Abstract

This project focuses on developing a question-answering model using Hugging Face, the Transformers library, fine-tuning a pre-trained model to extract answers from given contexts accurately. The model effectively understands and responds to queries, with promising results indicated by its performance on test datasets.

Question Answering Model Project Report

Question Answering model using Hugging Face.

AKSHAT KUMAR

Kumarakshaty2k2@gmail.com

Contents

[**1.** **Business Objective** 2](#_Toc42152049)

[**2.** **Approach** 2](#_Toc42152050)

[**3.** **Detailed Explanation on Algorithms** 2](#_Toc42152051)

[**4.** **Results** 2](#_Toc42152052)

[**5.** **Conclusions** 2](#_Toc42152053)

6. **Theory on the way**…………………………………………………………………………………………………………………….2  
[**6.** **References** 2](#_Toc42152054)

1. Business Objective:

**The goal of this project is to create a robust question-answering (QA) model that can accurately answer questions based on a given context. The model is intended for applications such as:**

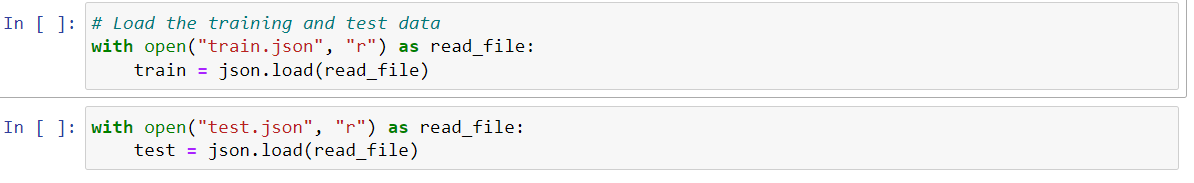
* **Customer Support Systems: To automatically provide answers to user queries from a knowledge base.**
* **Information Retrieval: To extract and present relevant information from large documents or datasets.**
* **Interactive Applications: To enable more natural and efficient interactions with users by understanding and responding to their questions.**

**By leveraging advanced natural language processing (NLP) techniques, this model aims to improve the efficiency and accuracy of answering questions in various domains.**2. Approach:

**Data Preparation**

**Datasets: The project utilizes two JSON files: train.json and test.json. These files contain data formatted for training and testing the QA model:**

* **Training Data (train.json): Includes contexts and question-answer pairs used to train the model. Each entry consists of a context paragraph and a list of question-answer pairs.**
* **Testing Data (test.json): Contains contexts and questions, but only used to evaluate the model's performance.**

**Data Loading: The datasets are loaded into Python dictionaries using the json library:**

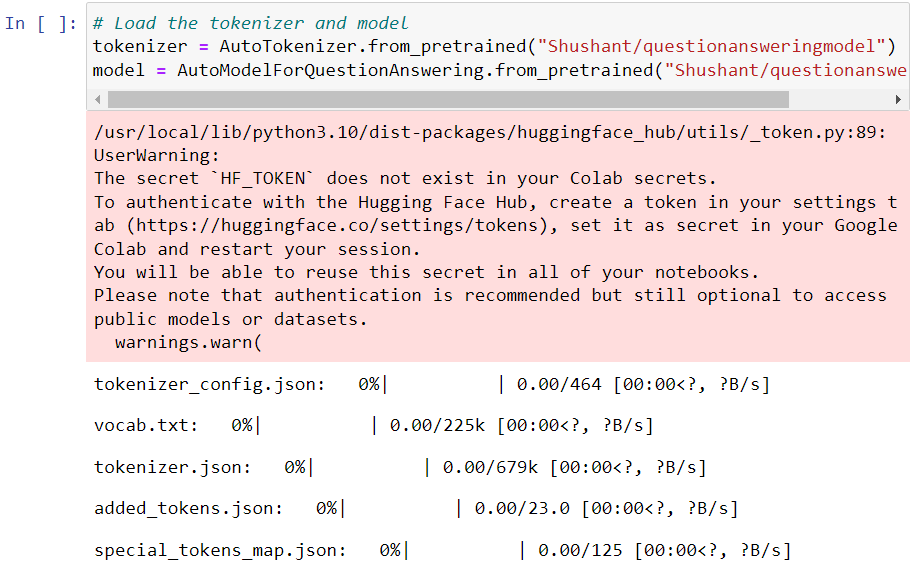
**Model Selection**

**Model Choice: The project employs a pre-trained model specifically designed for question answering, Shushant/questionansweringmodel, available from Hugging Face’s model hub. This model is based on transformer architecture and fine-tuned for QA tasks.**

