Building and Mining Knowledge graphs

(KEN4256)

Lecture 10: Retrieval Augmented Generation

Maastricht University
Institute of Data Science

id: KEN4256_L10
version: 1.2024.0
created: March 17, 2024
last modified: March 26, 2024
published on: March 26, 2024

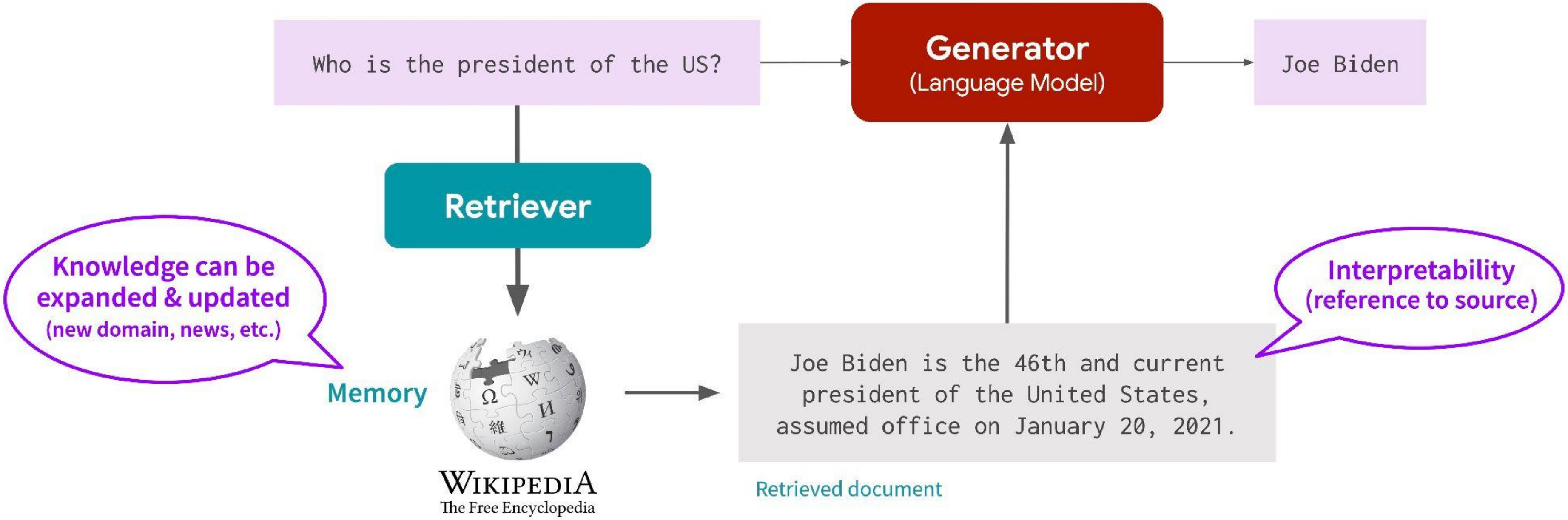
Introduction

Retrieval Augmented Generation is a methodology by which a generative model (e.g. transformer-based neural network) is augmented with external knowledge retrieval. This helps address limitations in LLMs including factual accuracy, timeliness, and access to non-public knowledge.

