A1. Background on you/your team

- **Competition Name**: LLM 20 Questions
- **Team Name**: yukky_maru
- **Private Leaderboard Score**: 1159.5
- **Private Leaderboard Place**: 5th

Name: yukky_maru
Location: "Since 'Model_Summary_yukky_maru.md' is publicly available as open-source material, I will refrain from including it here. (This information is included in the 'Model_Summary_yukky_maru.md' submitted in the 'File Drop' section under 'Team.')" **Email**: yukimidaifuku715@gmail.com

A2. Background on you/your team

"Since 'Model_Summary_yukky_maru.md' is publicly available as open-source material, I will refrain from including it here. (This information is included in the 'Model_Summary_yukky_maru.md' submitted in the 'File Drop' section under 'Team.')"

A3. Summary

Objects in this world have two aspects: the aspect of a name and the aspect of the attributes the object contains. When searching for a keyword, if all objects are assumed to be equally likely to be chosen, a binary search tree minimizes the expected number of operations. However, if the probabilities differ, a Huffman search tree minimizes the expected number of operations. However, since it is nearly impossible to perform a binary search on the keyword space based on the attributes contained by the objects, I focused solely on the aspect of the name in this strategy. I also assumed the probability of a word being chosen as word_frequency. **Therefore, for the questioning strategy, I used a binary search tree ordered alphabetically for the names of the objects in the early stages, and a Huffman tree-like method based on word_frequency in the later stages. For the answering strategy, I employed painstakingly hard-coded responses, as well as VAGOsolutions' Llama 3.1 SauerkrautLM 8b Instruct model. For the inference strategy, I used word_frequency to efficiently infer the keyword.**

image

A4. Feature Selection / Engineering

The difference between myself and other players who used binary search lies in the method of creating the keyword list and utilizing word frequency. Therefore, it can be concluded that word frequency functioned as a powerful feature within the keyword space.

The method for creating the keyword list was based on ensuring that keyword.py mainly represented actual objects, so I limited the collected keywords to objects. Specifically, I extracted only nouns from datasets such as Conceptual Captions, Coco Caption image caption datasets, and Amazon reviews, and then used Gemini Flash to process these nouns in batches of 100, filtering the necessary nouns. Additionally, I added nouns from WordNet to the list as well.

A5. Training Method(s)

For this competition, we used the pre-trained Llama 3.1 SauerkrautLM 8b Instruct model without any fine-tuning. We selected this model because of its strong ability to follow instructions and adhere to the provided prompt, as indicated by its **high IFF score**. **Fine-tuning is generally adept at altering response formats but is quite poor at knowledge insertion. Therefore, we believed that even if we fine-tuned the model, the improvement in its performance would be limited.** If we were to consider any adjustments, it would be more logical to construct an architecture similar to RAG, where we insert documents about the secret keyword into the prompt. **However, considering the high performance of llama3.1 from the outset, we decided it was sufficient to achieve high performance without fine-tuning and chose to allocate time to the keyword list in the questioning strategy discussed later.** Moreover, popular public notebooks employed a strategy to create a truth table for keywords and questions using vllm with llama3. Thus, we concluded that **the key to ranking high in this competition, with its unique evaluation method of strong response type, was not necessarily to provide correct answers to attribute-related questions but rather to give the same answers as llama3**. This was the second reason for not performing fine-tuning.

A6. Interesting Findings

In this competition, the top competitors were mostly using binary search-based agents (binary search decision trees), so this aspect did not serve as a point of differentiation from the other competitors. **What clearly set me apart from the others was the way I created the keyword list and the use of word frequency for making guesses, as well as the fact that I switched from a binary search decision tree to a Huffman decision tree strategy midway through.** Therefore, I believe these two points were the most important tricks and the key factors that differentiated me from the other competitors. Additionally, I don't think there were any other competitors who manually hardcoded responses for the answering agent at my level, so I believe that configuring the agent to provide effective responses for narrowing down object names other than the Alpha Agent was also beneficial, especially in the early stages of the game.

A7. Simple Features and Methods

In this specific competition with a limited number of participants, we manually hard-coded responses to various questions about the keyword structure using regular expressions. However, this approach is not scalable to a larger, more diverse set of participants.

To simplify the model and reduce manual effort, a method of outputting executable Python code for questions about the keyword's letter structure, without using hard coding, could be considered. While this might be challenging with the llama3.1-8B model, it could be possible with a slightly larger model or models like GPT-4o mini or Gemini Flash.

A8. Model Execution Time

As no fine-tuning was performed, the training time for our model was 0. With the max_new_tokens parameter set to 3 and the prompt designed to elicit only "yes" or "no" responses, the inference time for generating a response to a single question was typically a few seconds.

A9. References

The following resources were used in the development of this solution:

- **LLM Model**:

  - Llama 3.1 SauerkrautLM 8b Instruct model

- **Keyword List**:
  - conceptual-captions
  - coco-captions
  - AmazonReviews
  - wordnet
  - gemini
  - wordfreq