Order Dependency Problem in Large Language Models

The order dependency problem refers to the sensitivity of large language models like GPT, Claude, LLaMA, Mistral, etc., where the ordering of the input tokens has a direct effect on the response from the LLM. The ordering comes into play because the model predicts based on the sequences it has been given. The position can drastically change the response. This also has ties into the term hallucinations, where the model predicts a word and or phrase that sends the model off into another direction without intuitive intelligence to recover or stay on track.

The order dependency problem has already been proven in many real word scenarios beyond multiple choice questions. Here are a few examples:

- **Financial advice:** Users may input data in varying order. If the model places higher weight on earlier entries without properly considering subsequent details like wanting to allocate 50% of capital in US based stocks it could produce flawed investment advice.
- **Medical diagnosis:** A condition might base its output heavily on early symptoms mentioned by the patient, while potentially disregarding later critical symptoms, leading to an incorrect diagnosis.
- **Legal Summarizing:** An LLM might overemphasize clauses that appear early in the text while neglecting important details that come later. This could lead to summaries that are incomplete and or inaccurate.
- **Support chatbots:** If a user provides important information later in a conversation, such as being unable to log in on their mobile device but stating much later on in the conversation that they can log in on their laptop, the model might prioritize earlier context and offer troubleshooting tips for resetting the password, rather than addressing the actual issue.

Through my own experience and research, I have explored several strategies to address the order dependency problem, including:

- **Break up the task:** I have found it effective to break up the request into multiple parts, with each part focused on a single task. While this increases costs due to multiple requests, it results in much more accurate responses. In some cases, this method may not be as effective, but for multiple-choice questions, it has significantly improved accuracy to an acceptable level.
- **Reweighting of Input Data:** This approach involves adjusting how the model weighs different parts of the input sequence. For example, using keywords like IMPORTANT and EXTREMELY IMPORTANT can adjust those weightings, which will ultimately send the signals to ensure adequate consideration. In addition, there have been use cases where simply adding the phrase “take a deep breath before answering the question” adjusts the weights on certain styles of prompting.
- **Fine tuning/Multishot:** On specific datasets this technique can help the LLM learn the proper balance of context in different applications. By exposing the model to examples where later information is more important, it can learn to adjust its sensitivity to input order.

- **Human in the loop:** In certain use cases, human reviewers can intervene to ensure that the model's output makes sense given the importance of the request. This process is also important when looking to produce enough data for multishot and fine tuning data.