Knowledge Base and Amazon Bedrock components

Ingestion and transformation

Objective

The objective of this component is to preprocess the document files to be indexed in the Knowledge Base. Depending on the source and type, each group of documents will be treated slightly different (see Details).

Transformation

The papers are stored in a staging s3 bucket. According to the category, they are processed and transformed, as following:

- **doi**, **sccs**, **zotero-papers** - copied
- **cir** - deduplicated and copied
- **pubmed**: parse JSON and
  - (a) generate metadata
  - (b) copy pdf or, if no pdf, extract abstract and copy as *.txt in the transformed s3 bucket

Details

The initial batch of documents is of 5 types:

- **cir** - Cosmetic Ingredient Review documents
- **doi** - journal papers, pdf only
- **pubmed** - journal papers scrapped from pubmed, 90% of content, pdf and metadata (json), ~70% of files with only abstract → for these files, we ingest both content and metadata.
- **sccs** - Scientific Committee on Consumer Safety documents, pdf only
- **zotero-papers** - Zotero is an reference management tool that can be used as an assistant for your research activity - pdf documents

Amazon Bedrock components

Amazon Bedrock Models

The following models are used:

Model	Role	Details
Titan Text Embeddings v2	Embeddings for Knowledge Base	Through Amazon Bedrock
Anthropic Claude Sonnet 3	• Knowledge Base (augmented generation) • Agents • Evaluation Framework	Through Amazon Bedrock
BERT	• Evaluation Framework (BERT score)	bert-score Python package

Amazon Bedrock Agents

The following agents are currently implemented:

Agent	Role	Comments
Knowledge Base Agent	Retrieve and summarise information from the Knowledge Base	Documented in Knowledge Base RAG Fusion Agent
RAG Fusion Rephrase Agent	Generate variation to the initial question before serving it as an input/question to Knowledge Base Agent	Documented in Knowledge Base RAG Fusion Agent
Response Agregator Agent	Gathers responses from the pool of Knowledge Base Agents and create summarization	Documented in Knowledge Base RAG Fusion Agent
SQL Generator Agent	Use the DB schema & user input to generate an SQL queryuser input to generate an SQL query	Documented in Vitic Agent V 1
SQL Answer Generator Agent	Generate the response from the result of running SQL query	Documented in Vitic Agent V 1
Aggregator Agent	Aggregate SQL response with KB (RAG Fusion) response in one unique response	Documented in Combined RAG Fusion V2
SQL Agent using Action Group	Use function calls to: • retrieve DB schema • run SQL query Generate SQL query Analyse the result, perform reasoning, iterate (max steps) until get best answer	Implemented, not integrated, see: Vitic Agent with Action Group.

	Uses collaborator pattern with Action Group.	
Broker Agent	Generate user inputs adapted to KB and to Vitic and distribute them. Uses function calls to: • call KB agent • call SQL Agent	To be documented
Agentic RAG Agent	Use KB connectivity to retrieve question from KB, perform response analysis, reasoning, iterate (max steps) until get best answer (alternative to RAG fusion workflow).	Implemented, see: Agentic RAG
Agentic RAG Multisource Agent	Use KB connectivity to retrieve question from KB, Vitic connectivity to retrieve data from Vitic DB, perform response analysis per each retrieval type in paralel, reasoning/reflection on both analytics, iterate (max steps) until get best answer (alternative to RAG fusion workflow).	Implemented, see: Agentic RAG - Multisource

Amazon Bedrock Knowledge Base

Implementation of Amazon Bedrock Knowledge Base uses:

- **OpenSearch serverless** Vector Database service
- **Titan Text Embeddings v2** embedding model, floating point embeddings, 1024 vector dimension
- **Data source**: S3, indexed 129,952 files, 121,263 metadata (40 GB). 300 tokens chunks, 20% superposition.
- **Anthropic Claude 3 Sonnet** model (28K context window) model used with the Knowledge Base agent.

Areas of improvement

- Custom chunking, inverse text to document indexing
- Optimize embeddings
- Reduce vector database total cost (indexing, searching). Adoption of alternative (and cheaper) vector database
- Smaller, faster models for some of the tasks (e.g. summarisation).
- Leverage graph information from metadata to enrich the alternative query generation (from Agentic RAG to Graph RAG).