InstiGPT

Challenges with Large Datasets in RAG

Data Sparsity:

As the dataset used by InstiGPT is limited to our insti's data, data sparsity could lead to suboptimal performance and increased hallucinations.

Active learning techniques can be employed to select and label the most informative instances for training, reducing data sparsity and improving model performance while minimizing the need for extensive manual annotation.

Overfitting:

Large datasets increase the risk of overfitting (model becomes too specialized in capturing the training data patterns). Regularization techniques, such as dropout or weight decay, can be employed during training to mitigate overfitting and improve generalization.

Also, data augmentation techniques, such as back-translation, can be utilized to generate additional training samples, increasing the diversity of the dataset and mitigating overfitting.

Model Drift:

With continuous data updating, the model may drift away from its initial behaviour and generate less reliable responses. Regular retraining and monitoring of the model's performance can help detect and address model drift, ensuring consistent and accurate responses.

Continuous Data Updating and Efficient Embedding Generation

Data Updating:

We would update dataset incrementally in mini-batch updates. Increasing complexity with each update making InstiGPT a more comprehensive tool for students.

Efficient Embedding Generation

Generating embeddings from scratch for each query would increase response time and .computational cost. So pre-trained embeddings can be utilized using models like Word2Vec or GloVe.

Alternative to Google Search and Performance Optimization

Alternative to Google Search:

Instead of relying on Google to scrap data from Insti’s website and give results we could ourself Scrap websites and create Knowledge graphs. Now these Knowledge graphs could be used to create it’s own query engine. At the end we could make a hybrid query engine that uses both KG query engine and vector query engine to answer the question. This would also reduce hallucinations in the model.

Performance Optimization:

We can use Caching. Frequently accessed data and embeddings can be cached to speed up the retrieval process. Also students can ask same or similar query so we could use CacheGPT. CacheGPT uses semantic search to match queries against previously asked queries, and if there’s a match, you simply return the last context instead of performing a full retrieval.

Fine-Tuning and Deployment

Few-shot Learning:

Incorporating few-shot learning techniques, such as meta-learning or prototypical networks, can enable InstiGPT to adapt and generalize to new tasks or domains with limited fine-tuning data, enhancing its performance while reducing the need for extensive retraining.

Addressing Hallucinations

Active Learning with Human-in-the-Loop:

We can employ active learning strategies where students provide feedback on generated responses which can help reduce hallucinations. By iteratively incorporating human feedback into the training process, our model will learn to generate more reliable and contextually relevant responses. We can also incorporate reinforcement learning on given feedback.

Reranking:

We can incorporate reranking techniques to reduce hallucinations in the retrieval augmented generation (RAG) model. Reranking involves reevaluating and reordering candidate responses based on their relevance, quality, and diversity. By considering relevance-based reranking, where responses are ranked based on their similarity to the input query, and quality-based reranking, which considers metrics like fluency and coherence, we can filter out hallucinatory responses and prioritize more accurate and contextually relevant ones.

Providing context:

We can provide context for the response. The context could be the citation of the document or the chunk used for response, on which we could further take feedback. And if context can’t be given “Sorry” would be given as response.