RUTGERS
UNIVERSITY | NEW BRUNSWICK

Data Science

Special Topic Data Science Project - Fall 2023

Question Answering System

Team Number 12

Sparsh Jain (sj983),

Mayank Shah (ms3462),

Aalap Kishor Patil (akp177),

Rushabh Karnavat (rpk68)

Contents

1 Introduction **3**

2 Literature Review **3**

3 Methodology **4**

3.1 Data Preprocessing 4

3.2 BERT QA Model 5

3.3 Model From Scratch 6

3.4 Evaluation 7

4 Experiments **7**

5 Results **8**

6 Analysis **9**

7 Challenges **10**

8 Conclusion **11**

9 Team Contribution **11**

10 References **11**

1 Introduction

SQuAD 2.0 stands for the Stanford Questions and answering dataset, which contains reading comprehension. Squad 2.0 contains questions with no answers and such questions tend to trick the model, which make this a challenging problem. Question answering in natural language processing involves designing systems or models that can understand and respond to user queries by extracting relevant information from text or knowledge bases, often utilizing techniques like information retrieval, knowledge-based approaches, semantic parsing, or machine learning. This project aims to build an advanced Reading Comprehension specific, Question Answering (QA) system by utilizing the BERT (Bidirectional Encoder Representations from Transformers) model.

The approach was to use the pre-trained model - BERTForQuestionAnswering, and then fine-tune it on the SQuAD 2.0 dataset and evaluate the model for results. Next, the project also focused on modifying the BERT architecture for any downstream task, in this case, question-answering. Finally, various experiments were performed to select the parameters for the models for training. In the end, the evaluation results for both the fine-tuned and the model built from scratch were compared, and also performed some interesting analysis on them.

2 Literature Review

"Know What You Don't Know: Unanswerable Questions for SQuAD," authored by Rajpurkar et al. in (2018) [1], discusses the inclusion of unanswerable questions in SQuAD 2.0, as proposed and represents a substantial advancement in the area of Question Answering (QA) research, as it effectively tackles the significant drawbacks of current datasets. SQuAD 2.0 diverges from typical QA datasets, such as SQuAD 1.1, by not only requiring models to extract answers from questions but also to evaluate their plausibility and recognize situations when no solution can be found within the given context. This is consistent with research that supports the use of QA systems that go beyond basic information retrieval and include the ability to engage in critical reasoning.

Nevertheless, the utilization of human-written "adversarial" questions in SQuAD 2.0 brings about both advantages and difficulties. The realistic and complex nature of these questions, in contrast to machine created instances, renders them a more rigorous test of NLU models. However, the presence of subjectivity and potential bias in questions created by humans raises concerns regarding the evaluation of models and the absence of a definitive "correct" answer in specific situations. In order to fully use SQuAD 2.0's potential to help make responsible and reliable NLU models, these problems must be solved. One way to do this is to look into bias in depth and think about other ways to give answers, like confidence scores or reasons for not answering.

By leveraging prior research on question answering (QA) and recognizing the possible drawbacks of subjectivity, SQuAD 2.0 offers a hopeful direction for the progress of natural language understanding (NLU). In order to ensure that models trained on this dataset make a valuable contribution to the development of responsible and trustworthy machine learning models, it is crucial to conduct additional research that specifically addresses bias reduction and the exploration of larger question types.

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," authored by Devlin et al. in 2018 [2], brought about a significant transformation in the domain of Natural Language Processing (NLP) with the introduction of a pre-training approach centered around Transformers. Previous NLP models were usually trained on language models that processed text in a sequential manner, either from left to right or from right to left. This approach restricted their ability to comprehend the links and dependencies between words. The revolutionary masked language modeling objective of BERT utilizes both contextual aspects

simultaneously, similar to how humans read and understand language. By employing a bidirectional strategy, as depicted in the visualization below, BERT is able to effectively capture more profound semantic linkages, resulting in exceptional performance across many natural language processing (NLP) applications.

