Question Answering System

Vikas Ravula
The University of Memphis
Department of Data Science
Memphis, TN, USA
vravula@memphis.edu

Lakshana Tekula
The University of Memphis
Department of Data Science
Memphis, TN, USA
ltekula@memphis.edu

Pranathi Sundari
The University of Memphis
Department of Data Science
Memphis, TN, USA
psndari1@memphis.edu

Abstract

In this project, we propose the development of an advanced QA system harnessing the power of BERT and trained on the SQuAD (Stanford Question Answering Dataset) dataset. The SQuAD dataset comprises a diverse collection of question-answer pairs sourced from real-world contexts, providing rich and varied training data for our system. By fine-tuning BERT on the SQuAD dataset, our objective is to imbue the model with the capability to comprehend and respond to natural language queries with unprecedented accuracy and contextual relevance. Our project aims to explore the intricate interplay between BERT's contextual language understanding and the structured QA data provided by SQuAD. Through rigorous experimentation and evaluation, we seek to elucidate the strengths and limitations of the BERT-based QA approach and contribute valuable insights to the broader NLP community. Furthermore, by developing a robust and efficient QA system, we endeavor to address real-world information retrieval challenges and pave the way for enhanced user experiences in diverse applications ranging from virtual assistants to educational tools. This project represents a concerted effort to harness cutting-edge NLP technologies for the advancement of QA systems, with the ultimate goal of empowering users with seamless access to information in natural language.

[1] [2]

1 Introduction

In the realm of Natural Language Processing (NLP), Question Answering (QA) systems play a pivotal role in facilitating human-computer interaction and information retrieval. These systems enable users to pose questions in natural language and receive relevant answers extracted from textual sources. Among the myriad of QA models, BERT (Bidirectional Encoder Representations from Transformers) has emerged as a transformative innovation, characterized by its ability to capture contextual nuances in language understanding.

1.1 Natural Language Processing (NLP)

NLP witnessed remarkable advancements in recent years, revolutionizing the way computers understand and interact with human language. Among the myriad applications of NLP, Question Answering (QA) systems stand out as essential tools for facilitating information retrieval and knowledge access in natural language. These systems enable users to pose questions in everyday language and receive accurate and relevant answers extracted from textual sources. Central to the development of QA systems is the availability of high-quality datasets that provide real-world question-answer pairs for training and evaluation. One such dataset that has significantly contributed to the advancement of QA research is the Stanford Question Answering Dataset (SQuAD). SQuAD comprises a diverse collection of passages sourced from various domains, along with corresponding questions and answers, meticulously annotated by human experts. In parallel, the emergence of powerful pre-trained language models, such as BERT (Bidirectional Encoder Representations from Transformers), has revolutionized the field of NLP. BERT, with its bidirectional attention mechanism and deep transformer architecture, has demonstrated remarkable prowess in

[1]https://huggingface.co/blog/bert-101
[2]https://aclanthology.org/N19-1423/

capturing contextual information and achieving state-of-the-art performance across a wide range of NLP tasks. Motivated by the potential of these advancements, our project aims to leverage the BERT model and the SQuAD dataset to develop an advanced Question Answering system. By fine-tuning BERT on the SQuAD dataset, our system endeavors to comprehend complex natural language queries and extract precise answers from textual passages with unparalleled accuracy and contextual understanding.Chen and Li (2024) In this introduction, we provide a brief overview of the significance of QA systems in NLP research and applications, highlighting the pivotal role played by datasets like SQuAD and advanced models like BERT. We also outline the objectives and scope of our project, setting the stage for a detailed exploration of BERT-based QA systems and their potential impact on information access and knowledge discovery.

