Hack to Hire 2024

Ambar Sood
New Delhi, 110014 | 9555888566 | ambarsood@gmail.com

ABSTRACT

This report presents a detailed overview of the processes involved in analyzing, preprocessing, and applying machine learning models to the Quora Question Answer Dataset. It covers data integrity checks, text preprocessing techniques, and visualizations to understand the dataset's characteristics. The report also includes the fine-tuning of a GPT-2 model for generating responses, the creation of an interactive interface using Gradio, and the evaluation of model performance using various metrics. Additionally, different models and techniques for similarity-based query answering are explored and compared. The methodologies and insights provided serve as a comprehensive guide for implementing effective question-answering systems in NLP.

Keywords: Quora Question Answer Dataset, Data Preprocessing, GPT-2 Fine-Tuning, Gradio Interface, Similarity-Based Query Answering, NLP

I. INTRODUCTION

This report provides a comprehensive overview of the methodologies and processes employed in analyzing and preprocessing the Quora Question Answer Dataset. The primary objective is to prepare the dataset for advanced applications such as machine learning and natural language processing (NLP). The report is divided into three main sections: Data Preprocessing and Analysis, Fine-Tuning GPT-2, and Similarity-Based Query Answering.

In the **Data Preprocessing and Analysis** section, we delve into the initial steps taken to load and inspect the dataset, ensuring data integrity through checks for null values and duplicates. The text data undergoes various preprocessing techniques, including tokenization, stopword removal, stemming, and lemmatization. These steps are crucial for cleaning the dataset and making it suitable for further analysis. Visualizations, such as histograms, word clouds, and n-gram analysis, provide insights into the dataset's characteristics, aiding in understanding the distribution and common terms used in the questions.

The **Fine-Tuning GPT-2** section outlines the approach to adapting a pre-trained GPT-2 model to our specific dataset. The process involves preparing the dataset, configuring the tokenizer, and fine-tuning the model using the Hugging Face library. We also describe the creation of an interactive interface with Gradio, allowing users to input questions and receive responses from the fine-tuned model. The model's performance is evaluated using metrics like ROUGE, BLEU, and F1-score, highlighting areas for improvement.

In the **Similarity-Based Query Answering** section, we explore various models and techniques to match user queries with the most relevant predefined questions and answers. These models include Bag of Words (BOW), Word2Vec, GloVe, and BERT. Each model is described in detail, covering their training, embedding generation, and similarity measurement processes. We compare the strengths and limitations of each approach, concluding that BERT offers the best performance for understanding and handling complex queries.

By following these detailed methodologies, we ensure a robust preparation of the dataset and a thorough evaluation of different approaches to question answering. This report serves as a guide for future improvements and implementations in the field of NLP and machine learning.

The organization of this document is as follows. In Section 2 (Methods and Material), I'll give detail of any modifications to equipment or equipment constructed specifically for the study and, if pertinent,

provide illustrations of the modifications. In Section 3 (Result and Discussion), present your research findings and your analysis of those findings. Discussed in Section 4(Conclusion) a conclusion is the last part of something, its end or result.

II. METHODOLOGY

[1] Data Loading and Initial Inspection

1. **Loading the Dataset:**
   - The dataset was loaded from a JSON file using the pandas library. This format was suitable for handling the line-delimited JSON data.
2. **Initial Data Inspection:**
   - **Data Structure:** The dataset contains **56,402** entries with two columns: question and answer.
   - **Column Details:**
     - **question:** Contains the questions from the dataset.
     - **answer:** Contains the corresponding answers to the questions.
   - **Data Type:** Both columns are of type object, indicating they are text-based fields.
3. **Data Integrity Check:**
   - **Null Values:** There were no missing values in either column, as confirmed by the isnull().sum() method.
   - **Descriptive Statistics:**
     - **question:** 32,34 unique questions, with the most frequent question appearing 106 times.
     - **answer:** 54,726 unique answers, with the most frequent answer appearing 89 times.

[2] Data Cleaning

1. **Removing Duplicates:**
   - Duplicates were removed from the dataset to ensure each question-answer pair is unique.
2. **Handling Missing Values:**
   - Although there were no missing values initially, a general practice was demonstrated using dropna() and fillna() methods.

