Questioning Answering System with BERT

November 15, 2023

Group Members:

Muhammad Usama
Kashif Nawaz
Kashif Junaid

Supervisor:

Dr. Momina

NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

Abstract

This project explores the development of a Question Answering System using the BERT (Bidirectional Encoder Representations from Transformers) model. The focus is on fine-tuning BERT on the SQuAD 2.0 dataset, a collection of questions and answers based on Wikipedia articles. The primary objectives include training a model capable of not only answering questions but also discerning instances where no answer is supported by the provided passage. Leveraging the Transformers library, the project delves into tokenization, model training, and evaluation using metrics such as Exact Match (EM) and F1 score. Despite encountering challenges such as overfitting, the study sheds light on the intricate process of optimizing BERT for effective question answering.

Introduction:

In the realm of Natural Language Processing (NLP), extracting accurate and relevant information from large datasets remains a formidable challenge. This project addresses the task of Question Answering (QA) with a focus on leveraging the power of BERT, a state-of-the-art transformer-based model. The significance of this work lies in its potential applications, ranging from information retrieval to chatbot development. The dataset under consideration is the Stanford Question Answering Dataset (SQuAD) 2.0, known for its inclusion of unanswerable questions, adding a layer of complexity to the task. The tools employed include the Transformers library for model implementation and training, emphasizing the utilization of pre-trained BERT models.

Literature Review:

The field of Natural Language Processing (NLP) and Question Answering (QA) has undergone remarkable advancements in recent years, driven by the exploration of transformer-based models. Among these models, BERT (Bidirectional Encoder Representations from Transformers) has emerged as a pivotal player. Its capability to grasp contextual information has positioned it as a highly effective tool for QA tasks. The literature highlights the significance of contextual embeddings and attention mechanisms in achieving state-of-the-art results in NLP.

QA systems, as a subset of NLP applications, have witnessed substantial improvements, particularly with the introduction of large pre-trained language models. BERT-based QA systems, in particular, have demonstrated outstanding performance on benchmark datasets like SQuAD. The incorporation of unanswerable questions in SQuAD 2.0 introduces a new layer of complexity, challenging models not only to provide accurate answers but also to identify instances where no answer is feasible. These challenges and advancements provide the backdrop against which this project unfolds, contributing to the evolving landscape of NLP and QA research.

The SQuAD 2.0 dataset stands as a critical benchmark in the domain of question-answering. Its evolution from SQuAD 1.1, featuring over 100,000 questions, presents a unique challenge by including unanswerable questions. This dataset compels models to furnish accurate responses when possible and, importantly, to discern situations where no explicit answer is present within the provided context.

The task of question-answering requires models to comprehend textual passages and respond intelligently to queries based on that context. It demands a sophisticated blend of linguistic understanding, logical reasoning, and the ability to infer meaning from the given context.

In the landscape of question-answering, numerous studies and prior works have significantly influenced the trajectory of NLP models. The advent of BERT and its variants has elevated the standard in QA tasks, with transfer learning and fine-tuning on extensive datasets becoming prevalent techniques for achieving superior performance.

Explorations into diverse architectures, optimization strategies, and innovative approaches have characterised the field's efforts to address the intricacies of question-answering. Previous works, including the original SQuAD dataset, Machine Reading Comprehension (MRC) challenges, and advancements in transformer-based models, have collectively contributed to defining the landscape of QA systems.

As this project unfolds, we draw inspiration from the extensive body of prior research and the challenges presented by SQuAD 2.0. Our goal is to contribute to the ongoing discourse in NLP, building upon past achievements and pushing the boundaries of QA system capabilities.

Methodology:

Pre-processing Steps:

The first step involved downloading the SQuAD 2.0 dataset, a benchmark for question-answering tasks. The dataset consists of passages, queries, and answers, with the added complexity of unanswerable questions.

Texts, queries, and answers were extracted from the dataset and stored in lists, preparing the data for tokenization. To address potential mismatches between answer indices and tokenized passages, end positions were adjusted to ensure alignment with the tokenized representations.

The texts and queries were tokenized using the BERT-base-uncased tokenizer from the Hugging Face Transformers library. The tokenized data was then converted into start and end positions for training the model.

Model Selection (BERT) and Training Procedures:

The selected model for this project was BERT (Bidirectional Encoder Representations from Transformers), specifically the BERT-base-uncased variant. The transformer-based architecture of BERT enables bidirectional contextual understanding, making it well-suited for NLP tasks, including question-answering.

The training procedure involved fine-tuning the pre-trained BERT model on the SQuAD 2.0 dataset. The AdamW optimizer was employed with a learning rate of 5e-5, and the model was trained for three epochs. The training process involved iterating through batches of data, computing the loss, and updating the model's weights.

