**Report on Indigo Hack2Hire-2024 Data Scientist: Case Study**

**Dataset Overview**

The provided dataset consists of 56,402 entries with two columns: 'question' and 'answer.' Both columns contain textual data, and there are no missing values. However, duplicates are present, and the dataset requires cleaning to ensure the quality and reliability of the data for fine-tuning SATA NLP models aimed at QA-chatbots.

**Initial Dataset Analysis**

1. **Shape and Size**:
   * The dataset comprises 56,402 rows and 2 columns.
   * Memory usage: Approximately 881.4 KB.
2. **Column Details**:
   * Question: Contains the text of the questions.
   * Answer: Contains the text of the answers.
   * Both columns are of the 'object' data type (string).
3. **Descriptive Statistics**:
   * **Count**: Both columns have 56,402 non-null entries.
   * **Unique Values**:
     + Questions: 3,234 unique questions.
     + Answers: 54,726 unique answers.
   * **Most Frequent Question**: "Would Hillary Clinton have made a better President than Donald Trump?"
     + Frequency: 106 times.
   * **Most Frequent Answer**: "No"
     + Frequency: 89 times.

**Data Cleaning Process**

1. **Removing Duplicates**:
   * Identified 1,220 duplicate rows in the dataset.
   * After removing duplicates, the dataset is reduced to 55,182 rows.
2. **Text Cleaning**:
   * **HTML Tags**: Removal of any HTML tags to ensure clean text.
   * **URLs**: Stripping out URLs to prevent irrelevant content.
   * **White Spaces**: Eliminating leading, trailing, and excessive white spaces.
   * **Special Characters**: Removing special characters to standardize the text format.

**Post-Cleaning Dataset**

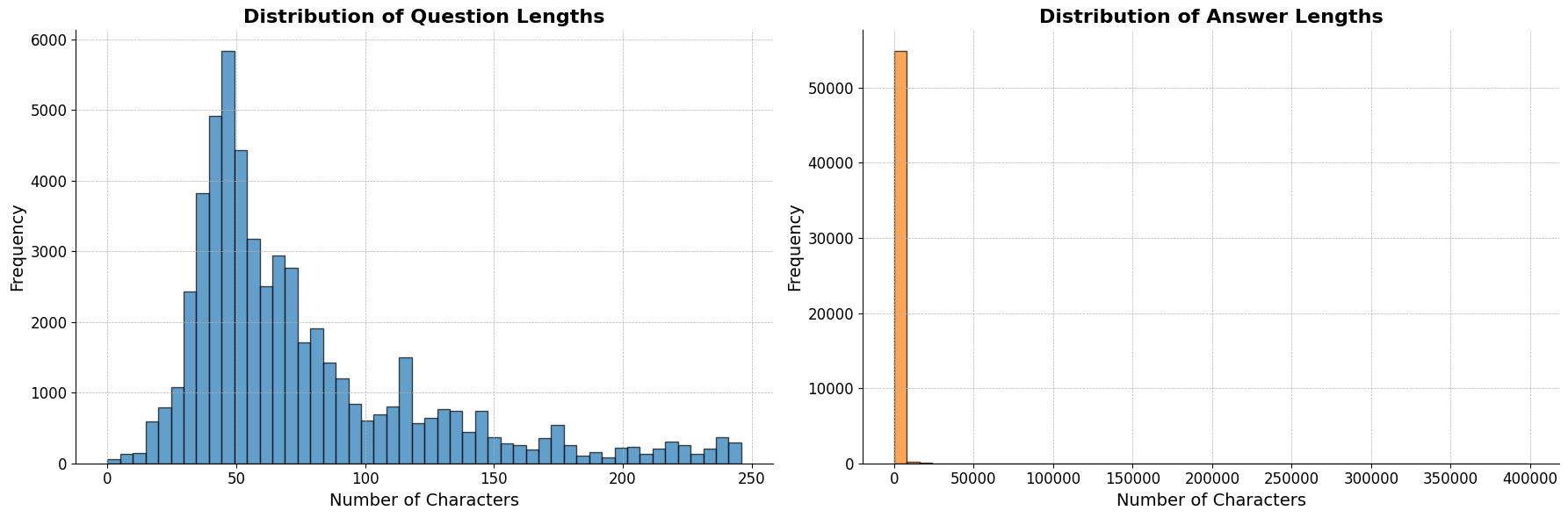
* The dataset, after cleaning and removing duplicates, contains 55,182 rows and 2 columns.