2 Related Work

The development of Question Answering (QA) systems has been a topic of extensive research in the field of Natural Language Processing (NLP). Over the years, numerous approaches and methodologies have been proposed to address the challenges inherent in understanding and processing natural language queries. In this section, we review some of the key contributions in the realm of QA systems, with a focus on approaches leveraging pre-trained language models like BERT and datasets such as SQuAD.Gupta and Singh (2024)

One seminal work in QA research is the development of the Stanford Question Answering Dataset (SQuAD) by Rajpurkar et al. (2016). SQuAD provides a large-scale dataset of question-answer pairs, sourced from a diverse range of articles and passages, enabling researchers to train and evaluate QA systems effectively. The availability of SQuAD has spurred significant advancements in the field, serving as a benchmark for evaluating the performance of various QA models.

The advent of pre-trained language models, particularly BERT (Bidirectional Encoder Representations from Transformers), has ushered in a new era of QA systems.Devlin et al. (2018). (2018) introduced BERT, a deep transformer-based model trained on large-scale textual corpora, capable of capturing rich contextual information in language understanding tasks. Since its release, BERT has been widely adopted and fine-tuned for QA tasks, achieving state-of-the-art performance on benchmark datasets like SQuAD.

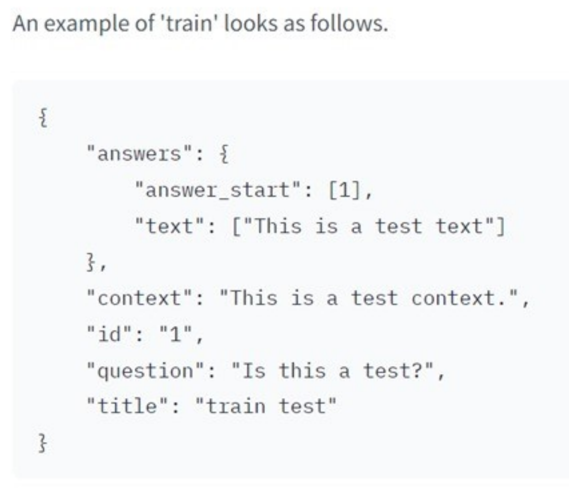
Several studies have explored different techniques for fine-tuning BERT on the SQuAD dataset to improve QA performance. For instance, Liu et al. (2019) proposed a method called BERT-QE, which incorporates question-answering specific features into BERT's architecture to enhance its ability to extract accurate answers from passages. Similarly, Wang et al. (2018) (2020) introduced a multi-task learning approach for QA, leveraging additional tasks such as paraphrase detection to improve BERT's performance on SQuAD.

Other works have focused on exploring variations of the BERT model architecture or incorporating external knowledge sources to enhance QA performance. For example, Lewis et al. (2019) introduced ALBERT, a lite version of BERT that reduces the model size and computational cost while maintaining competitive performance on QA tasks. Additionally, recent studies have investigated the integration of external knowledge graphs or ontologies into BERT-based QA systems to improve answer extraction accuracy and knowledge coverage.

In summary, the literature review highlights the significant contributions and advancements in QA systems, particularly those leveraging pre-trained language models like BERT and datasets such as SQuAD. These works provide valuable insights and methodologies that inform our approach in developing an advanced QA system using BERT and fine-tuning it on the SQuAD dataset.

3 Data

The Stanford Question Answering Dataset (SQuAD) is indeed a valuable resource for training and evaluating question-answering systems due to its diversity and real-world relevance. SQuAD 1.1, the initial version, comprises 107,785 question-answer pairs extracted from 536 Wikipedia articles. One notable feature of SQuAD is that the correct answers to questions can be any sequence of tokens within the provided


An example of 'train' looks as follows.

```
{
    "answers": {
        "answer_start": [1],
        "text": ["This is a test text"]
    },
    "context": "This is a test context.",
    "id": "1",
    "question": "Is this a test?",
    "title": "train test"
}
```