BERT has demonstrated exceptional performance on various NLP benchmarks, such as question answering, sentiment analysis, and natural language inference, beating the previous best results by a wide margin.In contrast to architectures that are designed for specific tasks and require extensive modifications for each new application, BERT may be easily adjusted with minor changes for different tasks, resulting in a large reduction in the required training data and making it very flexible. In contrast to more complex NLP models, BERT's structure is comparatively uncomplicated and more accessible for downstreaming for many applications.

The process of pre-training and fine-tuning BERT requires substantial computer resources, which may restrict its availability Data Bias: The extensive pre-training dataset utilized for BERT may have inherent biases that can manifest in subsequent tasks. To address these biases, it is necessary to carefully choose the training data and create effective methods for identifying and rectifying biases. Despite BERT's outstanding performance, understanding its internal reasoning process and the methodology behind its precise predictions remains a challenging task.

3 Methodology

3.1 Data Preprocessing

This project will use the SQuAD 2.0 dataset. The dataset was obtained from Github repository[3]. The data is retrieved by using three JSON files for train, validation, and test, which contain 130319, 13220, and 5915 data points, respectively. These files were loaded and segmented into three lists - texts, queries and answers. The "texts" list contained the Passage, the "queries" list contained the Questions and the "answers" list contained a dictionary with the 'Answer_text' and 'Answer_start' as keys. The same procedure was followed for the questions with no answers, where the "answers" list was appended with 'Answer_text' = '' and 'Answer_start' = -1. Samples from the dataset are shown in Fig. 1 and 2.

```
Passage:  Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ bee-YON-say) (born September 4, 1981) is an American singer,
songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and da
ncing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child.
Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Th
eir hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo ar
tist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and
"Baby Boy".

Query:  When did Beyonce start becoming popular?

Answer:  {'text': 'in the late 1990s', 'answer_start': 269}
```

Figure 1: Data point containing answer taken from Database

```
Passage:  The Legend of Zelda: Twilight Princess (Japanese: ゼルダの伝説 トワイライトプリンセス, Hepburn: Zeruda no Dense
tsu: Towairaito Purinsesu?) is an action-adventure game developed and published by Nintendo for the GameCube and Wi
i home video game consoles. It is the thirteenth installment in the The Legend of Zelda series. Originally planned
for release on the GameCube in November 2005, Twilight Princess was delayed by Nintendo to allow its developers to
refine the game, add more content, and port it to the Wii. The Wii version was released alongside the console in No
rth America in November 2006, and in Japan, Europe, and Australia the following month. The GameCube version was rel
eased worldwide in December 2006.[b]

Query:  What consoles can be used to play Australia Twilight?

Answer:  {'text': '', 'answer_start': -1}
```

Figure 2: Data point without answer taken from Database

Next, the calculation of the end position character was done by adding the length of the 'Answer_text' to the value of 'Answer_start' from the "answers" list. In the following step, the "queries" and "texts" lists were tokenized by employing the BERT-based AutoTokenizer to get the input encodings. Padding was added to the final encodings as shown in Fig. 3, which were used to equalize the sequence length of 512 inputs. So, if the sequence length exceeded 512, the inputs were truncated.

```
[CLS] what was the horizontal average slip on the guanxian - anxian fault? [SEP] the longmen shan fault system is s
ituated in the eastern border of the tibetan plateau and contains several faults. this earthquake ruptured at least
two imbricate structures in longmen shan fault system, i. e. the beichuan fault and the guanxian – anxian fault. in
the epicentral area, the average slip in beichuan fault was about 3. 5 metres ( 11 ft ) vertical, 3. 5 metres ( 11
ft ) horizontal - parallel to the fault, and 4. 8 metres ( 16 ft ) horizontal - perpendicular to the fault. in the
area about 30 kilometres ( 19 mi ) northeast of the epicenter, the surface slip on beichuan fault was almost purely
dextral strike - slip up to about 3 metres ( 9. 8 ft ), while the average slip in guanxian – anxian fault was about
2 metres ( 6 ft 7 in ) vertical and 2. 3 metres ( 7 ft 7 in ) horizontal. [SEP] [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]
```

Figure 3: Data point taken from Database after preprocessing

Finally, the start and end positions were converted into start and end tokens, which helped in calculating the "span" of the answer from the Passage. For no answers, the start and end tokens were both set to 0, that is, both pointing at the [CLS] token.