[3] Text Preprocessing

1. **Tokenization:**
   - **Purpose:** To split the questions into individual words or tokens.
   - **Method:** Applied word_tokenize from the NLTK library to tokenize each question.
2. **Removing Stopwords:**
   - **Purpose:** To eliminate common words that do not contribute meaningful information for analysis.
   - **Method:** Defined a custom function to filter out stopwords using NLTK's predefined set of English stopwords.
3. **Stemming and Lemmatization:**
   - **Stemming:**
     - **Purpose:** To reduce words to their base or root form.
     - **Method:** Used PorterStemmer from NLTK.
   - **Lemmatization:**
     - **Purpose:** To convert words to their base or dictionary form, considering the context.
     - **Method:** Used WordNetLemmatizer from NLTK.

[4] Data Analysis

1. **Question Length Distribution:**
   - **Objective:** To visualize the distribution of question lengths.
   - **Method:** Plotted a histogram with kernel density estimation (KDE) to display the frequency distribution of question lengths.
2. **Word Cloud Visualization:**
   - **Objective:** To provide a visual representation of the most frequent words in the questions.
   - **Method:** Generated a word cloud where the size of each word reflects its frequency across all questions.
3. **N-gram Analysis:**
   - **Objective:** To analyze and visualize common word pairs (bigrams) and triplets (trigrams) within the questions.
   - **Method:** Used CountVectorizer from sklearn to extract n-grams and plot their frequency.

III. DATA LOADING AND INITIAL DISCUSSION]

The objective is to provide a detailed explanation of the data preprocessing and analysis procedures performed on the Quora Question Answer Dataset. The dataset was loaded and analyzed to prepare it for further applications, such as machine learning or natural language processing tasks.

Data Loading and Initial Inspection

1. **Loading the Dataset:**

   The dataset was loaded from a JSONL file using the pandas library. This format was

suitable for handling the line-delimited JSON data.

2. **Initial Data Inspection:**

   Data Structure: The dataset contains 56,402 entries with two columns: question and answer.

   Column Details:

   - question: Contains the questions from the dataset.
   - answer: Contains the corresponding answers to the questions.

