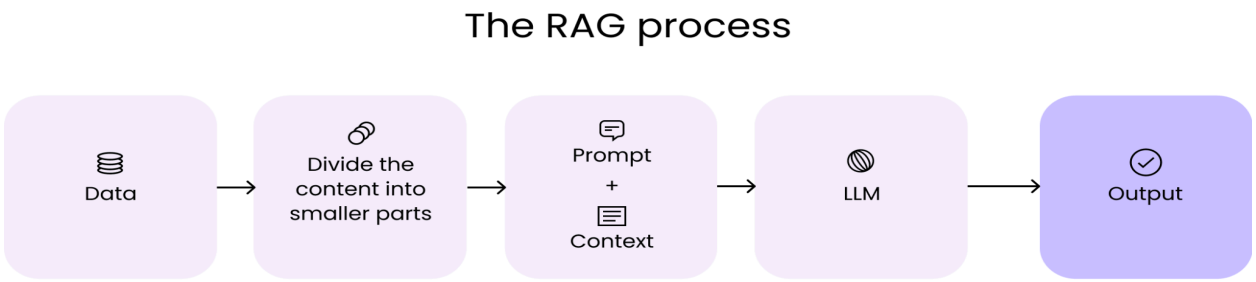
Contract Advisor RAG: Interim Report

Retrieval-augmented generation (RAG)

Retrieval-augmented generation (RAG) is a breakthrough in natural language processing, merging information retrieval with text generation to improve AI-generated content. RAG addresses the shortcomings of large language models (LLMs) by accessing contextually relevant information from external sources like company databases or other sources.

Components of Retrieval-Augmented Generation (RAG)



RAG Components:

- **Information Preprocessing**: Structuring data for accessibility and relevance.
- **Context Integration**: Storing organized data for seamless retrieval and incorporation.
- **Text Generation**: Utilizing LLMs for crafting quality, contextually relevant responses.
- **Post-Processing and Output**: Refining generated text to meet quality standards.
- **User Interface/Deployment**: Delivering the final output through user-friendly interfaces.

Evaluating Retrieval-Augmented Generation (RAG) Systems

Key Data Columns in RAG systems evaluation:

- **Question**: Questions for RAG pipeline evaluation.
- **Answer**: RAG pipeline-generated answers presented to users.
- **Contexts**: Contexts passed into LLM for answering questions.
- **Ground Truths**: Ground truth answers corresponding to questions.

Evaluation Outputs:

- **Faithfulness**: Measures factual accuracy by validating statements in generated answers against context. Two-step process using LLM to identify and verify statements.
- **Answer Relevancy**: Assesses relevance and precision of answers to questions. LLM determines probable questions for a given answer and computes similarity.

- **Context Relevance**: Gauges signal-to-noise ratio in retrieved contextLLM identifies sentences required to answer a question, calculating a ratio based on the total sentences in the context.
- **Context Recall**: Measures retriever's ability to gather necessary information for answering questions. Utilizes ground_truth answers and LLM to check if each statement can be found in retrieved context.

The evaluation of RAG systems entails assessing both the generative and retrieval aspects of the model. Various evaluation methods and metrics have been discussed to measure the performance of RAG systems effectively.

Evaluation Metrics

- **Generation Metrics**: Evaluates the output's faithfulness and relevance to ensure the correctness of the generated text.
- **Retrieval Metrics**: Includes commonly used metrics like normalized Discounted Cumulative Gain (nDCG), Recall, and Precision, as well as more intricate measures such as context precision and recall rated by LLMs.
- **Indexing Metrics**: Focuses on the model's ability to recall information correctly, examining the frequency and impact of errors from Approximate Nearest Neighbor (ANN) algorithms on the retrieval process.

Improving Efficiency and Scalability of RAG Systems:

Optimizing Retrieval and Generation:

- **Data Preparation:** Ensure clean, relevant data and experiment with chunking strategies and embedding models.
- **Retrieval:** Utilize multi-indexing, reranking algorithms, and approximate nearest neighbors for faster retrieval.
- **Generation:** Implement early stopping, knowledge distillation, and pruning for efficient text generation.

Architectural Improvements for Scalability:

- **Distributed systems:** Deploy retrieval and generation components on different machines or clusters for parallel processing.
- **Cloud-based infrastructure:** Leverage cloud platforms with auto-scaling capabilities to handle varying user loads.
- **Caching:** Cache frequently accessed data and retrieval results to avoid redundant processing.

Improving Personalization and Contextualization in RAG Systems

Here are some ways to improve RAG systems in terms of personalization and contextualization:

Understanding User Context:

- **Explicit & Implicit Feedback:** Gather user preferences through ratings or behavior analysis.
- **Contextual Cues:** Utilize conversation history and user location for query context.

Utilizing User Context in Retrieval:

- **Personalized Ranking:** Prioritize documents based on user interests.
- **Context-Aware Retrieval:** Refine searches based on context.

Utilizing User Context in Generation:

- **Personalization Template**: Generate responses using templates tailored to specific users.
- **User-Specific Language Models**: Train models based on individual interaction.

Bias Detection and Mitigation

Detection:

- **Dataset Analysis:** Identify biases by analyzing training data for imbalances in representation and language use.
- **Explainable AI (XAI) Techniques:** Use LIME or SHAP to understand model decisions and detect potential bias sources.
- **Human Evaluation:** Assess responses for fairness through human evaluations.

Mitigation:

- **Data Augmentation**: Supplement training data with diverse examples to counter existing biases.
- **Debiasing Techniques**: Apply adversarial training or counterfactual data augmentation to penalize biased predictions.
- **Fairness-Aware Training Objectives**: Train the model with fairness-promoting objectives alongside other metrics.
- **Monitoring and Continuous Improvement**: Regularly monitor fairness metrics and iterate improvements.

Project Progress

- Data retriever function
- Langchain pipeline function
- Ragas evaluation function