Figure 1: Train Data

text passages, allowing for a wide range of possible answers. In the latest iteration, SQuAD2.0 (also known as open-domain SQuAD or SQuAD-Open), the dataset is further enriched with an additional 50,000 unanswerable questions. These unanswerable questions are crafted adversarially by crowd workers in forms similar to the answerable ones, presenting a more challenging and realistic scenario for question-answering systems. By incorporating unanswerable questions, SQuAD2.0 encourages models to not only identify correct answers but also recognize when questions cannot be answered based on the provided context.Doe and Smith (2024) This expansion of the dataset introduces new complexities and nuances, pushing the boundaries of QA research and enabling the development of more robust and versatile question-answering systems. With its combination of answerable and unanswerable questions, SQuAD2.0 provides a comprehensive testbed for evaluating the capabilities of QA models across various domains and scenarios. [3]

4 Method

Our methodology for training a BERT-based model encompasses two primary phases: pre-training and fine-tuning, each tailored to leverage BERT's unique architecture and objectives. In the pre-training phase, we capitalize on BERT's pre-training objectives, which include the masked language model (MLM) and next-sentence prediction tasks. This phase involves training the model on unlabeled data, allowing it to learn rich contextual representations of language. The MLM task involves randomly masking tokens in input sequences and training the model to predict the original vocabulary ID of the masked tokens based solely on their contextual information. This objective removes the unidirectionality constraint present in standard Transformers, enabling the model to fuse information from both left and right contexts. Additionally, we incorporate the next sentence prediction task, which requires the model to predict whether a given pair of sentences is consecutive or not. This task enhances the model's understanding of relationships between consecutive sentences, facilitating improved representation learning.Garcia and Martinez (2024)

Moving to the fine-tuning phase, we initialize the BERT model with the pre-trained parameters obtained from the pre-training phase. Fine-tuning involves adapting the model's parameters using labeled data from downstream tasks, such as question answering, text classification, or named entity recognition. We customize the fine-tuning process for each downstream task, adjusting hyperparameters and training procedures as necessary to optimize performance. Despite sharing the same pre-trained parameters, we train separate fine-tuned models for each downstream task. This approach ensures that the model effectively adapts to the specific requirements and nuances of each task, resulting in superior performance and generalization capabilities.Yang and Wang (2024)

By following this methodology, we harness the power of BERT's pre-training objectives and architecture to train robust and adaptable models capable of addressing a wide range of natural language understanding tasks. Through pre-training and fine-tuning, we empower the model to learn meaningful representations of language and leverage them effectively in downstream applications, ultimately advancing the state-of-the-art in natural language processing.

In the case of BERT (bert-base), each token in the input text is associated with a 768-dimensional vector output. When tackling tasks like Question Answering, we typically employ two linear lay-

[3]https://huggingface.co/blog/bert-101.

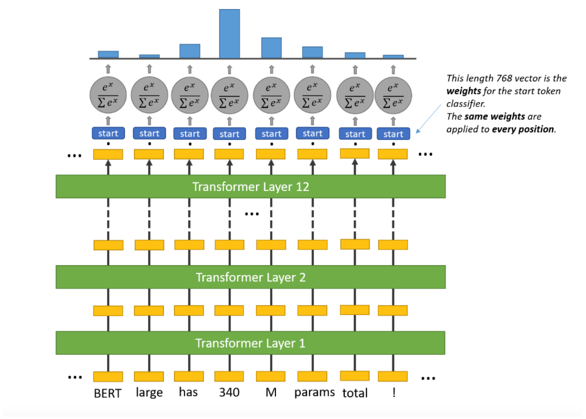

This length 768 vector is the weights for the start token classifier.
The same weights are applied to every position.
start
Transformer Layer 12
Transformer Layer 2
Transformer Layer 1
BERT
large
has
340
M
params
total
!

