Deploying AI

Retrieval Augmented Generation (RAG)

```
$ echo "Data Science Institute"
```

Introduction

Agenda

Agenda

- RAG Architecture
- Retrieval Algorithms and optimization

Introduction

- To solve a task, a model needs both instructions and the necessary information.
- Models are more likely to hallucinate when missing context.
- Context differs per query, while instructions are generally fixed.
- Two dominant context construction patterns are retrieval-augmented generation (RAG) and agents.
- RAG retrieves information from external data sources.
- Agents use external tools, enabling broader capabilities including world interaction.

Why RAG and Agents Matter

- RAG enhances a model’s generation by retrieving relevant information.
- Agents expand capabilities by leveraging tools like search APIs or code execution.
- Both patterns address models’ weaknesses and extend their utility.
- These approaches have produced impressive demos and practical applications.
- They are widely seen as the future of AI-powered systems.

Retrieval-Augmented Generation (RAG)

RAG Overview

- RAG retrieves relevant information from external memory sources to supplement model outputs.
- External sources can include internal databases, prior conversations, or the web.
- This technique was first formalized in research around open-domain question answering.
- RAG helps models generate more accurate and less hallucinated answers.

Why RAG Exists

- Foundation models have limited context length.
- Users produce more data than can fit in context windows.
- Long contexts are costly and may reduce model efficiency.
- RAG selectively retrieves relevant chunks, keeping inputs concise and focused.
- Retrieval acts as context construction, similar to feature engineering in ML.

RAG Architecture

- A RAG system consists of two components: a retriever and a generator.
- The retriever indexes and retrieves relevant chunks from external memory.
- The generator produces responses conditioned on retrieved data.
- These two parts may be trained separately or jointly.
- System performance depends heavily on retriever quality.

Retrieval Algorithms

- Retrieval has a century-long history in information systems.
- Algorithms differ in how they score document relevance.
- Two broad categories are term-based retrieval and embedding-based retrieval.
- Retrieval solutions also include hybrid approaches that combine both.

Term-Based Retrieval

- Term-based methods focus on lexical matches, such as keyword overlaps.
- Popular algorithms include TF-IDF and BM25.
- Term frequency measures how often a term appears in a document.
- Inverse document frequency downweights common terms.
- BM25 improves TF-IDF by adjusting for document length.

Embedding-Based Retrieval

- Embedding-based methods retrieve by semantic similarity rather than exact terms.
- Queries and documents are converted into embeddings.
- The retriever finds the nearest embeddings in vector space.
- Requires a vector database to efficiently store and search embeddings.
- Common libraries include FAISS, ScaNN, Annoy, and Hnswlib.

Sparse vs. Dense Retrieval

- Sparse retrieval methods represent data with mostly zero values.
- Dense retrieval uses embeddings with nonzero values across dimensions.
- SPLADE is an example of sparse embedding retrieval.
- Dense methods capture semantics better but can be costlier to compute.

Comparing Retrieval Approaches

- Term-based retrieval is fast, simple, and inexpensive.
- Embedding-based retrieval enables natural queries but requires embeddings and vector search.
- Hybrid approaches combine both, often using term-based retrieval first and semantic reranking later.
- The choice depends on trade-offs between speed, cost, and accuracy.

Evaluating Retrieval

- Key metrics include context precision and recall.
- Precision measures what percentage of retrieved results are relevant.
- Recall measures what percentage of relevant results are retrieved.
- Additional ranking metrics include NDCG, MAP, and MRR.
- Ultimately, retrieval is valuable only if it improves generative outputs.

Retrieval Optimization

- Several tactics can optimize retrieval performance.
- **Chunking**: splitting documents into manageable parts.
- **Reranking**: refining the ranking of retrieved documents.
- **Query rewriting**: clarifying ambiguous queries.
- **Contextual retrieval**: enriching documents with metadata or context.

Chunking Strategies

- Documents may be split by characters, words, sentences, or paragraphs.
- Recursive chunking reduces arbitrary breaks.
- Overlaps prevent loss of key boundary information.
- Chunk size affects both coverage and efficiency.
- There is no universal best practice; experimentation is necessary.

Query Rewriting

- Queries are rewritten to make intent explicit and unambiguous.
- Ambiguous follow-ups, such as “How about Emily?”, require contextual rewriting.
- AI models can automate query rewriting with carefully designed prompts.
- Rewriting may involve identity resolution or knowledge lookup.

Contextual Retrieval

- Chunks can be augmented with metadata like tags or entities.
- Additional questions may be linked to articles to aid retrieval.
- Document titles and summaries can help situate chunks.
- Generated context improves retrievability without altering core content.

RAG Beyond Text

- RAG can operate on multimodal and structured data.
- **Multimodal RAG** augments queries with images, audio, or video.
- **Tabular RAG** retrieves and executes queries on structured datasets.
- SQL executors are often used to query relational data.
- These variations extend RAG beyond text-only use cases.

Summary

Key Takeaways

- RAG retrieves external knowledge to improve model responses.
- Retrieval methods include term-based, embedding-based, and hybrid approaches.
- Optimizations such as chunking and reranking boost retrieval quality.
- RAG extends beyond text to multimodal and tabular contexts.
- Agents integrate reasoning, planning, and tool use to act in environments.
- Tools extend agent capabilities in perception, reasoning, and action.
- Evaluating agents requires balancing accuracy, cost, latency, and safety.

References

References

- Huyen, Chip. Designing machine learning systems. O'Reilly Media, Inc., 2022