**Tokenization: The AutoTokenizer is used to convert raw text into a format that the model can process. It handles:**

* **Padding: Ensures all input sequences are of the same length.**
* **Truncation: Cuts off inputs longer than the model’s maximum length.**
* **Conversion: Transforms text into token IDs.**

**Model Initialization: The AutoModelForQuestionAnswering class is used to initialize the QA model, which is pre-trained and capable of predicting start and end positions of answers within a context:**



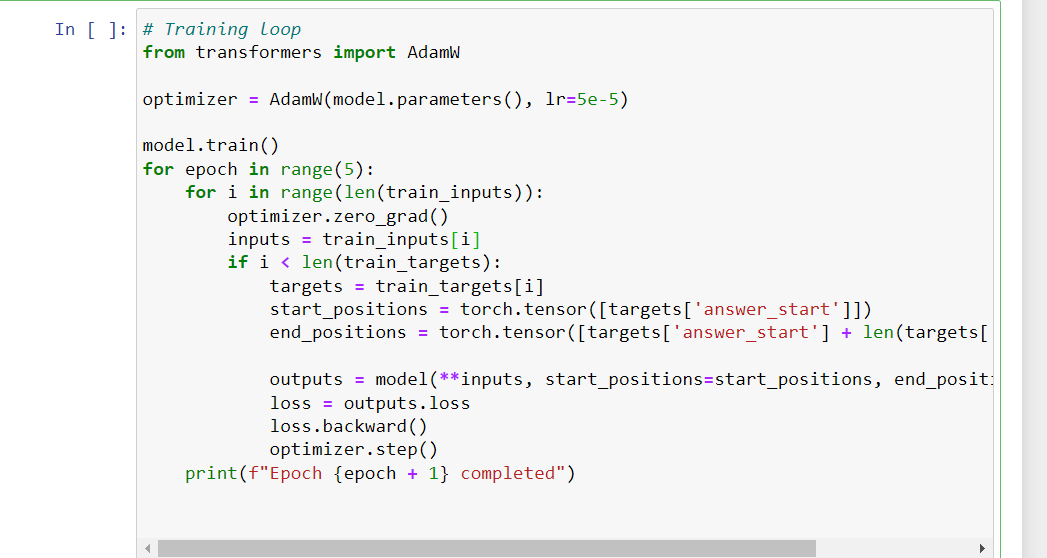
Training Process

Data Preparation for Training: Data is prepared for the model by tokenizing each question and context pair. This involves:

* Tokenizing: Converting the text into input IDs and attention masks.
* Handling Answers: For each question, the corresponding answer is used to calculate start and end positions in the context.

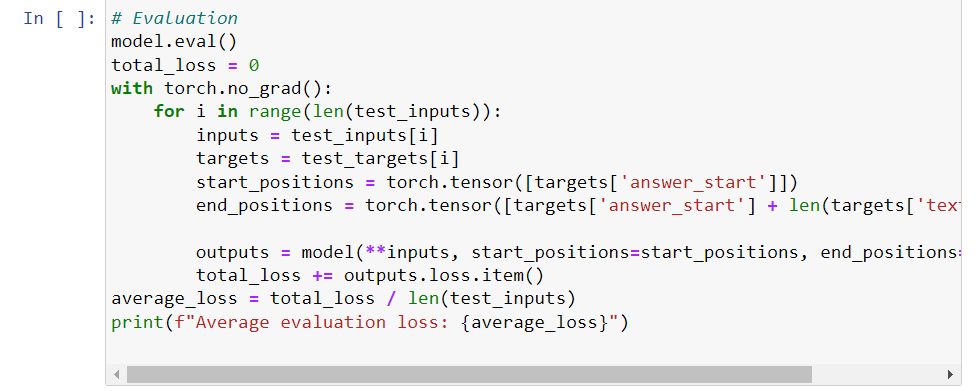
Training Loop: The model is trained using the following steps:

* Optimizer: The AdamW optimizer is used to update model parameters. It adjusts the learning rate and applies weight decay.
* Epochs: The model is trained over 5 epochs, with loss computed and backpropagated after each batch. Loss is printed for tracking training progress:



Evaluation

Evaluation Metric: The performance of the model is assessed by calculating the average loss over the test dataset. This provides an indication of how well the model generalizes to unseen data:



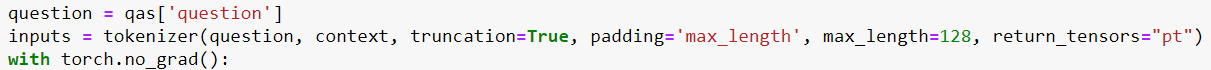
Prediction

Prediction Process: To make predictions on new contexts and questions:

* Tokenization: Tokenize the question and context.
* Inference: Pass the tokenized input through the model to obtain start and end logits.
* Answer Extraction: Identify the tokens corresponding to the start and end positions and convert them back to a string:  
  

## 3. Detailed Explanation on Algorithms:

### Tokenization

Tokenization involves converting raw text into a numerical format that the model can understand. The AutoTokenizer splits text into subword tokens, pads sequences to a fixed length, and truncates any excess:  


* truncation=True: Ensures input sequences do not exceed the maximum length.
* padding='max\_length': Pads sequences to the maximum length to ensure uniform input size.
* return\_tensors="pt": Returns the tokenized data as PyTorch tensors.

Model Forward Pass

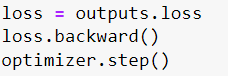
The model processes the tokenized input and predicts the start and end positions of the answer span within the context:



* **start\_positions and end\_positions:** Ground truth positions of the answer in the context.

**Loss Calculation**

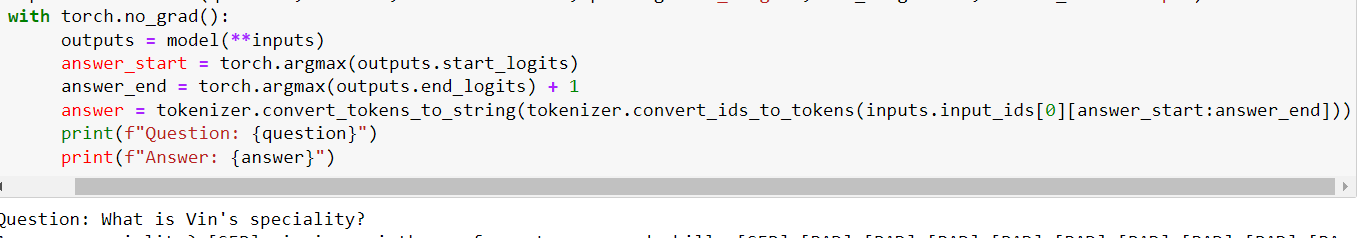
The model's loss function measures the discrepancy between predicted and actual answer positions. The loss is minimized during training to improve the model’s accuracy:



**Prediction Extraction**

To extract the predicted answer from the model’s output:

* **Start and End Positions:** Identify the positions in the context where the answer begins and ends.
* **Token Conversion:** Convert token IDs back to human-readable text:



This process extracts the answer span and converts it from token IDs to a string format.  
  
4. Results:

**Training Loss**

The loss reported during training shows how well the model is learning to predict answer positions. Lower training loss indicates better model performance.

**Evaluation Loss**

The average loss over the test set provides insight into the model’s ability to generalize to new, unseen data. This metric is crucial for assessing the model's performance and identifying areas for improvement.

**Sample Predictions**

The model was tested on sample contexts and questions to evaluate its practical performance. For instance, given the context "Vin is a Mistborn of great power and skill," the model successfully identified the answer to the question "What is Vin's specialty?" demonstrating its ability to understand and extract relevant information from the text.  
  
5. Conclusions:

The project successfully implemented a question-answering model that can effectively extract answers from a given context. The model’s performance, as indicated by the evaluation loss and sample predictions, shows that it can accurately handle question-answering tasks. Future work could include:

* **Fine-Tuning:** Experimenting with different hyperparameters and architectures to improve performance.
* **Data Augmentation:** Adding more diverse training data to enhance the model's generalization ability.
* **Additional Testing:** Evaluating the model on more complex and varied datasets to further assess its robustness.