External knowledge source could involve

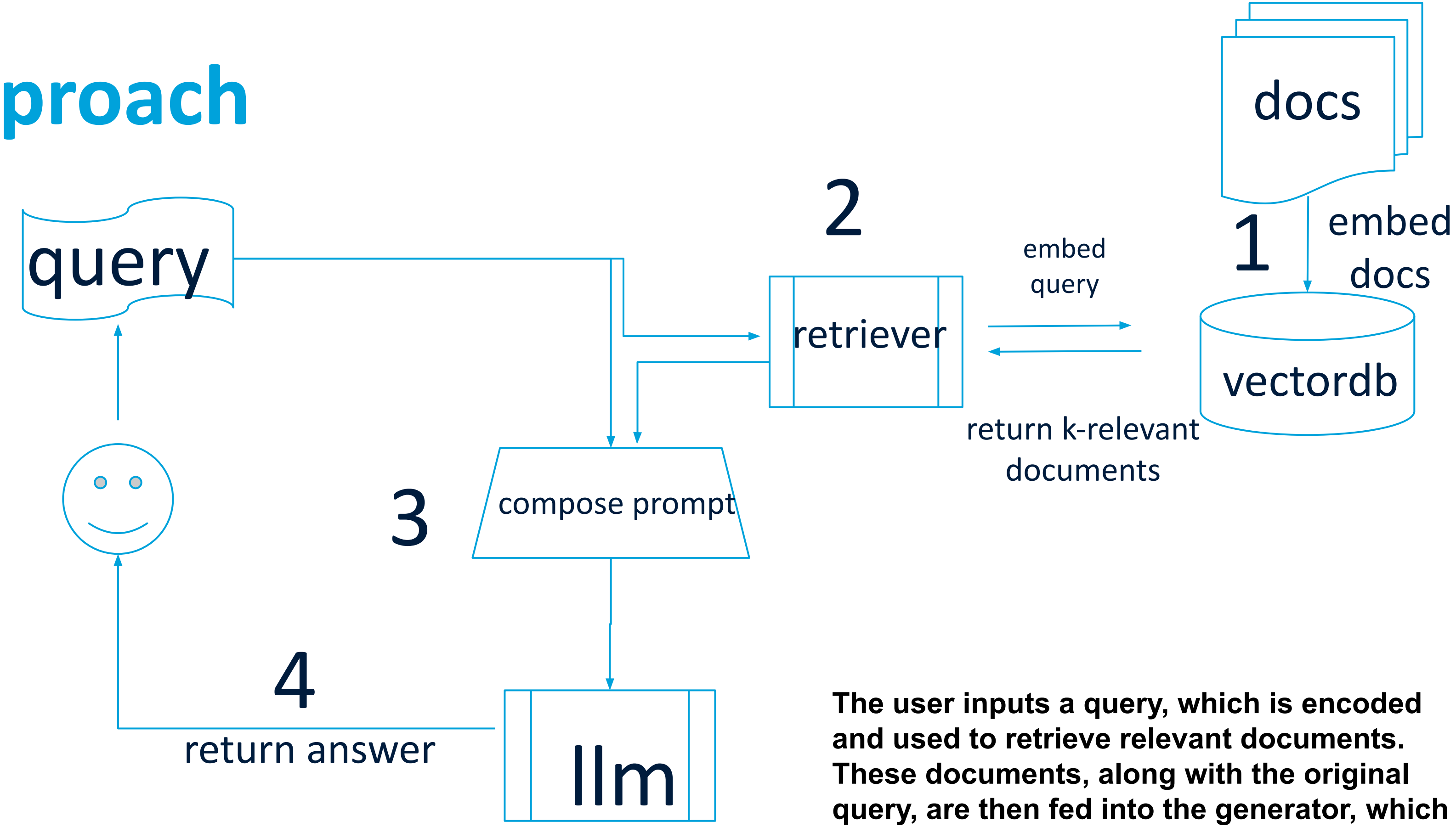
- web search
- access to documents / databases / knowledge graphs
- web services