**Visualization and Analysis:**

To understand the data better, visualization techniques could include:

**1. Distribution of Question-and-Answer Lengths**

To understand the variability and distribution of lengths for both questions and answers in the dataset, we can plot histograms showing the number of characters in each question and answer.



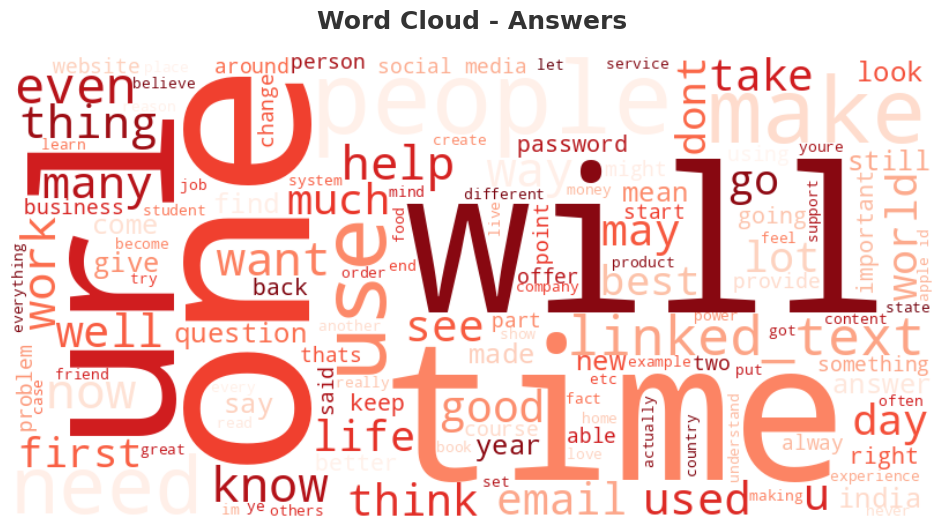
* This visualization helps in understanding the typical length of questions and answers, which is important for model training and optimization.

**2. Word Cloud for Questions and Answers**

Word clouds are a visual representation of text data where the size of each word indicates its frequency or importance.

* **Word Clouds** are useful for quickly identifying the most common words in a text corpus.
* By creating separate word clouds for questions and answers, we can highlight the key terms and topics frequently discussed.

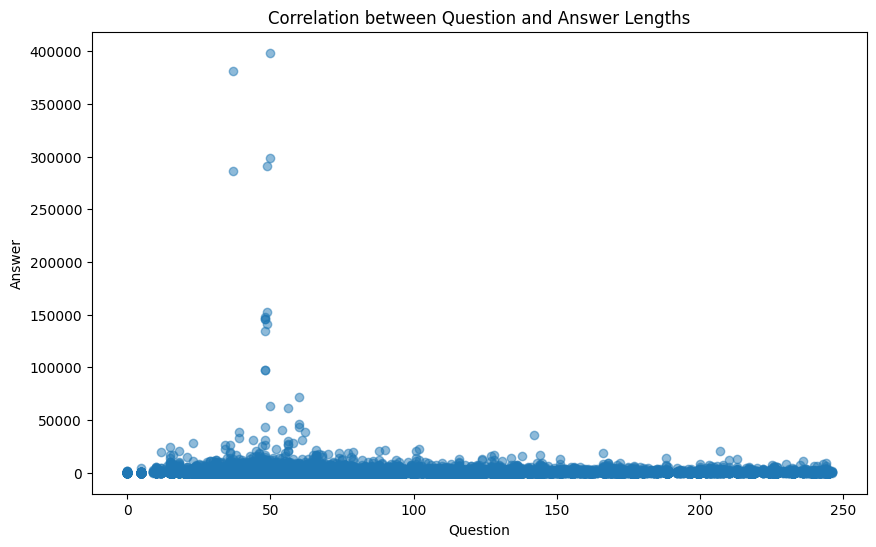




**3. Correlation Between Question-and-Answer Lengths**

To explore the relationship between the lengths of questions and their corresponding answers, we can use a scatter plot to visualize this correlation.

