AIAA 5023 - Long Writer with Large Language Model

Linjie Qiu

1 Introduction

This assignment explores the potential of using a large language model (GLM-4-Flash) to generate coherent, high-quality long texts despite token limits. The main challenge is to overcome token constraints while maintaining the text's relevance and readability. The selected topic for this task is "Application of Nanotechnology in Medicine."

The approach involves utilizing planning and scheduling techniques to split the task into manageable segments and generate content piece by piece. The final deliverable is a 10,000+ word text generated using multiple API calls.

2 Implementation Details

2.1 Code Structure

The implementation consists of two primary Python scripts:

Assignment3.py:

Handles the main process, including API calls, text planning, generation, and saving/loading results. Core functions include: ***planning():*** Create the structure for the long text and iteratively generate content based on the structure. ***text_generation():*** Communicates with the GLM-4-Flash API to generate responses based on prompts. ***save_files()*** and ***load_files():*** Save and retrieve generated content.

utils.py:

Provides utilities for formatting prompts, extracting structured data, and calculating word counts. Core functions include: ***complete_Planning_Prompt():*** Generates a JSON-formatted prompt to create an outline. ***complete_Writing_Prompt():*** Generates a prompt for writing detailed paragraphs based on the outline. ***extract_json_from_text():*** Extracts JSON objects from the text response.

2.2 Workflow

Planning: The program begins by generating an outline of the text using the planning() function. Each section is defined with a specific idea and word count. **Segmented Generation:** Using the generated outline, the program creates individual paragraphs via iterative API calls. **Saving Results:** The content is saved to a file *(50022955.txt)* for evaluation and further processing.

3 Challenges and Solutions

3.1 Token Limitations

The GLM-4-Flash model has a token limit of 4096, making it impossible to generate lengthy text in a single call.

Solution: The text is divided into smaller sections with structured prompts, ensuring coherence across multiple API calls.

3.2 Maintaining Coherence

Ensuring logical flow and relevance between paragraphs when generated independently.

Solution: The planning process creates an outline that guides the content generation, while prompts are carefully designed to include contextual information.

3.3 API Efficiency

Reducing the number of API calls to improve efficiency.

Solution: Each API call generates a complete paragraph with precise word counts, avoiding excessive back-and-forth adjustments.

4 Novelty of approach

4.1 Multi-Stage Planning for Long Texts

The implementation uses a hierarchical structure to generate long texts. By first dividing the overall content into sections through a planning phase, the program ensures logical progression and thematic consistency across paragraphs. This structured methodology is particularly effective for managing large content requirements.

4.2 Seamless Integration of Pre- and Post-Processing

Custom-designed prompts leverage detailed instructions and JSON formatting to ensure output consistency. These prompts also align with specific requirements for text length and structure. Utility functions handle extracting, parsing, and reformatting generated outputs. The modular ***extract_json_from_text()*** function ensures clean and usable data even when the API response is not perfectly formatted.

4.3 Adaptive Paragraph Length Allocation

Instead of evenly distributing word counts, the solution dynamically adjusts paragraph lengths based on the complexity of each topic. This adds flexibility, as certain sections may naturally require more detail.

4.4 API Efficiency through Minimal Call Design

The method minimizes redundant API calls by generating entire paragraphs in one call. This is made possible through careful planning, ensuring that each prompt carries enough context for the model to produce coherent and standalone outputs.

5 Evaluation

5.1 Statistics

The paragraph generated in *(50022955.txt)* meets the assignment requirements.

Text Length: 11200

Number of API Calls: 25

The text was assessed using a quality coherence, relevance, fluency, and readability rubric. Minor edits were made to ensure formatting consistency. The scores evaluated by the LLM are as follows: coherence: 9, relevance: 9, fluency: 9, readability: 8.