Dataset Understanding

Structure of the Dataset:

The dataset is loaded from a CSV file named 'train.csv' using the Pandas library. The dataset consists of the following columns:

- `Question`: Contains input questions for the model.
- `Answer`: Contains corresponding answers or responses to the questions.
- `qtype`: Represents the type of the question.

Data Preprocessing

Text Cleaning:

- Special characters are not explicitly removed in the provided code.
- Text is lowercased to ensure uniformity.
- Tokenization is performed on the questions.

Handling Missing Values and Duplicates:

- Missing values are checked using `df.isnull().sum()` to identify any NaN values.
- Missing values in numerical columns are filled with the mean using `df.fillna(df.mean(), inplace=True).`
- Temporary columns created for analysis are dropped using `df.drop(['question_length', 'answer_length'], axis=1, inplace=True).`

Splitting Data:

- The dataset is split into training, validation, and test sets using `train_test_split` from sklearn.

Exploratory Data Analysis (EDA)

Question Type Distribution:

- The code identifies the top 10 question types and visualizes their distribution using a countplot.

Question and Answer Lengths:

- The distribution of question lengths is explored using a histogram.
- The relationship between question types and lengths is visualized using boxplots.

Word Counts:

- Word counts in questions and answers are visualized using histograms.
- The correlation between question word count and answer word count is presented in a heatmap.

N-grams:

- Top unigrams in questions and top bigrams in answers are visualized using bar plots.

My Approach:

- Initial dataset analysis revealed the task's nature as a generative one rather than context-only, leading me to consider the Retrieval-Augmented Generative (RAG) approach.
- Developed a basic retriever using GPT-2 as a baseline model, trained on all dataset answers as a paragraph for answer generation.
- Effective for simple queries but struggled with complex ones due to limited dataset for creating a robust dense retrieval model.
- Explored alternative approaches, leading to the identification of the Google/flan-T5 model, a family of large language models (LLMs) with enhanced performance over T5 counterparts.
- Key features of FLAN-T5 include improved performance, diverse task fine-tuning, multiple model sizes, and strong few-shot learning capabilities.
- Conducted fine-tuning on FLAN-T5 using various hyperparameters (e.g., learning rate, batch size, epochs), limited to three epochs due to computational constraints.
- Initially used ROUGE score metrics for evaluation but later found it unsuitable for assessing model performance in this context.
- Explored another model, GeneZC/MiniChat-1.5-3B, a language model distilled and fine-tuned from an adapted version of LLaMA2-7B.

- Noteworthy performance against various 3B and even some 7B chat models, despite its substantial size (3 billion parameters).
- Computationally unfeasible to fine-tune on a large dataset, so employed the peft technique for fine-tuning on a reduced number of data points.
- Despite limited training data, GeneZC/MiniChat-1.5-3B exhibited commendable results, surpassing the performance of previous models and the baseline model after fine-tuning.

Evaluation Metrics:

- Tried different metrics but human evaluation comes out to best in these type of cases.
- And on the basis of human evaluation **MiniChat-1.5-3B** this model turns out to be best. However, improvement can be done on other model by increasing computational power.

Project Documentation: Question Answering and Model Training

Overview:

This Assignment aims to address question answering tasks through a combination of data analysis, traditional machine learning, and transformer-based deep learning approaches. The code encompasses various models, including GPT-2, T5, and MiniChat-1.5-3B, to handle different aspects of the question-answering process.

Code Structure:

The code is structured into several sections, each addressing specific tasks and experiments. The main sections include:

Installations and Imports:

- Install and import necessary libraries for evaluation and visualization.

Data Exploration and Analysis:

- Import libraries, load the dataset, and explore the data.
- Visualize question types, lengths, and relationships.
- Handle missing values and create additional features.

N-gram Analysis:

- Utilize CountVectorizer to analyze top unigrams and bigrams in questions and answers.

Simple Retrieval and RAG Model:

- Implement a basic keyword-based retrieval model.
- Develop a more sophisticated Retrieval-Augmented Generation (RAG) model using GPT-2.

T5 Model for Question Answering:

- Train a T5 model for question answering using the Hugging Face Transformers library.

Finetuning a MiniChat Model:

- Prepare a dataset and train a Seq2Seq model for MiniChat.

Dataset Preparation and Training Configurations:

- Prepare datasets for training.
- Experiment with various training configurations.

Various Configurations and Experiments:

- Experiment with GPU usage, gradient checkpointing, and lora configurations.

Additional Training and Exploration:

- Iterate through training and experimentation based on ongoing findings.

Algorithms Used:

The project employs a combination of traditional machine learning and transformer-based deep learning algorithms:

- CountVectorizer (N-gram Analysis):
    - Utilized for analyzing the frequency of unigrams and bigrams in questions and answers.
- GPT-2 and T5 (Retrieval and Generation):
    - GPT-2 and T5 models are used for both retrieval and generation tasks in question answering.
- Seq2Seq (MiniChat Model):
    - A Seq2Seq model is trained for a specific task in MiniChat.

Reasoning and Choices:

Model Selection:

- GPT-2 and T5 were chosen for their effectiveness in natural language understanding and generation.

Experimentation with Retrieval and Generation:

- Combining retrieval and generation models (RAG model) aims to enhance the response generation process.

Training Configurations:

- Iterative experimentation with batch sizes, learning rates, and training epochs to find optimal configurations.

Documentation and Comments:

- Extensive comments and documentation have been added to enhance code readability and understanding.

Findings:

The Assignment has yielded insights into various aspects, including dataset characteristics, model performance, and the impact of different configurations on training outcomes. Ongoing experimentation and documentation have facilitated a systematic approach to addressing challenges and refining models.

This documentation provides a high-level overview of the project, highlighting key aspects, choices, and findings. For a more detailed understanding of each section and its implementation, refer to the corresponding code comments and documentation within the codebase.