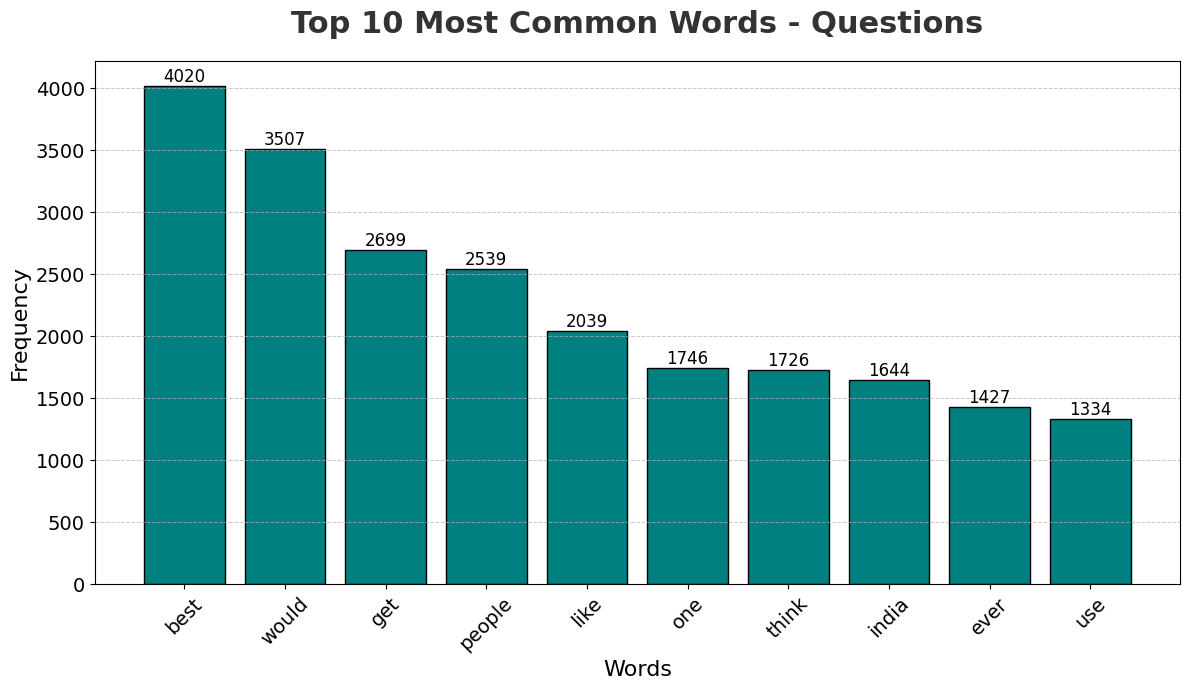
* **Scatter Plots** display the relationship between two numerical variables. Each point represents a pair of values (question length, answer length).
* By plotting these lengths, we can observe whether there is any correlation, such as longer questions leading to longer answers.