3.2 BERT QA Model

For the scope of this project, BERTForQuestionAnswering was chosen as the baseline model. This is a pre-trained model that is available on Hugging Face[4]. It has a span classification head on top for extractive question-answering tasks. This acted as the baseline because this model is specifically built for the question and answer task, and this would serve as a good benchmark for future model evaluation comparisons. This model uses a BERT base architecture with a custom layer added on top, specifically designed for question and answering tasks. The model takes the input encodings and passes them into the BERT architecture to get a contextualized output. Using the span classification head, this contextualized output is used to predict the start and end probabilities of each input sequence. Based on these probabilities, the most likely span can be calculated.

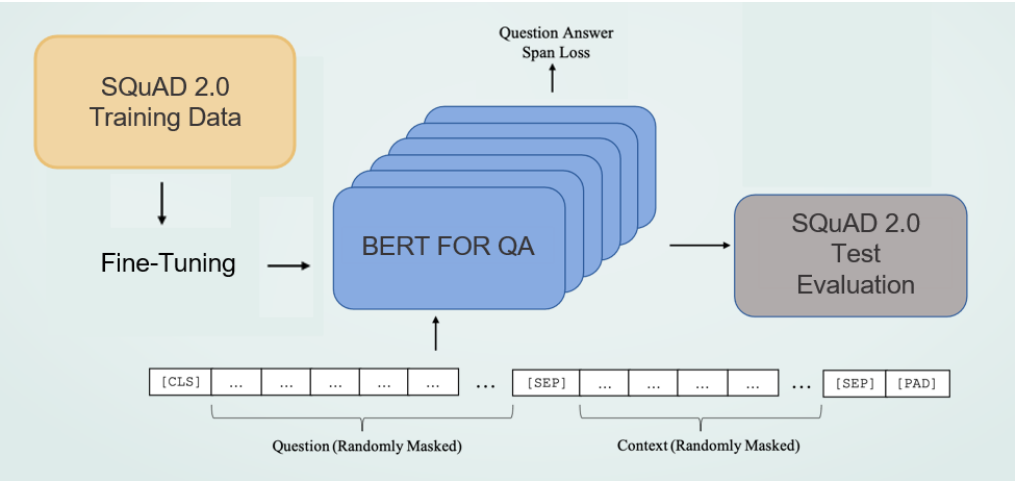


Figure 4: Fine Tuning BERT For QA

An evaluation of this model showed that, although it performed well for questions with answers, it performed extremely poorly for incorrect questions without answers. So to evaluate this model's performance, it was fine-tuned by updating the weights on the Squad 2.0 dataset as shown in Fig 4., and its results were evaluated.

3.3 Model From Scratch

In order to explore the language model's other potential, this project also aimed to take the basic LLM architecture and adapt it to any NLP activity. For this purpose, the BERT model for the question-answering task as shown in Fig. 5 was taken. This project advanced by selecting the BERT base uncased with 100 million parameters. Considering the fact that the baseline model was BERTforQuestionAnswering and also taking into account the limited amount of computation power, the BERT base uncased came out to be the best fit for the foundation of our scratch model. Referring to the research papers, we learned that BERT can be downstream to any task-specific model by adding a task-specific head to it. Since it already contained the transformer layers in its architecture, this project does not take into consideration for adding another transformer layer to its architecture. The final hidden state output layer from the BERT containing 768 nodes is followed by the Question-Answering head.

