Large Language Models and Retrieval-Augmented Generation (RAG)

Acme Research Report, 2024

Introduction to Large Language Models

Large Language Models (LLMs) are deep neural networks trained on vast text corpora to predict and generate human-like language. They excel at general knowledge, reasoning, and text completion tasks but are limited by static training data and hallucination risks. To address this, a new class of systems emerged — Retrieval-Augmented Generation (RAG) — that blends LLM reasoning with dynamic knowledge retrieval.

What is RAG?

Retrieval-Augmented Generation (RAG) enhances LLMs by connecting them to an external knowledge base or vector database. When a user asks a question, the system retrieves the most relevant text snippets based on semantic similarity, then feeds these snippets into the LLM prompt. Popular implementations of RAG use tools like FAISS, Qdrant, or Milvus to store and search embeddings efficiently.

System Architecture

A RAG system typically has three layers: 1. Document Ingestion – PDFs, manuals, or reports are split into text chunks and embedded into vector representations using models like text-embedding-3-large. 2. Retrieval – When queried, similar chunks are fetched from a vector database based on cosine similarity. 3. Generation – The retrieved context is added to the LLM prompt for grounded, explainable answers.

Limitations and Future Work

While RAG reduces hallucinations, it depends heavily on the quality of retrieved documents. Irrelevant or noisy snippets can still lead to incorrect answers. Emerging improvements include contextual compression, tool-augmented reasoning, and hybrid search that mixes semantic and keyword-based retrieval. Future research aims to make RAG agentic, allowing the model to autonomously decide when to search, how to reformulate queries, and how to verify responses using multi-hop reasoning.

References

Lewis et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. OpenAI (2024). GPT-4 Technical Report. Qdrant Docs (2024). Scalable Vector Search for AI Applications.