Retrieval augmentation

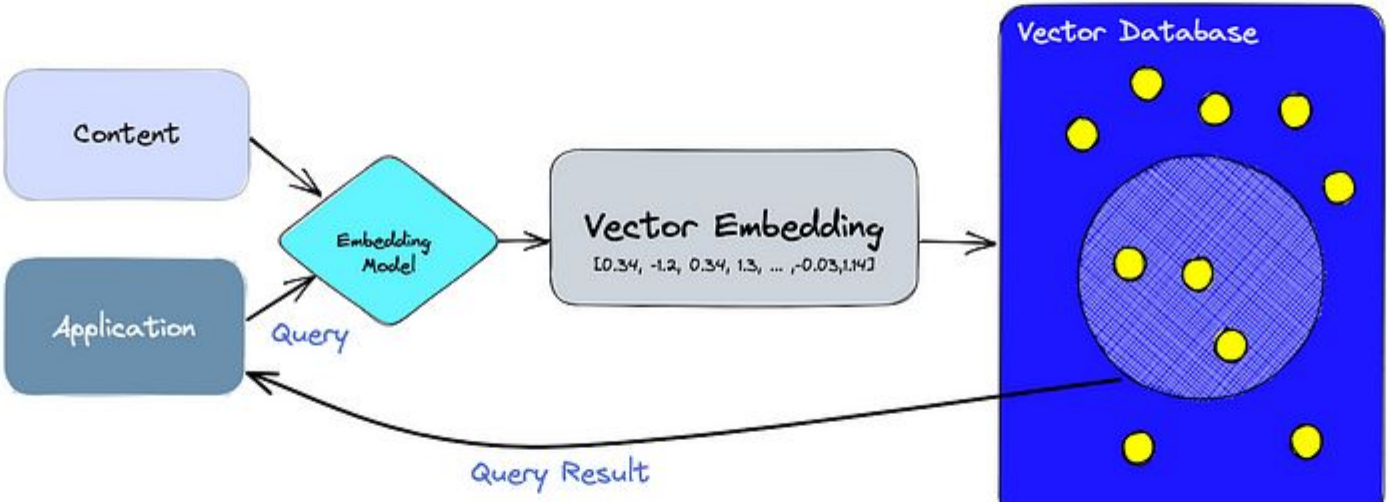


https://cs.stanford.edu/~myasu/blog/racm3/

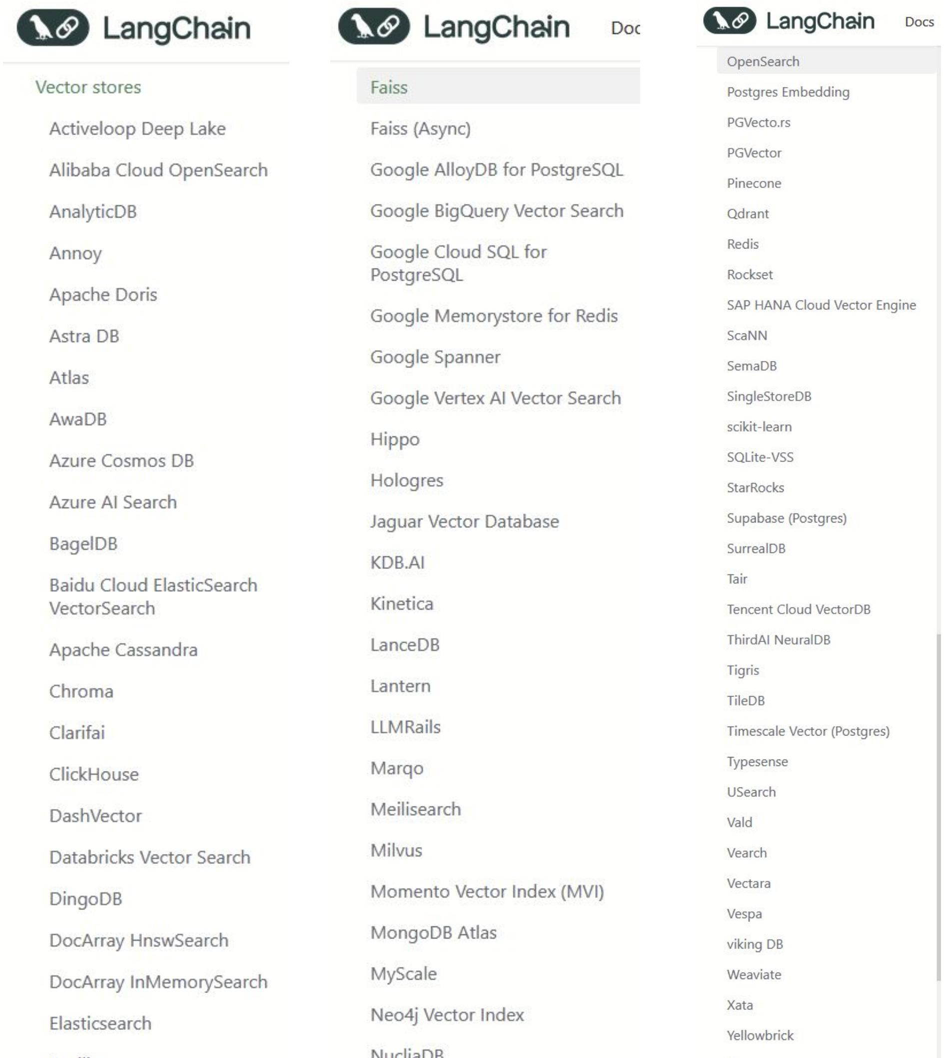
Approach



The user inputs a query, which is encoded and used to retrieve relevant documents. These documents, along with the original query, are then fed into the generator, which produces the final output.