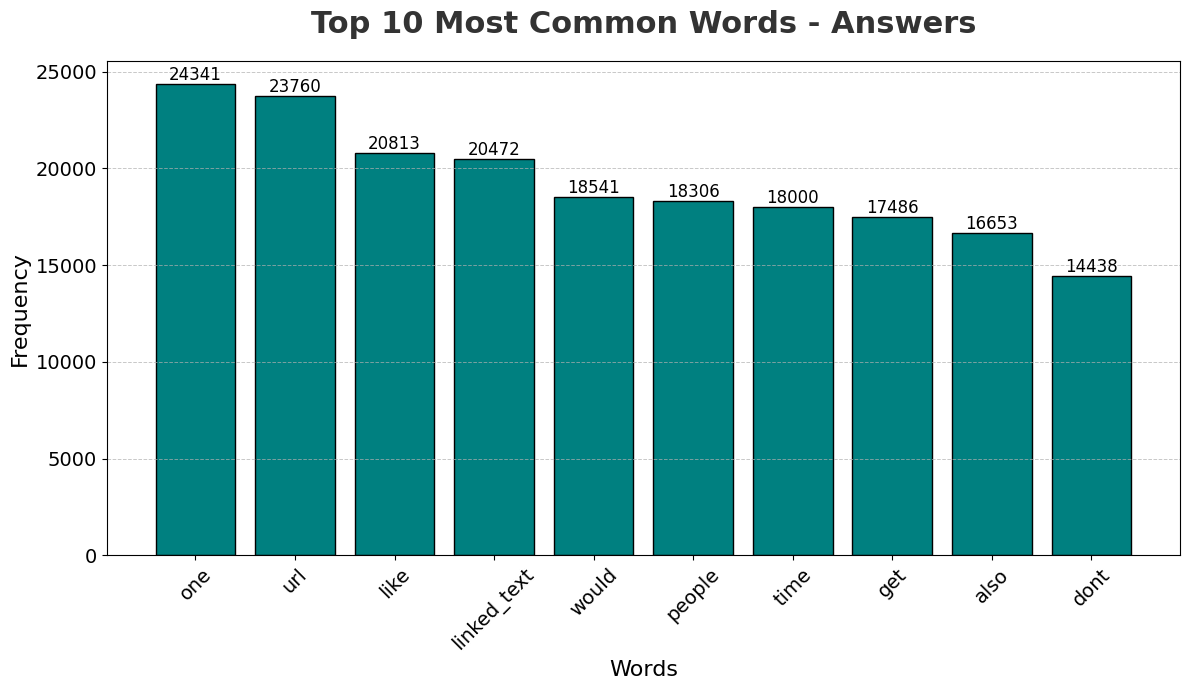


**4. Top 10 Most Common Words in Questions and Answers**

Identifying the most common words in questions and answers provides insights into the main topics and themes in the dataset.

* **Frequency Plots** are used to display the most common words and their frequencies.
* By plotting the top 10 most common words, we can see which terms appear most frequently in the dataset.





These visualizations provide valuable insights into the structure and content of the dataset, aiding in better understanding and preparation for fine-tuning QA-chatbots. The images for these visualizations will be added later.

### Feature Engineering

### Combining Questions and Answers for Context and Further Classification

### Creating Context Column

To enhance the dataset and provide more context for each question-answer pair, we combined the questions and answers into a single 'context' column. This approach helps in better understanding and processing the interactions between questions and answers.

**Implementation**:

* We created a new column 'context' by concatenating the question and answer with a separator '[SEP]' to clearly distinguish between the two

**Classifying Question Types**

To gain further insights into the nature of the questions, we classified each question based on its type. This classification helps in understanding the distribution of question types and can be valuable for model training and evaluation.

**Explanation**:

* We defined a function classify\_question\_type that classifies questions into categories such as 'Who,' 'What,' 'Where,' 'When,' 'Why,' 'How,' and 'Other' based on the starting word of each question.
* This function was applied to create a new column 'question\_type.'

**Encoding Question Types**

To make the 'question\_type' column suitable for machine learning models, we encoded these categories using label encoding. This converts the categorical data into numerical format, which is required for most machine learning algorithms.

**Implementation**:

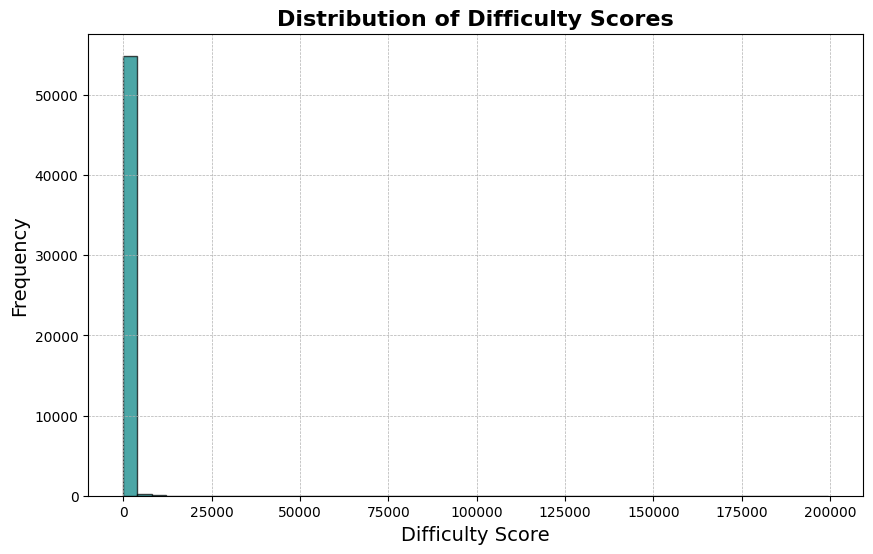
* Used LabelEncoder from sklearn.preprocessing to encode the 'question\_type' column into numerical values and created a new column 'question\_type\_encoded.'

**Creating a Difficulty Score**

We introduced a 'difficulty\_score' based on the lengths of the questions and answers. This score can provide insights into the complexity of each question-answer pair and can be used to assess model performance on varying difficulty levels.

**Explanation**:

* The difficulty score is calculated as the average of the lengths of the question and the answer.
* This score helps in identifying pairs that are longer and potentially more complex, providing a quantitative measure of difficulty.



**Fine-Tuning GPT Model:**

Based on the processed dataset, we plan to fine-tune three models: T5, BERT, and GPT. Each of these models has unique strengths and can contribute to building a robust QA-chatbot.

