Name: Yashwardhan Chaudhuri

SHL Labs RE Assignment

Introduction:

In this project, synthetic review texts were generated for a specific category utilizing the llama2 7b model. Due to computational constraints, synthetic reviews were generated for only five products within the category. llama2 was chosen for its open-source nature and cost-effectiveness. The data cleaning process involved loading two datasets, namely `product_asin.csv` and `reviews_supplements.csv`, into pandas DataFrames. Missing values were handled, and review texts were processed by adding a prefix. Each product's reviews were then grouped by a common identifier and concatenated. Additionally, non-empty category columns were concatenated to create a streamlined column. The resulting dataset was merged to ensure comprehensive data was ready for analysis.

Cosine Similarity Metric and Prompting Method:

Cosine similarity was chosen as the metric to evaluate the similarity between generated and existing reviews due to its effectiveness in measuring the similarity of textual data. It calculates the cosine of the angle between two vectors, representing the texts' embeddings, and provides a measure of similarity irrespective of the vector's magnitude. For prompting the llama2 model, the product name, previous review texts, and product category were included. This prompt provided contextual information necessary for generating relevant and coherent synthetic reviews to each product. The prompt ensured that the generated reviews maintained relevance to the product's domain and incorporated insights from existing user feedback and coherence of the generated text.

Please find the details of the models, metrics, and approaches that were tried for the following assignment below.

Data Cleaning and Preparation:

The data cleaning process involved loading `product_asin.csv` and `reviews_supplements.csv` into pandas DataFrames. Missing values were handled by replacing them with empty strings.

Each review text was prepended with a standardized phrase to delineate individual reviews. Reviews were grouped by `parent_asin` and concatenated for each product. A function concatenated non-empty category columns, producing a streamlined `concatenated` column. Key columns—`title`, `parent_asin`, and `concatenated`—were selected. A dictionary was created with product titles as keys, linking them to `parent_asin` and `concatenated` values. Reviews and product data were merged on `parent_asin`. This rigorous process ensures a comprehensive dataset ready for analysis.

Preparation for llama2 model:

Quantization Configuration:

- Defines a quantization configuration (bnb_config) for model weights and activations, specifying 4-bit quantization with float16 computation. This is beneficial for reducing model size and inference latency while maintaining performance.

Tokenizer Initialization:

- Initializes an evaluation tokenizer (eval_tokenizer) from a pretrained model (model_name) for text generation. The tokenizer configuration includes adding a beginning-of-sequence (BOS) token and setting the pad token to the end-of-sequence (EOS) token.

Text Generation Function:

- Defines a text generation function (gen) that takes a prompt (p) as input and generates text using the provided model. It utilizes the generate method of the model with specified parameters such as maximum token length, sampling strategy, temperature, and number of beams.

Model Initialization:

- Initializes a language model (model) for causal text generation using a pretrained model (MODEL_NAME). Custom configurations include using float16 data type for model weights, enabling trust in remote code execution, and specifying automatic device placement.

Generation Configuration:

- Defines a generation configuration (generation_config) with parameters such as maximum new tokens, temperature, top-p sampling, and repetition penalty. These settings control the diversity and quality of generated text.

Text Generation Pipeline:

- Sets up a text generation pipeline (text_pipeline) using the initialized model and tokenizer. This pipeline is configured to return full generated text, ensuring complete output sequences.

HuggingFace Pipeline:

- Combines the text generation pipeline (text_pipeline) into a HuggingFacePipeline (llm).. This pipeline encapsulates the entire text generation process, allowing generation of text by passing prompts.

Data Cleaning and Preparation:

Tokenizer and Model Initialization:

- Initializes a BERT tokenizer (tokenizer) and a BERT model (model) from the 'bert-base-uncased' pre-trained model. If the tokenizer doesn't have a padding token, it adds one and resizes the token embeddings accordingly.

BERT Embedding Extraction:

- Defines a function (get_bert_embedding) to obtain the BERT embeddings for a given text input. It tokenizes the text using the tokenizer, feeds it to the BERT model, and extracts the mean of the last hidden state along the token dimension.