The initial layer of the Question-Answering head contains the Linear layer containing the 512 nodes. This layer is followed by an activation layer, ReLU. This model was then experimented with and without a dropout layer. This experiment concluded the presence of the dropout layer with a probability of 0.2 because it seems that the model without the drop-out layer overfitted the data. The dropout layer was then followed by the final output layer which contained 2 nodes. The two nodes represent the start and end tokens. This layer is then followed by the SoftMax activation layer, giving out the probabilities for each input token. Later on, the start and end indexes were taken out by calculating the maximum probability from each node.

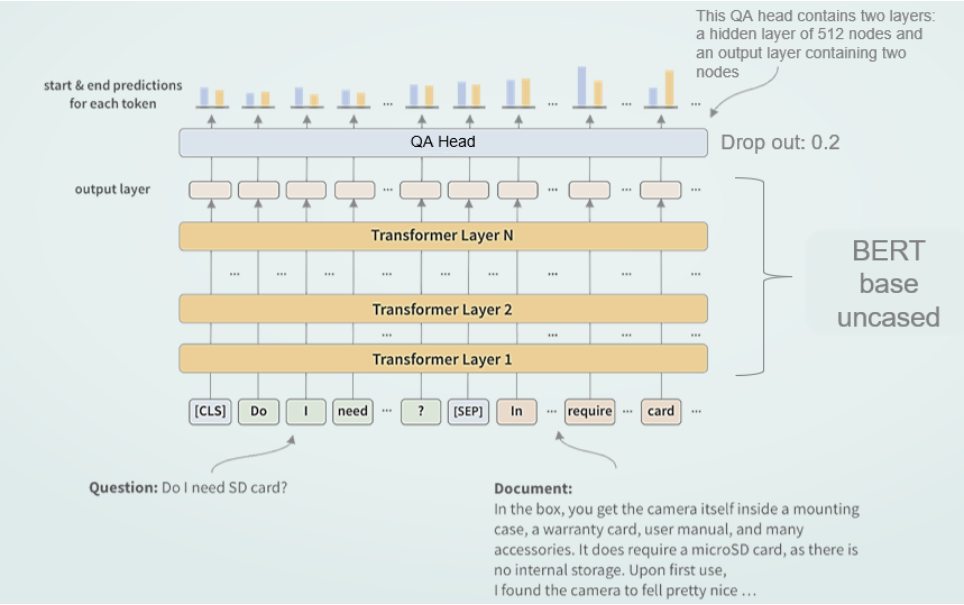


Figure 5: An architecture of task-specific BERT

This model has its own defined loss function. The loss function mainly takes care of the wrong prediction error for the start and end indexes. It is the sum of the losses for the start index and the end index, calculated separately. The loss function minimizes the negative of the log of the probabilities of the correct start and end index predicted by the model. The mathematical expression for the function is given below. Then, lastly, after calculating the loss function stated above, the model returns the start and end probabilities for each input token, the attention mask, and the output of the last hidden state.

$$\text{Loss} = -\log P_{start} - \log P_{end}$$

3.4 Evaluation

The pre-trained model and the model built from scratch both return start and end logits as outputs. The start and end logits contain the probabilities of each input sequence being the start and end of the answer span. A score of each candidate span is calculated using the formula below,

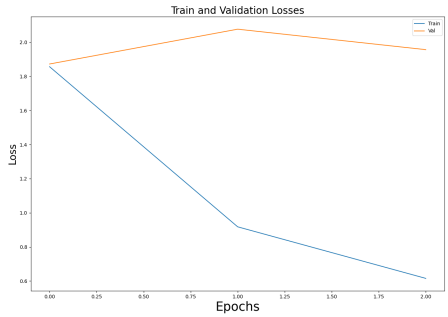
$$\text{Span} = P_{starti} + P_{endj}$$

Where P_{starti} is the probability of the start token and P_{endj} is the end token for every i, j in range (0, length of input), where j >= i. The candidate span with the highest value is selected as the predicted span. In the event that the prediction span returns just the [CLS] token, the prediction is set as a blank string (“ ”). The evaluation was performed on the test data with 2,000 examples without an answer and 2,000 examples that have an answer. The metrics for evaluation were Exact Match (EM) and F1-score.