   Data Type: Both columns are of type object, indicating they are text-based fields.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56402 entries, 0 to 56401
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   question  56402 non-null  object
 1   answer    56402 non-null  object
dtypes: object(2)
memory usage: 881.4+ KB
None
```

3. **Data Integrity Check:**

   Null Values: There were no missing values in either column, as confirmed by the isnull().sum() method.

   Descriptive Statistics:

   - question: 32,34 unique questions, with the most frequent question appearing 106 times.
   - answer: 54,726 unique answers, with the most frequent answer appearing 89 times.

Data Cleaning

1. **Removing Duplicates:**

   Duplicates were removed from the dataset to ensure each question-answer pair is unique.

2. **Handling Missing Values:**

   Although there were no missing values initially, a general practice was demonstrated using dropna() and fillna() methods.

Text Preprocessing

1. **Tokenization:**

   Purpose: To split the questions into individual words or tokens.

   Method: Applied word_tokenize from the NLTK library to tokenize each question.

2. **Removing Stopwords:**

   Purpose: To eliminate common words that do not contribute meaningful information for analysis.

   Method: Defined a custom function to filter out stopwords using NLTK's predefined set of English stopwords.

3. **Stemming and Lemmatization:**

Stemming:

- Purpose: To reduce words to their base or root form.
- Method: Used PorterStemmer from NLTK.

Lemmatization:

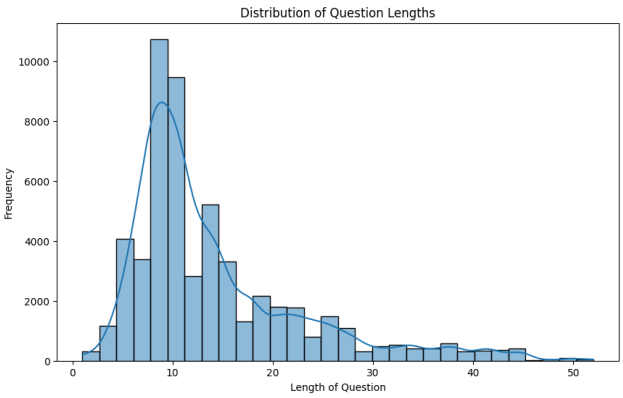
- Purpose: To convert words to their base or dictionary form, considering the context.
- Method: Used WordNetLemmatizer from NLTK.

Data Analysis

1. **Question Length Distribution:**

Objective: To visualize the distribution of question lengths.

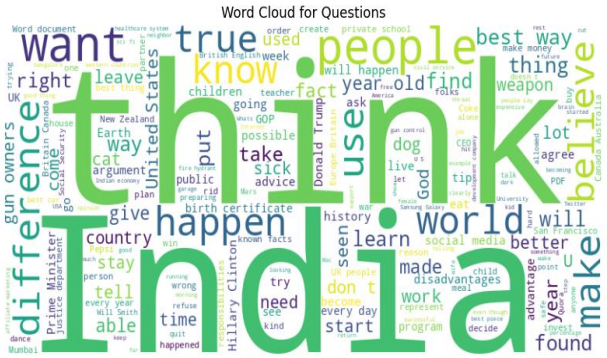
Method: Plotted a histogram with kernel density estimation (KDE) to display the frequency distribution of question lengths.



2. Word Cloud Visualization:

Objective: To provide a visual representation of the most frequent words in the questions.

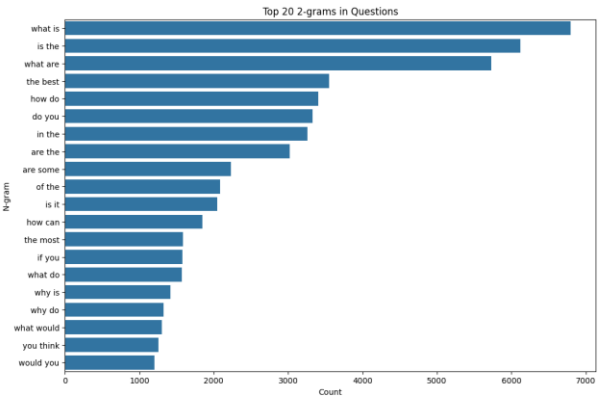
Method: Generated a word cloud where the size of each word reflects its frequency across all questions.

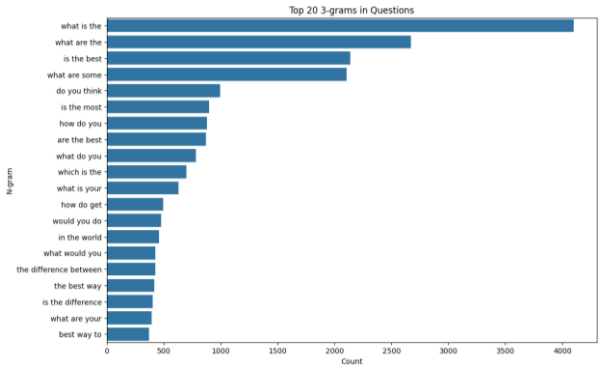


3. N-gram Analysis:

Objective: To analyze and visualize common word pairs (bigrams) and triplets (trigrams) within the questions.

Method: Used CountVectorizer from sklearn to extract n-grams and plot their frequency.





Summary

The preprocessing steps ensured that the dataset was cleaned and text was appropriately prepared for analysis. The visualizations provided insights into the characteristics of the questions, such as their length distribution and common terms. N-gram analysis further helped in understanding the frequent phrases used in the dataset. These steps are critical for preparing the dataset for more advanced analyses or machine learning tasks.

IV. FINETUNING GPT-2

This approach is to fine-tuning a GPT-2 model on a custom dataset and creating an interactive interface using Gradio. The process involves preparing a dataset, training a model, and evaluating its performance. Additionally, an interface is created for user interaction.

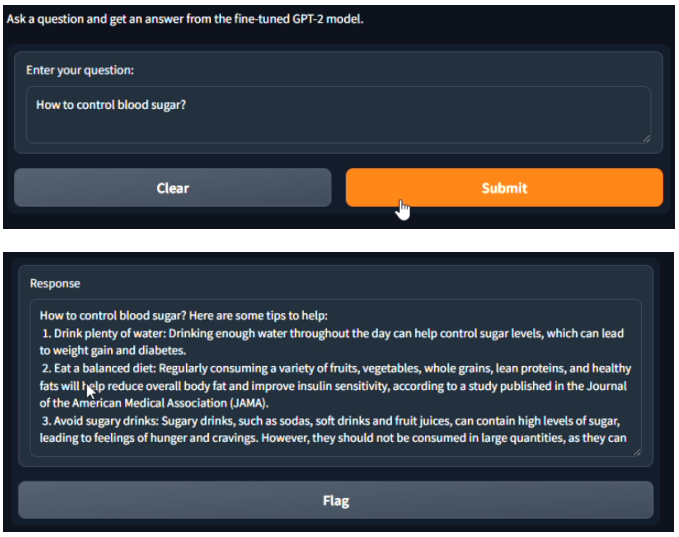
Approach Breakdown

1. Dataset Preparation:

- **Loading Data:**
  - The dataset is loaded from a CSV file using Pandas.
  - The dataset consists of question-answer pairs.
- **Data Preprocessing:**
  - Each row in the dataset is combined into a single text string formatted as "Question: {question} Answer: {answer}".

- This combined text is used to create a new dataset in the Hugging Face Dataset format.

2. **Model Fine-tuning:**
   - **Tokenizer and Model Initialization:**
     - The GPT-2 tokenizer and model are loaded from the Hugging Face library.
     - The tokenizer is modified to include a padding token, which is set to the same value as the end-of-sequence token.
   - **Tokenization:**
     - The text data is tokenized with padding and truncation to ensure that input sequences fit within the model's maximum length (512 tokens).
     - The tokenized dataset is prepared for training.
   - **Training Configuration:**
     - Training arguments are specified, including the number of epochs, batch sizes, and learning rate parameters.
     - The Trainer class from Hugging Face is used to manage the training process.
   - **Model Training:**
     - The model is trained on the prepared dataset.
     - The trained model and tokenizer are saved to disk for later use.
3. **Model Interaction:**
   - **Generating Responses:**