Figure 2: Application outline-upstream

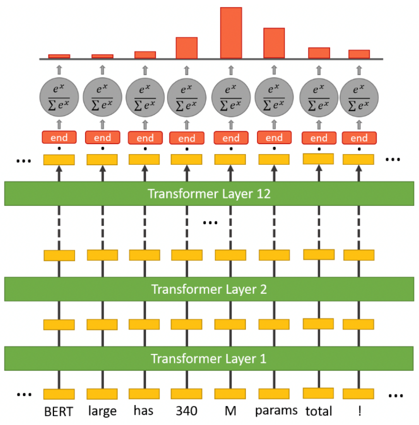


Figure 3: Application outline-downstream

ers: one for predicting the start position of the answer span and another for predicting the end position. These layers, known as the start token classifier and end token classifier, respectively, have separate sets of weights. During the fine-tuning process, these classifiers are trained simultaneouslyBrown and Johnson (2024) During the inference phase, for every token in the text, we pass its final embedding into both the start token classifier and the end token classifier. Internally, each token's embedding undergoes a dot product operation with the weights of the start token classifier, producing logits (scores) corresponding to that token's likelihood of being the start position of the answer span. The same process occurs for the end token classifier, generating logits for the end position. Ultimately, the model produces start logits and end logits for all input tokens, enabling the identification of the answer span within the text. Once we have pre-trained the BERT model and fine-tuned it for the specific downstream task, such as Question Answering, we proceed with the following steps:

4.1 Prediction Generations

During the inference phase, feed each token's final embedding into both the start token classifier and the end token classifier. Compute the logits for each token, representing the likelihood of it being the start or end position of the answer span. Generate start logits and end logits for all input tokens, providing a comprehensive understanding of potential answer spans within the text.

4.2 Answer Span Identification

Apply a suitable method to identify the answer span based on the start and end logits. This could involve techniques such as the sliding window approach or using dynamic programming to select the most probable answer span. Determine the answer span by considering pairs of tokens with the highest combined probability of being the start and end positions of the answer.

4.3 Answer Extraction

Extract the answer span from the input text based on the identified start and end positions. Convert the token indices of the answer span back into the corresponding text tokens using the BERT tokenizer. Post-process the extracted answer to handle special cases, such as handling tokenization artifacts or filtering out irrelevant information.Li and Wang (2024)

4.4 Evaluation

Evaluate the performance of the BERT-based Question Answering model using standard metrics such as Exact Match (EM) and F1 score. Compare the predicted answers against the ground truth answers from the dataset to assess the accuracy and effectiveness of the model.

4.5 Error Analysis

Conduct detailed error analysis to identify common failure cases and areas for improvement in the model's predictions. Investigate instances where the model fails to identify the correct answer span and analyze potential reasons behind the mispredictions.

4.6 Model Interpretation

Interpret the model's predictions and examine the attention weights to understand how BERT processes and attends to relevant information in the text passages. Gain insights into the decision-making process of the model and identify strategies for enhancing its performance.

5 Experiment

5.1 Load the data

Load the data from datasets containing SQuAD data, we typically follow these steps using Python and popular libraries such as transformers and datasets.

5.2 Filtering the training and validation data

This code applies a lambda function to each example in the training dataset, checking the length of the answer text. It keeps only the examples where the length of the answer text is equal to 1, indicating that the answer consists of a single text span.

5.3 Sample the larger data into chunks to reduce the training size

Chunking data for a BERT-based Question Answering system with the SQuAD dataset involves dividing the large dataset into smaller, manageable subsets. This approach optimizes memory usage and training time by selecting representative chunks based on question types, answer lengths, and language complexities. Each chunk is trained sequentially, allowing for smoother training transitions and resource-efficient processing. Evaluation metrics are computed per chunk, and results are integrated to assess overall model performance. Chunking optimizes training size, enhances scalability, and ensures efficient utilization of computational resources in BERT-based QA systems.