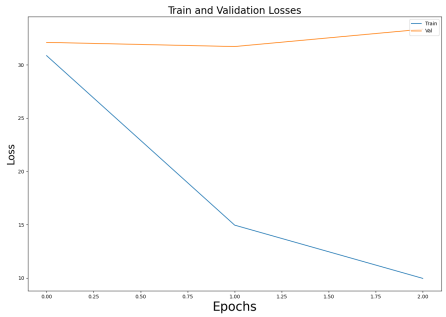
4 Experiments

In this project, various experiments were performed during training the model to select the learning rate, batch size and number of epochs. Training of the model for 4 epochs exhausted the GPU resources of google colab before the completion of the training, thus all further training was done with 3 epochs. Gradually increasing the batch size in accordance with the computational resources at hand enabled the acceleration of convergence while maintaining accuracy. Selecting the batch size of more than 12 also became extremely computationally expensive, thus the training was done with a batch size of 8.

In order to optimize hyperparameters, preliminary training trials were performed using a reduced dataset consisting of 10,000 data points and a validation set of 5000 data points. The optimization of the learning rate utilized a comparable iterative methodology. All the learning rates were investigated, starting with a 5e-6. This learning rate caused minor overfitting, as indicated by the training loss exceeding the validation loss after only three epochs in Fig 6. This suggested that the slow learning rate resulted in inadequate training time. In contrast, increased learning rates resulted in accelerated convergence, albeit at the potential expense of precision. In the end, we determined that a learning rate 5e-5 struck the ideal balance between performance and training efficiency. Subsequently, this configuration was implemented during training using the complete dataset.

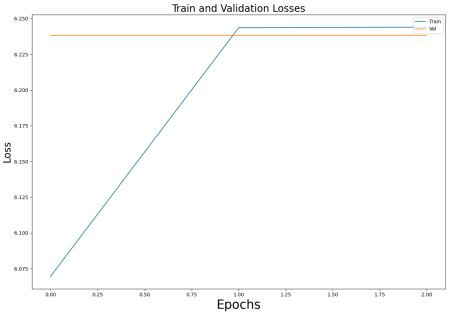


(a)

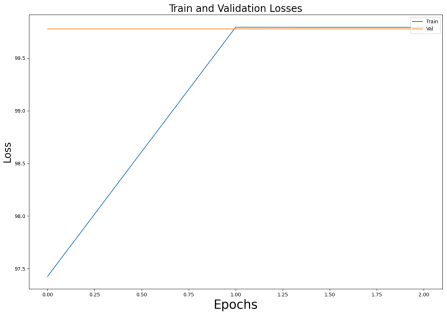


(b)

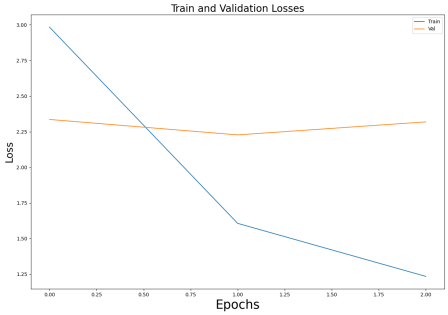
Figure 6: (a) Loss Vs Epochs of Fine Tuned Model for Learning Rate 5e-5 (b) Loss Vs Epochs of Scratch Model for Learning Rate 5e-5



