Title: Optimization Report for Text-to-LLM Algorithm Implementation

-Tushar Goyal, Varshil Nakum, Parul, Ajay Sanwal

Executive Summary:

This report outlines the optimization strategies employed in the text-to-LLM algorithm implemented in Part 1. The algorithm utilizes hashing and a minheap (implemented as a maxheap using negation) to store corpus words and sort scores in strictly increasing order. Additionally, prompt engineering techniques have been incorporated for efficient processing. This report details these optimizations and outlines a structured approach for handling data size constraints.

Algorithm implementation to feed text to LLM:

- We are storing the corpus in a vector of tokenised sentences.
- We are traversing over each token and checking if it belongs to the tokenised query, if it does, we are incrementing the fp of that word for that paragraph.
- We are creating a dummy hashmap to store helper words or common words like : if, had, of, explain, with etc. We are marking their fp = 0 so that they donot contribute to the Parascore.

Scoring a word of a Query:

Sp depicts the “rarity” of the word of the Query in the corpus. Mathematically, Sp = 1/(word count in corpus +1), with Sp = 0 if word is absent in the corpus. This is done because lesser the occurrence of the word in the corpus, more important it becomes and it should reflect in the parascore prominently.

- Parascore = ∑fp*(sp)^2. This is because we are to give more weight to the “score” of a word rather than it’s frequency in the paragraph. Since if a word like “movement” and “Gandhi” are to clash, we want the rarity of “movement” to compensate the frequency of “Gandhi” in the paragraph. After examining the scores for several queries we concluded that squaring the score term allows this compensation.
-

Optimization Techniques:

1. Use of Data Structures:

Hashing for Dictionary Construction:

Hashing is employed to efficiently build a dictionary for storing corpus words. This allows for quick retrieval and lookup operations.

Minheap (Maxheap using Negation):

A minheap is utilized to store scores in strictly increasing order. By multiplying values with -1, the minheap effectively functions as a maxheap, allowing for easy retrieval of the highest scores.

2. Prompt Engineering Techniques:

Structured Prompt Format:

The algorithm processes input text in a specific format: `"above given is a question. You have to answer it using ONLY the information in the passage below my me. The passage is from a corpus about mahatma gandhi. So make out how to contruct a fulfulling and logical answer from the limited information that you have"`

We want chatgpt to understand 2 major things:

- The above given statement is a question to be answered.
- The answer should be constructed strictly from the knowledge of the given passage.

  Also it should understand that the answer should be to the point and fulfil the question. Basically, it should not be a garbled sentence out of the given passage.

Conclusion:

The implemented algorithm demonstrates an effective combination of data structures and prompt engineering techniques to optimize the text-to-LLM process. The use of hashing and a minheap efficiently manages corpus words and scores, while prompt structuring ensures meaningful and appropriately sized responses. These optimizations contribute to the algorithm's ability to handle varying data sizes while maintaining response quality. Ongoing

monitoring and adjustment may be necessary to adapt to changing requirements or data characteristics.