     - The model is loaded for inference.
     - For a given input question, the model generates a response using parameters such as beam search, temperature, and nucleus sampling.
     - The generated response is decoded from token IDs to text.
4. **Interface Creation:**
   - **Gradio Interface:**
     - Gradio is used to create a web-based interface for user interaction with the fine-tuned model.
     - The interface allows users to input questions and receive responses from the model.
     - The interface is defined with input and output components, a title, and a description.



5. **Model Evaluation:**
   - **Metrics Calculation:**
     - Evaluation is performed using ROUGE, BLEU, and F1-score metrics.
     - ROUGE measures the overlap of n-grams between generated responses and reference answers.
     - BLEU evaluates the precision of n-grams and the overall similarity to reference responses.

- F1-score assesses the balance between precision and recall of the responses.

Metric	Type	Low Precision	Mid Precision	High Precision
ROUGE-1	AggregateScore	0.0083	0.0083	0.0083
ROUGE-2	AggregateScore	0	0	0
ROUGE-L	AggregateScore	0.0083	0.0083	0.0083
ROUGE-Lsum	AggregateScore	0.0083	0.0083	0.0083
BLEU	Score	0	-	-
BLEU (Precisions)	List	0.0087	0	0
Brevity Penalty	Value	1	-	-
Length Ratio	Ratio	115	-	-
F1-score	Score	0	-	-

Metric	Type	Low Recall	Mid Recall	High Recall
ROUGE-1	AggregateScore	1	1	1
ROUGE-2	AggregateScore	0	0	0
ROUGE-L	AggregateScore	1	1	1
ROUGE-Lsum	AggregateScore	1	1	1
BLEU	Score	-	-	-
BLEU (Precisions)	List	-	-	-
Brevity Penalty	Value	-	-	-
Length Ratio	Ratio	-	-	-
F1-score	Score	-	-	-

Metric	Type	Mid F-measure	High F-measure	Additional Details
ROUGE-1	AggregateScore	0.0165	0.0165	High recall but low precision for unigram overlap.
ROUGE-2	AggregateScore	0	0	No overlap in bigrams.
ROUGE-L	AggregateScore	0.0165	0.0165	Similar to ROUGE-1; longest common subsequence results.
ROUGE-Lsum	AggregateScore	0.0165	0.0165	Similar to ROUGE-L; used for summarization tasks.
BLEU	Score	-	-	No n-gram overlap; precision values for n-grams are zero.
BLEU (Precisions)	List	-	-	1-gram precision is low; higher-order n-grams have zero precision.
Brevity Penalty	Value	-	-	The generated text is longer than the reference text.
Length Ratio	Ratio	-	-	Length of generated text is 115, reference length is 1.
F1-score	Score			

- **Evaluation Results:**
  - ROUGE Scores: Indicate low precision and recall, suggesting that the model's generated responses have less overlap with reference answers.
  - BLEU Score: Shows a score near 0, reflecting poor precision of generated n-grams.
  - F1-score: Indicates a score near 0, highlighting the inadequacy in generating responses that match reference answers.

Summary

The approach involves the end-to-end process of fine-tuning GPT-2 on a custom dataset, creating an interactive interface with Gradio, and evaluating the model's performance. The fine-tuning process adapts the model to specific question-answer pairs, while the interface facilitates user interaction. Evaluation metrics suggest that further refinement is needed to improve the model's performance, as indicated by the low scores across ROUGE, BLEU, and F1 metrics.

Future improvements could include adjusting training parameters, with better resources and more epochs better score will be achieved, alternative to this method is semantics similarity approach.

V. SIMILARITY-BASED QUERY ANSWERING

This approach is for answering user queries by automatically retrieving the most similar question and corresponding answer from a predefined set of questions and answers. The approach involves calculating the semantic similarity between the user's query and stored questions using various models and techniques. These models include Bag of Words (BOW), Word2Vec, GloVe, and BERT.

1. Introduction

The objective of this approach is to match user queries with the closest predefined question based on semantic similarity. This process involves transforming questions and queries into vector representations and using similarity measures to identify the most relevant question-answer pair.

2. Models and Techniques

2.1 Bag of Words (BOW)

Description: The Bag of Words (BOW) model is a fundamental technique in Natural Language Processing (NLP) where each text (e.g., a sentence or a document) is represented as a vector. In this model, a sentence is transformed into a vector of fixed length, where each element of the vector corresponds to a word in the vocabulary. The value in each position of the vector represents the frequency or binary presence of the word in the sentence.

Process:

1. **Preprocessing:** Text is converted to lowercase, non-alphanumeric characters are removed, and optionally, stopwords (common words that do not add significant meaning) are eliminated.
2. **Vector Representation:** Each sentence is converted into a vector based on the word frequency or presence in the vocabulary.
3. **Similarity Measurement:** The similarity between the query and predefined questions is computed using cosine similarity, which measures the cosine of the angle between two vectors. This determines how closely related the vectors are.

Limitations: BOW does not capture semantic meaning or context beyond word frequency, making it less effective for understanding nuanced queries.

2.2 Word2Vec

Description: Word2Vec is a model that creates word embeddings, which are dense vector representations of words. These embeddings capture semantic meaning based on the context in which words appear. Word2Vec is typically trained using the Skip-gram model, which predicts surrounding words given a central word.

Process:

1. **Training:** Word2Vec is trained on a large corpus of text to learn word embeddings. Pre-trained models, such as the Google News Word2Vec model, are commonly used.
2. **Phrase Embeddings:** To represent a sentence or phrase, individual word embeddings are averaged or summed to form a vector representation.
3. **Similarity Measurement:** The similarity between vectors representing the query and predefined questions is calculated using cosine similarity.

Advantages: Word2Vec captures semantic relationships between words, allowing for better understanding of context compared to BOW.

Limitations: Word2Vec may not handle out-of-vocabulary words well and may miss more subtle semantic nuances.

2.3 GloVe

Description: GloVe (Global Vectors for Word Representation) is another popular word embedding technique. Unlike Word2Vec, which uses local context, GloVe generates embeddings based on the global word-word co-occurrence statistics in a corpus. This approach utilizes matrix factorization

on the co-occurrence matrix to produce word vectors.

Process:

1. **Training:** GloVe is trained on a large corpus to capture global statistical information about word co-occurrences.
2. **Phrase Embeddings:** Similar to Word2Vec, phrase embeddings are obtained by aggregating the embeddings of individual words.
3. **Similarity Measurement:** Cosine similarity is used to compare the vector representations of queries and predefined questions.

Advantages: GloVe often performs well on tasks requiring an understanding of global word relationships.

Limitations: Like Word2Vec, GloVe may struggle with understanding complex or nuanced meanings in text.

2.4 BERT

Description: BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art language model that uses deep learning to generate context-aware embeddings. BERT processes text in both directions (left-to-right and right-to-left) and captures the meaning of words based on their surrounding context. This enables BERT to understand subtleties and nuances in language more effectively than previous models.

Process:

1. **Training:** BERT is pre-trained on a large corpus using masked language modeling and next sentence prediction tasks.
2. **Embedding Generation:** BERT generates embeddings for each word in a sentence based on its context. The sentence is represented by the embeddings of its words combined in a meaningful way.
3. **Similarity Measurement:** Cosine similarity is computed between the vector representation of the query and the vectors of predefined questions to find the closest match.

Advantages: BERT excels at capturing context and handling complex queries, making it highly effective for understanding nuanced language.

Limitations: BERT requires significant computational resources and may be more complex to implement compared to simpler models.

3. Comparison of Approaches

- **Bag of Words:** Simple and easy to implement but lacks the ability to understand semantic meaning beyond word frequency.
- **Word2Vec and GloVe:** Improve on BOW by capturing semantic relationships between words, but may still miss contextual nuances.
- **BERT:** Provides the most accurate understanding of context and nuances, making it ideal for complex queries, but is computationally intensive.

Summary

The similarity-based approach for answering queries involves converting text into vector representations and measuring similarity to find the most relevant answer. BERT offers the best performance in terms of understanding and handling complex queries, while Word2Vec and GloVe provide a balance between performance and computational efficiency. BOW, though simple, may not effectively capture the semantic meaning required for accurate query answering