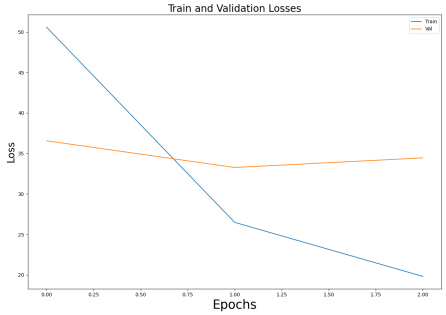
(a)



(b)



(c)

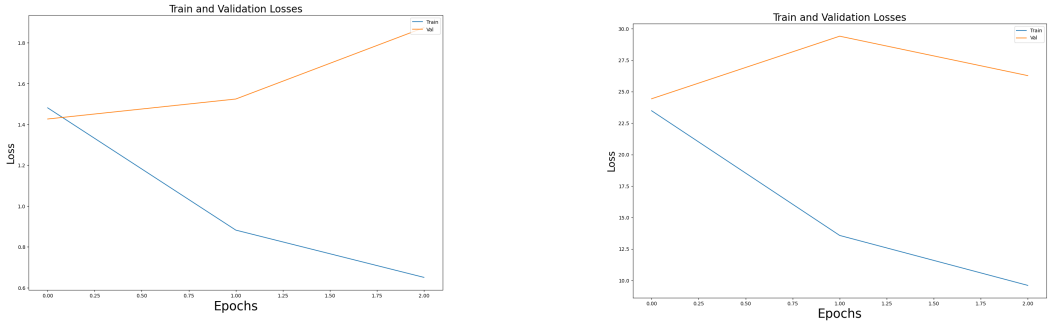


(d)

Figure 7: (a) Loss Vs Epochs of Fine Tuned Model for Learning Rate 5e-4 (b) Loss Vs Epochs of Scratch Model for Learning Rate 5e-4 (c) Loss Vs Epochs of Fine Tuned Model for Learning Rate 5e-6 (d) Loss Vs Epochs of Scratch Model for Learning Rate 5e-6

5 Results

After experimentation, both the fine-tuned and the model built from scratch were trained for a total of 3 epochs with 40,000 data points. Adam optimizer was used and batch size was set to 8. The training loss for the fine-tuned model was obtained to be 0.7 and the validation loss was 1.7 as shown in Fig 8(a). Further, a training loss of 9.6 and a validation loss of 26.7 were obtained on the model trained from scratch as seen in Fig 8(b).



(a) (b)

Figure 8: (a) Loss Vs Epochs of Fine Tuned Model for Learning Rate 5e-5 (b) Loss Vs Epochs of Scratch Model for Learning Rate 5e-5

The final test set was formed by combining 2000 question-answer samples from test data that had no answers and 2000 samples that had answers. All the model performances were evaluated using the f1-score and the exact match score. Using the pre-trained BERT For QA model, it was observed that it performed well for the queries that had answers in the context but could not recognize the queries without answers and gave a F1-score and EM of 0.

After fine-tuning the pre-trained model and training it with 40,000 samples for 3 epochs, the performance of the model for the queries with no answers improved significantly. Although the model performance for questions with answers reduced slightly, the overall performance of the model improved.

The final model, which was built by adding a custom question-answering head on top of the pre-trained Bert model and training it for 3 epochs, gave a similar performance as the fine-tuned model for queries with no answers but performed better for queries with answers, as seen from the EM score and F1-score in the Fig. 9.

Model	**40,000 training, 4000 Test (2000 withAns, 2000 withoutAns)**					
	With Answer		Without Answer		Total	
	Exact match	F1	Exact match	F1	Exact match	F1
BertQA	0.56	0.74	0.00	0.00	0.28	0.37
BertQAFineTuned	0.31	0.43	0.37	0.37	0.34	0.40
BertQAScratch	**0.36**	**0.47**	**0.37**	**0.37**	**0.36**	**0.42**

Figure 9: Comparison among all the models

Overall, the model, which was built by adding a custom question-answering head, performed slightly better than the fine-tuned model, which can be seen in the Fig. 9.