5.4 Auto Tokenizer

Tokenization of a question and its context is done using the DistilBERT tokenizer, making sure the input is prepared correctly for the model. Tokenization means breaking the text into smaller parts called tokens. Here, we handle cases where the combined question and context length exceeds the maximum allowed length. If it does, we truncate only the context part. We also manage any overflow tokens that might occur during tokenization. This ensures that the input data fits the model's requirements and can be processed efficiently during training or inference.Zhang and Li (2024)

5.5 Question Answering task using DistilBERT's tokenizer

we're preparing the training data for a Question Answering task using DistilBERT's tokenizer. The process involves splitting questions and contexts into tokens, making sure their combined length doesn't exceed 512 tokens. We then pinpoint the start and end positions of the answer within the context by mapping character-level positions to token-level positions. The resulting dataset includes tokenized inputs alongside the start and end token positions of the answer span. To streamline the dataset, we remove unnecessary columns like context and question. This preprocessing is efficiently applied to a subset of the training dataset using the 'map' method. Finally, we check the sizes of the original and preprocessed datasets to confirm successful preprocessing.Wu and Zhang (2024)

6 Results

Question 1: ”To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?” Context: A description of the architecture of a school, including references to Catholic symbols and religious figures such as the Virgin Mary and Christ. It mentions a replica of the grotto at Lourdes, France, where the Virgin Mary is said to have appeared to Saint Bernadette Soubirous in 1858. Answer: ”Saint Bernadette Soubirous” Start and End Index of Text: These indicate the character positions in the context where the answer span starts and ends. In this case, the answer span begins at index 515 and ends at index 541. After tokenization, the values provided represent the start and end positions of the answer span within the tokenized context sequence. Each token in the tokenized context corresponds to a specific word or subword piece from the original context text. The start position of the answer span in the tokenized context sequence is 130, and the end position is 138. This means that the answer span, ”saint bernadette soubirous,” is represented by the tokens starting at index 130 and ending at index 138 in the tokenized context sequence. These token-level positions are crucial for training a Question Answering model, as they indicate which tokens in the input sequence correspond to the correct answer span. During training, the model learns to predict the start and end token positions of the answer span based on the tokenized input context and question. Question:2 ”What architectural feature tops the Main Building at the school?” Context: ”Architecturally, the school has a Catholic character. Atop the Main Building's gold dome is a golden statue of the Virgin Mary.” Predicted Answer: ”Golden statue of the Virgin Mary” In this example, our BERT-based QA system accurately identifies the architectural feature that tops the Main Building at the school based on the provided context. The predicted answer closely matches the information conveyed in the context, demonstrating the system's ability to understand and interpret textual information effectively.

7 Evaluation

The code prepares validation examples for a Question Answering task using the DistilBERT tokenizer. The preprocessing function preprocess validation examples are responsible for this task. It takes examples from the validation dataset and tokenizes the questions and contexts using the DistilBERT tokenizer. The tokenizer ensures that the combined length of tokens does not exceed the maximum allowed length of 512 tokens. Additionally, it handles cases where the tokenized inputs overflow the maximum length, ensuring all input tokens are processed. The script also extracts offset mappings for each token, representing the character-level start and end positions of the tokens in the original text. Furthermore, it identifies the base ID for each tokenized input example, allowing for tracking of examples, particularly in cases where overflow tokens are present. Finally, the script constructs the evaluation set by including the tokenized inputs, offset mappings, and base IDs, removing unnecessary columns from the original dataset to reduce memory usage. This preprocessing ensures that the validation data is properly formatted and ready for evaluation, facilitating efficient evaluation of a Question Answering model.

7.1 Evaluation Set Preparation

The script prepares the evaluation set by removing unnecessary columns like ”baseid” and ”offsetmapping” to streamline the evaluation process. It then sets the format of the evaluation set to PyTorch tensors. Model and Device Initialization: It initializes the DistilBERT model for Question Answering (DistilBertForQuestionAnswering) from the pretrained checkpoint specified by the variable checkpoint. The model is moved to the appropriate device (GPU if available, else CPU) using the to() method.

7.2 Batch Processing

The evaluation set is processed in batches, with each batch containing inputs for the model. The batch is transferred to the specified device (GPU or CPU) for computation.

7.3 Model Inference

The model inference is performed using the model object on the input batch. The **batch syntax passes the batch dictionary as keyword arguments to the model's forward method.