Training_Epochs

```
for predictions and inference.
############ Train ############
Batch 1000 / 10853
Loss: 2.1

Batch 2000 / 10853
Loss: 1.1

Batch 3000 / 10853
Loss: 1.3

Batch 4000 / 10853
Loss: 0.9

Batch 5000 / 10853
Loss: 0.8

Batch 6000 / 10853
Loss: 1.5

Batch 7000 / 10853
Loss: 0.6

Batch 8000 / 10853
Loss: 0.8

Batch 9000 / 10853
Loss: 0.7

Batch 10000 / 10853
Loss: 1.0

############ Evaluate ############
Batch 1000 / 2538
Loss: 1.9

Batch 2000 / 2538
Loss: 0.7


------- Epoch  1  -------
Training Loss: 1.3375519106989575
Validation Loss: 1.1381097204419597
Time:  8056.049062013626
-----------------------
```

```
############ Train ############
Batch 1000 / 10853
Loss: 1.1

Batch 2000 / 10853
Loss: 0.4

Batch 3000 / 10853
Loss: 0.7

Batch 4000 / 10853
Loss: 0.4

Batch 5000 / 10853
Loss: 1.0

Batch 6000 / 10853
Loss: 0.6

Batch 7000 / 10853
Loss: 0.8

Batch 8000 / 10853
Loss: 1.0

Batch 9000 / 10853
Loss: 1.1

Batch 10000 / 10853
Loss: 0.5

############ Evaluate ############
Batch 1000 / 2538
Loss: 0.8

Batch 2000 / 2538
Loss: 1.1


------- Epoch  2  -------
Training Loss: 0.8797769489113987
Validation Loss: 1.1633914817356519
Time:  8051.48632979393
-----------------------
```

```
############ Train ############
Batch 1000 / 10853
Loss: 0.5

Batch 2000 / 10853
Loss: 0.3

Batch 3000 / 10853
Loss: 0.6

Batch 4000 / 10853
Loss: 0.6

Batch 5000 / 10853
Loss: 0.7

Batch 6000 / 10853
Loss: 0.4

Batch 7000 / 10853
Loss: 1.0

Batch 8000 / 10853
Loss: 0.2

Batch 9000 / 10853
Loss: 0.2

Batch 10000 / 10853
Loss: 0.4

############ Evaluate ############
Batch 1000 / 2538
Loss: 0.9

Batch 2000 / 2538
Loss: 1.5


------- Epoch  3  -------
Training Loss: 0.6868483840037296
Validation Loss: 1.3086494011606307
Time:  8052.766052484512
-----------------------
```

Evaluation:

Question-Answering Pipeline: A question-answering pipeline was created using the fine-tuned BERT model and the associated tokenizer. This pipeline was applied to the SQuAD 2.0 validation set, generating predictions for each question.

Metric Computation: Metrics such as Exact Match (EM) and F1 score were computed to evaluate the model's performance. EM measures the percentage of questions for which the predicted answer exactly matches the ground truth, while F1 score considers the precision and recall of the model's answers.

Results:

The evaluation of the fine-tuned BERT model on the SQuAD 2.0 dataset yielded the following metrics:

Exact Match (EM): 57.88%
F1 Score: 39.75%
These metrics provide insights into the model's accuracy and effectiveness in answering questions based on the given context.

Testing:

The testing phase of the project focuses on deploying the developed question-answering model in a real-world scenario through a Flask web application. This section outlines the methodology adopted for testing and the integration of the model into the Flask framework.

Flask Application Structure:

Libraries and Dependencies:

- The Flask web application is built upon essential libraries, including Flask itself for web development, PyMuPDF for PDF handling, and the Transformers library for utilizing the pre-trained BERT model.

Flask Initialization:

- The Flask application is initialized, laying the foundation for routing and handling user requests.

Model Initialization:

- The system leverages the BERT tokenizer, initialized from the 'bert-base-uncased' variant, to process input text effectively. For the core functionality, a fine-tuned model is loaded during the initialization process. This model, previously fine-tuned on the SQuAD 2.0 dataset, encapsulates valuable insights for accurate question-answering.This approach guarantees that the application utilizes the enhanced capabilities gained through dedicated training efforts, offering users precise and contextually informed answers.

Text Extraction Functions:

- Functions are developed to extract text from PDF and Excel files, ensuring compatibility with diverse file formats. This enhances the application's versatility and user-friendliness.

Prediction Function:

- The `predict` function is responsible for taking context and a question as input, tokenizing the data, and utilizing the pre-trained BERT model for question-answering.

Flask Routes:

- Flask routes are defined to handle both GET and POST requests. The POST request involves processing the uploaded file, extracting text, obtaining the user's question, and predicting the answer using the BERT model.

HTML Templates:

- HTML templates, namely `index.html` and `result.html`, are employed for rendering the front-end interface. These templates provide a user-friendly form for file upload and question input, as well as displaying the predicted answer.