6 Analysis

The predictions obtained from the model were further analyzed for structured and unstructured types of questions. For structured questions, the analysis was done for "WH-type" of questions like Why, Who, Where, etc. It was found that the model performed significantly better for "WH-type" questions than for unstructured questions, as seen from the F1-Score and Exact Match scores in the Fig. 10.

Type of question ?	Exact Match	F1- Score
When ?	0.71	0.74
Who ?	0.75	0.8
How ?	0.66	0.72
Why ?	0.48	0.58
Where ?	0.52	0.64
What ?	0.66	0.72
None of above	0.18	0.24

Figure 10: Analysis of "WH-" type questions

Further, to broaden the analysis on whether the model performed better for Short answer questions or Large answer questions, only the data that had answers was selected for this purpose. The test data with answers consisted of 1561 Short answer questions and 439 Long answer questions. As expected, the model performed significantly better for queries that had answers of less than or equal to 5 words than for queries that had long answers, as seen from the Fig. 11.

For queries with answers			
Length of Answer	Number of sentences	Exact match	F1- Score
Less than 5	1561	0.43	0.52
Greater than or equal to 5	439	0.08	0.3

Figure 11: Analysis based on Length

The reason for lower performance for Long answer questions in comparison to that of short answer questions is due to the lesser number of training examples (3,290 out of 40,000) for Long answer questions in the training data.

7 Challenges

In order to optimize a pre-existing model, such as BERT, for a specific purpose, it was crucial to carefully evaluate the hyperparameters, training data, and optimization approaches. Attaining the ideal balance posed a challenge. Noise and inconsistencies present in the SQuAD 2.0 dataset such as the test set containing only examples with no answers, affected the evaluation of the model. The process of training and fine-tuning large transformer models, like BERT, required a significant amount of time and computer resources. Adequate computational resources are essential for the efficient creation of models. Limitations in memory and storage had a negative impact on performance when the size and complexity of the model, as well as the amount of training data used, were restricted.

8 Conclusion

In conclusion, the project aimed to develop an advanced Question Answering (QA) system for reading comprehension, utilizing the SQuAD 2.0 dataset and the BERT model. The team achieved notable improvements by fine-tuning the pre-trained BERTForQuestionAnswering model, addressing the challenges posed by questions without answers in the dataset. Through systematic experiments, valuable insights were gained into optimizing model parameters, underscoring the nuanced balance required for optimal performance.

The comparison between the fine-tuned model and a model constructed from scratch underscored the benefits of leveraging pre-trained models for specific tasks. Moreover, By adding a task-specific head to the last hidden layer of BERT, it can be down streamed to any task-specific model.

The BERTForQA model, which was already trained, yielded highly satisfactory results when applied to questions that had corresponding answers. In the future, it is possible to add a classification layer in between the model which will determine whether the question can be answered or not. If an answer is available, the model will extract it from the context. Otherwise, the model will not predict any answer. This model can further be enhanced by implementing an ensemble architecture

9 Team Contribution

Sparsh and Mayank were responsible for making a project plan and architecture. Data Loading and exploring was handled by Rushabh, Sparsh was incharge of data pre-processing, Rushabh and Mayank were involved in loading a pre-trained model and fine tuning it. Aalap ran the required experiments on the pre-trained model. Aalap and Sparsh created a BERT based Question and Answering model from scratch. Mayank ran the required experiments on this model. Aalap and Rushabh were tasked with evaluation of the models and analysing the results. Report and documentation were done by all team members.

10 References

[1] Rajpurkar, Pranav, Robin Jia, and Percy Liang. "Know what you don't know: Unanswerable questions for SQuAD." arXiv preprint arXiv:1806.03822 (2018).
[2] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
[3] `https://github.com/chrischute/squad/tree/master/data`
[4] `https://huggingface.co/transformers/v3.0.2/model_doc/bert.html#bertforquestionanswering`