**Models**:

1. **T5 (Text-to-Text Transfer Transformer)**:
   * A versatile model that converts all NLP tasks into a text-to-text format.
   * Suitable for generating responses in the context of question-answering.
2. **BERT (Bidirectional Encoder Representations from Transformers)**:
   * A model that excels in understanding the context of words in a sentence.
   * Ideal for tasks requiring deep understanding of language semantics.
3. **GPT (Generative Pre-trained Transformer)**:
   * A powerful model for generating human-like text based on given prompts.
   * Effective for generating answers and engaging in conversational AI tasks.

#### Overview

In this section, we describe the process of fine-tuning a GPT-2 model on the question-answering dataset. The steps include loading the dataset, preprocessing the data, setting up the model and tokenizer, and training the model using the Hugging Face transformers library. We focus on the key details such as hyperparameters and training loss.

#### Step-by-Step Explanation

**1. Load the Datase**

**2. Load Pre-trained Model and Tokenizer**

Next, we load the pre-trained GPT-2 model and its corresponding tokenizer from the Hugging Face model hub.

**3. Set Padding Token**

Since the GPT-2 tokenizer does not have a padding token by default, we add one and adjust the model embeddings to account for the new token.

**4. Prepare the Dataset**

We preprocess the dataset by tokenizing the 'context' column, ensuring that the text is truncated and padded to a maximum length of 512 tokens.

**5. Create Data Collator**

We use a data collator to handle the padding of sequences during training. For language modelling, we disable masked language modelling (MLM) as GPT-2 is not an MLM model.

**6. Set Training Arguments**

We define the training arguments, including the output directory, number of epochs, batch size, and logging configuration for Tensor Board.

**Details**:

* **Output Directory**: "./gpt2\_qa\_model"
* **Overwrite Output Directory**: True
* **Number of Epochs**: 3
* **Batch Size**: 4 per device
* **Save Steps**: 10,000
* **Save Total Limit**: 2
* **Logging Directory**: './GPT-logs'
* **Logging Steps**: 500
* **Report to**: Tensor Board

**7. Initialize Trainer**

We initialize the Trainer class with the model, training arguments, data collator, and training dataset.

**8. Train the Model**

We train the model using the train method of the Trainer class. The training progress and metrics are logged to TensorBoard.

**Training Loss Plots**:

* During training, the model's performance is monitored by plotting the training loss at regular intervals.
* These plots help in understanding the convergence of the model and identifying any issues such as overfitting or underfitting.

**9. Save the Model**

Finally, we save the fine-tuned model to the specified directory.

**Fine-Tuning BERT Model**

#### Overview

This section outlines the fine-tuning process for a BERT model on a question-answering dataset. We will cover the steps involved, the hyperparameters used, and how the training loss is monitored.

#### Step-by-Step Process

1. **Load the Dataset**
2. Load Pre-trained Model and Tokenizer
3. **Prepare the Dataset**
4. **Set Up TensorBoard Writer**
5. **Training Loop**
6. **Save the Model and Logs**
7. **Hyperparameters**

* **Model**: bert-base-uncased
* **Learning Rate**: 5e-5
* **Number of Epochs**: 3
* **Batch Size**: 8
* **Maximum Sequence Length**: 512

**Training Loss Curves**

* **Visualization**: The training loss is monitored and visualized using TensorBoard.
* **Purpose**: The loss curves display the model's performance over epochs, indicating how well the model is learning.
* **Interpretation**: A decreasing loss curve suggests that the model is effectively learning from the data, while fluctuations or increases might indicate issues such as overfitting or learning rate problems.

**Fine-Tuning T5 Model**

#### Overview

This section details the fine-tuning process for a T5 model on a question-answering dataset. It includes data preparation, model setup, training, and monitoring of performance.

#### Step-by-Step Process

1. **Load the Dataset**
2. **Load Pre-trained Model and Tokenizer**
3. **Prepare the Dataset**
4. **Initialize TensorBoard Writer**
5. **Training Loop**
6. **Save the Model and Logs**
7. Hyperparameters

 **Model**: t5-small

 **Learning Rate**: 5e-5

 **Number of Epochs**: 3

 **Batch Size**: 8

 **Maximum Sequence Length**: 512

**Training Loss Curves**

* **Visualization**: Training loss is visualized using a performance graph created with matplotlib.
* **Purpose**: The graph displays the average loss per epoch, providing insights into the model's learning progress over time.
* **Interpretation**: A decreasing trend in the loss graph indicates effective learning. The graph helps in assessing whether the model is improving and whether additional training or adjustments are needed.

### Evaluation of Model Performance

Bottom of Form