Similarity Calculation:

- Implements a function (calculate_similarity) to compute the cosine similarity between two texts based on their BERT embeddings. It utilizes the BERT embeddings obtained from the previous step and computes their cosine similarity.

Text Generation with llama2:

- Iterates over a set of products (5 in this case) to generate similar reviews using the llama2 approach.
- For each product, it retrieves the product name, existing user reviews, and keywords.
- It then iterates to generate new reviews using llama2, ensuring that the generated reviews are similar to the existing ones while being diverse and relevant.
- The similarity threshold is set between 0.67 and 0.90 to filter out overly similar or dissimilar reviews.
- Finally, it constructs a dictionary (final_llama2_results) where each product is associated with a set of generated reviews.

Output Storage:

Iteration and Data Collection:

- Iterates over the final llama2 results stored in final_llama2_results.
- Extracts the product names (i), along with the first and second generated reviews (final_llama2_results[i][0] and final_llama2_results[i][1]).
- Appends these values to separate lists (col1, col2, and col3).

Construction of Final DataFrame:

- Constructs a dictionary (final_llama2) where keys represent column names ("name", "rev1", and "rev2"), and values are the lists containing the corresponding data (col1, col2, and col3).

DataFrame Conversion and Export:

- Converts the dictionary final_llama2 into a pandas DataFrame using pd.DataFrame().
- Writes the DataFrame to a CSV file named "synth_reviews.csv" using the to_csv() method.

How diversity of texts was managed?

For a particular product P, We only use previous reviews given to generate context-aware data. E.g. for the product Allegra, we use the earlier given 8 prompts provided as the previous context to generate new models. I also offer keywords and product names for refinement, although not much improvement was seen. Furthermore, just like a lower threshold, cosine similarity also had an upper threshold so that the reviews did not become too similar. In practice, this approach yielded promising results after some hyperparameter tuning.

How was the efficacy of synthetic datasets managed?

We use a cosine similarity-centric approach where BERT embedding of synthetic and ground_truth reviews is calculated. We use cosine similarity particularly because a score like the Jaccard Index would not be able to understand review lengths or the internal context of the generated review. The TF-IDF comparison was also tried but fetched poor results.

How do we ensure the synthetic dataset one generates is inspired by a source dataset but not a replica?

We ensure that generated reviews are not exact replicas by keeping a relatively high temperature for optimism and cosine similarity for ‘nearness’ to ground truth reviews. Cosine similarity of 0.95 or more shows a datapoint tending towards a replica.

What were the top challenges in solving this problem statement?

1. Creating suitable prompts
2. Understanding what data to send to prompt
3. Tuning parameters for the appropriate answer

Model Evaluation and Selection

Three models were explored for text generation: Phi-3 Mini-128K-Instruct, Phi-2, and llama2-7b. The Phi-3 Mini-128K-Instruct model, with 3.8 billion parameters, demonstrated moderate to poor results, exhibiting limited adaptability to various prompts. Phi-2, a Transformer with 2.7 billion parameters, showcased better than the Phi3 model but still not at par results with larger models. The Phi family was selected for its computational efficiency. llama2-7b, part of the llama2 family of models was used as the final model for this project, offering promising capabilities.

Evaluation Metrics

Several metrics were employed to assess the quality of generated text. While the Jaccard Index and cosine similarity with TF-IDF were tested, cosine similarity with BERT embeddings delivered the most reliable results. Leveraging BERT embeddings enabled the measurement of semantic similarity between texts, ensuring more accurate evaluations.

Prompting Strategies

Various prompting methods were explored to guide the text-generation process. The most effective approach involved including the product name, category, and reviews in the prompt. This method consistently produced outputs with length awareness and context relevance. Conversely, prompts containing only reviews exhibited moderate performance, occasionally resulting in erroneous outputs. Attempts to incorporate categories in the prompt consistently failed to yield meaningful outputs in the majority of cases.