Nishauri 2.0 - Leveraging Retrieval-Augmented Generation for Digital Health

Introduction:

The challenge posed in the Nishauri application development revolves around integrating Retrieval-Augmented Generation (RAG) to create an advanced chatbot system. This initiative aims to harness Generative AI to enhance digital health services, ensuring accurate and timely information dissemination to users/patients.

Understanding of the Case Study:

In the realm of healthcare, ensuring the precision of information shared is of utmost importance. Here, the integration of Retrieval-Augmented Generation (RAG) models emerges as a crucial strategy, acting as a bridge between dynamic technological advancements and the essential need for accurate healthcare information dissemination.

Definition of RAG:

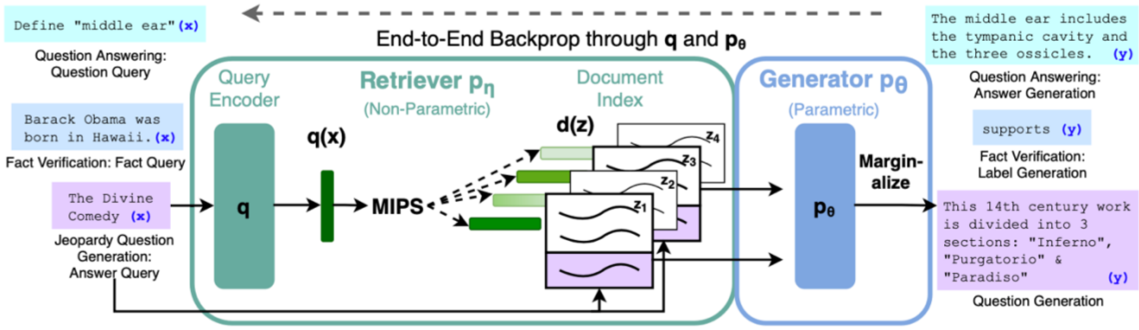
RAG is a sophisticated model combining a parametric (pre-trained large language model) and a non-parametric model. This fusion enables dynamic information retrieval and generation, ensuring the dissemination of accurate information to the users of Nishauri applications

Components of RAG:

1. The Parametric Model:
   Utilizes pre-trained large language models, ensuring robustness and efficiency in generating responses.
   Drawbacks and Advantages:
   While parametric models excel in their predictive capabilities, they are constrained by resource-intensive fine-tuning processes. However, their advantages lie in their adaptability and effectiveness in generating responses.

2. The Non-parametric Model:
   Employs a dynamic retrieval mechanism for acquiring relevant information.
   Drawbacks and Advantages:
   Non-parametric models offer flexibility in retrieving information but may lack the predictive capabilities of parametric models. However, they excel in scalability and adaptability.

Architecture of the Approach we intend to use



Overview of the approach step by step :

We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents zi. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

1. **Combine Pre-trained Retriever and Seq2Seq Model:**
    - Integrate a pre-trained retriever, comprising a Query Encoder and Document Index, with a pre-trained seq2seq model, known as the Generator.
2. **Fine-tune End-to-End:**
    - Implement end-to-end fine-tuning of the combined model to ensure seamless interaction between the retriever and the seq2seq model.
3. **Query Processing with MIPS:**
    - When presented with a query (denoted as x), employ Maximum Inner Product Search (MIPS) to retrieve the top-K relevant documents (denoted as zi).
4. **Prediction Generation:**
    - Treat the retrieved documents (zi) as latent variables.
    - Marginalize over seq2seq predictions considering the varied documents.
    - Generate the final prediction (denoted as y) based on the marginalized seq2seq predictions.

Challenges Encountered:

1. GPU Requirement: Fine-tuning pre-trained large language models necessitates significant GPU resources.
2. Cost Implications: Utilizing state-of-the-art parametric models may incur high API usage costs.

3. Memory Constraints: Effective fine-tuning mandates systems with larger memory pools.

Solutions Implemented:

1. Google Colab Premium: Subscribing to the premium version of Google Colab addresses both memory and GPU constraints efficiently.
2. Trial Period Access: Exploring access options to pre-trained models like Vertex, albeit temporary, offers potential solutions, although not optimal.