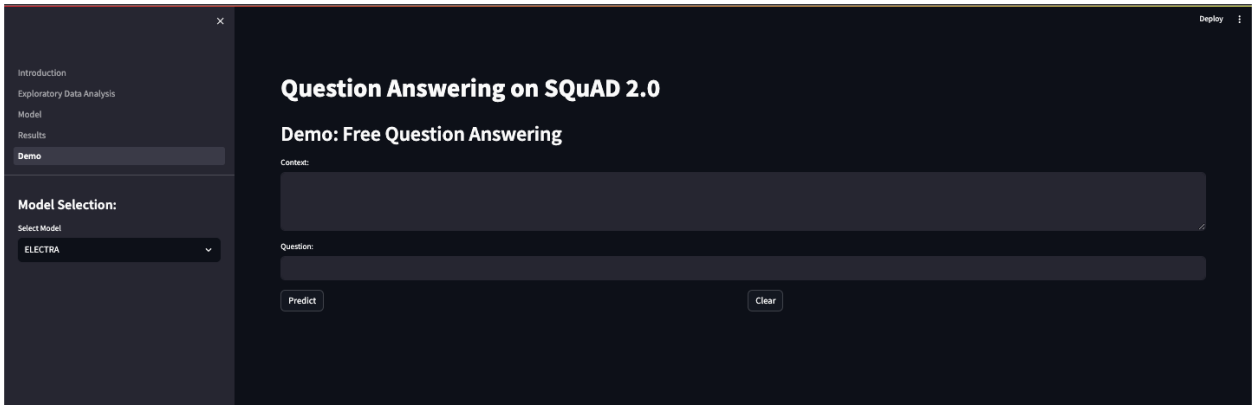
[8] Rajpurkar, P., Jia, R., & Liang, P. (n.d.). Know What You Don’t Know: Unanswerable Questions for SQuAD. https://arxiv.org/pdf/1806.03822.pdf

[9] Question answering. (n.d.). Huggingface. co. https://huggingface.co/docs/transformers/tasks/question_answering

[10] Transformers.modeling_electra — transformers 3.0.2 documentation. (n.d.). Huggingface. co. Retrieved December 10, 2023, from https://huggingface.co/transformers/v3.0.2/_modules/transformers/modeling_electra.html #ElectraForQuestionAnswering

[11] What is Question Answering? - Hugging Face. (2001, September 26). https://huggingface.co/tasks/question-answering

Appendix



Demo: The UI of the Question Answering Application on our trained models with Streamlit.