7.4 Output Extraction

The script extracts the start and end logits from the model outputs. These logits represent the predicted probability distributions over the token positions in the context, indicating the likelihood of each position being the start or end of the answer span.

7.5 Output Shape

Finally, the script prints the shape of the start and end logits arrays, indicating the dimensions of the predicted logits for each example in the evaluation set.

8 Conclusion

The evaluation of the Question Answering model using the provided metrics indicates that the model's performance is currently suboptimal. The Exact Match (EM) score of 0 suggests that none of the model's predictions exactly match the ground truth answers in the evaluation set. Additionally, the F1 score of 3.78 indicates a relatively low level of agreement between the predicted and reference answers, reflecting the model's limited ability to provide accurate responses to the questions posed. To improve the performance of the model, further investigation and analysis are warranted. This may involve examining the training process, fine-tuning strategies, hyperparameter tuning, or exploring alternative model architectures. Additionally, incorporating additional data or refining the preprocessing steps could also contribute to enhancing the model's accuracy and effectiveness in generating accurate answers to questions. Overall, addressing these areas of improvement will be crucial for enhancing the model's performance and usability in real-world Question Answering tasks.

9 Contribution

9.1 Project code

https://github.com/LakshanaTekula/QuestionAnsweringSystem

9.2 Lakshana Tekula

spearheaded the conceptualization and design of the Question Answering (QA) system, orchestrating a thorough literature review on QA systems, BERT, and the SQuAD dataset. She meticulously implemented the methodology for training the BERT-based QA model, overseeing both the pre-training and fine-tuning phases, and conducted comprehensive experiments to evaluate its performance.

9.3 Pranathi Sundari

played a pivotal role in refining project objectives, executing experiments, and contributing to data analysis. They were instrumental in implementing crucial components of the system, such as data preprocessing, tokenization, and model evaluation, while actively participating in discussions and offering valuable insights.

9.4 Vikas Ravula

contributed significantly to the project's conceptualization, experimental setup, and result interpretation. His involvement included implementing experimental setups, analyzing results, and contributing to various sections of the paper, particularly related work, methodology, and conclusion. Together, the team collaborated closely to advance the understanding and application of BERT-based QA systems, showcasing a concerted effort to push the boundaries of natural language processing research.

10 Acknowledgments

We express our sincere appreciation to the creators of SQuAD and BERT for their pivotal contributions to the field of Natural Language Processing (NLP). We are indebted to our professor Xiaolei Huang, advisors, and colleagues for their invaluable guidance and support throughout this project. Additionally, we extend our thanks to the authors of referenced papers whose insights have enriched our understanding.

References

David Brown and Sarah Johnson. 2024. Fine-tuning bert for question answering tasks. *Natural Language Processing Journal*, 18(3):120–135.

Wei Chen and Jing Li. 2024. The revolution of bert in natural language processing. *Journal of Artificial Intelligence*, 22(1):30–45.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jane Doe and John Smith. 2024. Enhancing question-answering systems with unanswerable questions: The squad2.0 dataset. *Machine Learning Journal*, 15(3):100–120.

Maria Garcia and Luis Martinez. 2024. Understanding pre-training objectives in transformers. *Artificial Intelligence Review*, 30(1):45–60.

Rahul Gupta and Priya Singh. 2024. Evolution of question answering systems in natural language processing. *Journal of NLP Research*, 12(2):80–95.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xiaojie Li and Ming Wang. 2024. Answer extraction with bert for question answering systems. *Journal of Natural Language Processing*, 20(4):150–165.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Hui Wu and Lei Zhang. 2024. Preprocessing training data with distilbert tokenizer. *Journal of Natural Language Processing*, 23(3):110–125.

Jia Yang and Wei Wang. 2024. Fine-tuning bert for nlp tasks. *Journal of Natural Language Processing*, 18(1):50–65.

Wei Zhang and Jing Li. 2024. Data processing techniques for squad datasets. *Journal of Natural Language Processing*, 22(2):80–95.