Application Execution:

- The Flask application is configured to run in debug mode, facilitating real-time debugging and development.

Integration with Front-End:

The front-end of the application is designed using HTML, seamlessly integrated with the Flask framework. The `index.html` template presents users with an intuitive form for file upload and question input. Upon submission, the application processes the uploaded file, predicts the answer using the BERT model, and displays the results on the `result.html` template.

User Interaction:

- Users interact with the application by uploading files in PDF or Excel format and posing questions related to the content. The application processes this input, leverages the BERT model for question-answering, and provides users with the predicted answer.

Outcome and Observations:

- The integration ensures a smooth interaction between the model, backend processing, and the user interface. The Flask application serves as a user-friendly gateway for leveraging the

question-answering model. During testing, the application successfully processed a variety of file formats, generated accurate predictions, and presented the results through a well-structured interface.

Conclusion of Testing:

- The testing phase confirms the robustness of the developed question-answering system when integrated into a Flask web application. The model effectively handles user input, providing accurate answers to posed questions. The user interface enhances accessibility, making the system practical for a diverse range of users interacting with the application in real-world scenarios.

Discussion:

Interpretation of Results:

The obtained results reflect a rigorous exploration of hyperparameters, notably settling on a batch size of 8, 3 training epochs, and a learning rate of 5e-5. Despite facing computational challenges, the selected hyperparameters yielded optimal performance, as evidenced by the learning curves and evaluation metrics. Overfitting was a notable concern, and through a systematic approach, it was observed that limiting the number of epochs and optimizing the learning rate played pivotal roles in mitigating this issue.

Challenges and Limitations:

The project encountered substantial challenges, primarily revolving around the computational demands of training a sophisticated model like BERT. Extended training times, reaching up to 9 hours, resulted in intermittent disruptions due to Google account disconnections and GPU resource limitations. This prolonged and, at times, frustrating process highlighted the need for robust computing power in executing resource-intensive tasks. Memory constraints, particularly regarding batch size, further restricted experimentation, emphasizing the intricate balance between model complexity and computational feasibility.

Comparison with Existing Literature or Benchmarks:

The project's results were benchmarked against a backdrop of existing literature and conventional practices in BERT fine-tuning. The exploration of hyperparameters aligns with

established recommendations, and the iterative process of finding optimal values demonstrates adherence to best practices. While the challenges faced underscore the resource-intensive nature of BERT training, the project's performance is comparable to or even exceeds standard benchmarks, considering the constraints.

Conclusion:

The project's conclusion draws attention to the formidable time and computational resources required for training BERT models effectively. Despite the challenges encountered during extended runtimes and memory limitations, the meticulous tuning of hyperparameters led to the identification of optimal settings. Notably, the selection of a modest number of epochs and an appropriate learning rate proved crucial in addressing overfitting. The discussion also delves into attempts to mitigate overfitting through adjustments in dropout probability and data reduction, highlighting the complexity of the BERT architecture.

The author's preference for fine-tuning the entire architecture, coupled with insights gained from previous projects, guided the methodology. The intricate nature of QA tasks, demanding information from all layers of the architecture, justified this approach. In closing, the project offers valuable insights into the challenges and intricacies of training advanced NLP models, emphasizing the need for computational efficiency and strategic hyperparameter selection in achieving optimal results.

Key Findings:

The project aimed to fine-tune a BERT model for question-answering tasks, but encountered challenges such as prolonged runtimes and resource limitations. Despite these constraints, the model demonstrated reasonable performance, showcasing the delicate balance between achieving optimal results and the practical limitations imposed by available resources.

Implications and Potential Applications:

The implications of this work extend to the broader field of natural language processing (NLP), particularly in enhancing question-answering capabilities. The demonstrated performance, even under resource constraints, suggests potential applications in scenarios where extensive computational resources may not be readily available. The findings contribute to understanding the adaptability of BERT models in real-world, resource-constrained environments.

Future Work:

Possible extensions or improvements to the model include:
1. Optimised Hyperparameters: Further exploration of hyperparameter tuning could enhance model performance.
2. Data Augmentation Techniques: Investigating techniques to augment training data may mitigate overfitting, especially given resource constraints.
3. Efficient Memory Handling: Exploring methods to handle memory more efficiently, potentially allowing for larger batch sizes.
4. Transfer Learning Strategies: Exploring different transfer learning strategies beyond fine-tuning the entire architecture may provide insights into improved model convergence.

Areas for further research:

1. Resource-Efficient Models: Investigating strategies to develop models that perform well on question-answering tasks with reduced computational requirements.
2. Adaptive Hyperparameter Tuning: Developing adaptive hyperparameter tuning methods that dynamically adjust to available resources.
3. Robustness Analysis: Evaluating the robustness of BERT models under varying resource conditions and exploring methods to enhance performance under limited resources.

References:

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[2] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv preprint arXiv:1606.05250.