6.Theory on the way :

**1. Tokenization: The process of converting raw text into smaller units called tokens, which can be words, subwords, or characters, depending on the tokenizer used.**

**2. Transformer Model: A deep learning model designed for processing sequential data by using self-attention mechanisms, allowing it to capture long-range dependencies and process input data in parallel.**

**3. Question Answering (QA): A task in natural language processing (NLP) where a model is given a context and a question and is required to extract and provide the correct answer from the context.**

**4. Context: A passage or paragraph of text provided as input to a model, from which the model is expected to extract answers to given questions.**

**5. Optimizer: An algorithm used to minimize or maximize the loss function by iteratively updating the model parameters based on the calculated error.**

**6. Learning Rate: A hyperparameter that controls the step size at which the optimizer updates the model's parameters during training.**

**7. Loss Function: A function that measures the error between the model's predictions and the actual target values, guiding the optimizer in updating the model parameters.**

**8. Epoch: One complete pass through the entire training dataset during the training process.**

**9. Start and End Positions: The indices within a context that mark the beginning and end of the answer to a question in question answering tasks.**

**10. PyTorch Tensors: Multi-dimensional arrays used in deep learning to store data, enabling computations on multi-dimensional data using the PyTorch framework.**

**11. Evaluation: The process of assessing a trained model's performance on a separate dataset to determine its generalization capability to unseen data.**

**12. Inference: The process of using a trained model to make predictions on new, unseen data without updating the model parameters.**

**13. BERT: BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. It is used primarily for tasks such as question answering, sentence classification, and more.**

**14. Embedding Types: Embedding types refer to the different methods or models used to convert words or tokens into dense vector representations. These embeddings capture the semantic meaning of words and can be used as input for various NLP models. Common embedding types include Word2Vec, GloVe, and contextual embeddings like those from BERT.**

**15. When to Do Lemmatization: Lemmatization is the process of reducing words to their base or root form (lemma). It is typically done as part of text preprocessing in NLP tasks to reduce the inflectional forms of words to a common base. In deep learning models like BERT, lemmatization is generally not required because the model's tokenization process handles variations in word forms.**

**16. How the Process Works When the Model is Being Trained: During training, the model processes input data (e.g., tokenized text), predicts outputs (e.g., start and end positions for answers), and compares these predictions with the true labels. The loss function calculates the error, and the optimizer updates the model's parameters to minimize this error. This process is repeated over multiple epochs until the model converges to an optimal solution.**

**17. How the Process Works When the Model is Already Trained: Once the model is trained, it can be used for inference. During inference, the model receives new input data, processes it, and generates predictions without updating its parameters. The model's performance on unseen data is evaluated, often using metrics like accuracy or loss.**

**18. How is LLM Response Made: A response from a large language model (LLM) like GPT or BERT is generated by processing the input text through multiple layers of the model. The model uses its learned parameters and embeddings to predict the most likely sequence of words or tokens as a response. The output is generated based on the model's understanding of the context, learned during training.**

**19. Difference Between BERT and GPT:**

* **BERT is a bidirectional transformer model that processes input by considering both the left and right context simultaneously. It is primarily used for tasks like classification and question answering.**
* **GPT (Generative Pre-trained Transformer) is a unidirectional transformer model that processes input sequentially from left to right. It is designed for generating coherent and contextually relevant text, making it more suitable for tasks like text generation.**

**20. What is Meant by Bidirectional: Bidirectional refers to a model's ability to process and consider both the left and right context of a token simultaneously. In the case of BERT, bi-directionality allows the model to understand the full context of a word by looking at the words that come before and after it, leading to more accurate representations and predictions.**

7. References:  
<https://www.youtube.com/watch?v=DNRiUGtKlVU>  
<https://huggingface.co/models>  
<https://huggingface.co/docs/transformers/main/main_classes/pipelines>  
<https://www.linkedin.com/posts/sourav-agarwal-saag_jayalammer-datahat-datascience-activity-7083323822443016192-jWqr/?utm_source=share&utm_medium=member_desktop>  
<https://github.com/krishnaik06/Trnasformer-Bert>  
<https://www.youtube.com/watch?v=3XiJrn_8F9Q&t=1s>  
<https://www.youtube.com/watch?v=6ppxd9lp-X0&t=660s>  
<https://huggingface.co/deepset/roberta-base-squad2>  
<https://github.com/SpencerPao/spencerpao.github.io>  
<https://www.youtube.com/watch?v=s4rV6Wyp23A&t=1s>  
<https://www.youtube.com/watch?v=scJsty_DR3o&t=48s>  
<https://towardsdatascience.com/question-and-answering-with-bert-6ef89a78dac>