https://www.pinecone.io/learn/vector-database/



Features

- self-hosted/managed
- ranking algorithms
- metadata filtering
- indexing
- access control
- security/privacy certification (SOC, HIPPA, GDPR)

Methodology

- The **query encoder** processes the input query to generate a representation suitable for retrieval.
- The **retriever** uses this representation to fetch relevant documents or data from the database.
- The **document database** is a pre-compiled collection of texts or knowledge that the retriever queries.
- The **generator** then takes the original query along with the retrieved documents to generate a coherent and contextually enriched response.

Frameworks

- LangChain (RAG)
- Microsoft Semantic kernel
- LlamaIndex (RAG, notebook)
- `llmware`

LLM provisioning

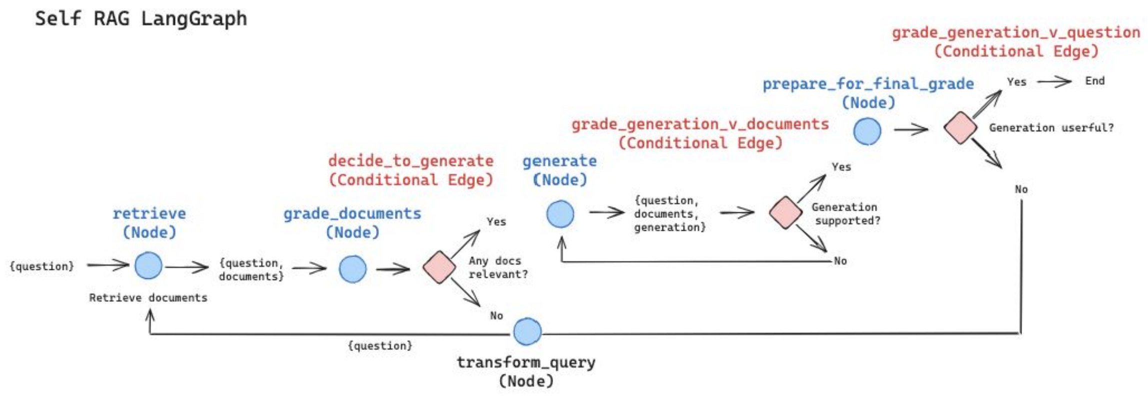
- Ollama - download models and serve them up via http
- LlamaCpp
- vLLM

Langgraph

LangGraph is a library for building stateful, multi-actor applications with LLMs, built on top of (and intended to be used with) LangChain. It extends the LangChain Expression Language with the ability to coordinate multiple chains (or actors) across multiple steps of computation in a cyclic manner.
https://github.com/langchain-ai/langgraph

The main use is for adding **cycles** to your LLM application. Crucially, this is NOT a **DAG** framework. If you want to build a DAG, you should just use LangChain Expression Language.

Cycles are important for agent-like behaviors, where you call an LLM in a loop, asking it what action to take next.



Document loaders

First step is to load a collection of documents for processing. Many integrations available.

Document loaders
- CSV
- File Directory
- HTML
- JSON
- Markdown
- PDF

Document loaders
- acreom
- AirbyteLoader
- Airbyte CDK (Deprecated)
- Airbyte Gong (Deprecated)
- Airbyte Hubspot (Deprecated)
- Airbyte JSON (Deprecated)
- Airbyte Salesforce (Deprecated)
- Airbyte Shopify (Deprecated)
- Airbyte Stripe (Deprecated)
- Airbyte Typeform (Deprecated)
- Airbyte Zendesk Support (Deprecated)
- Airtable
- Alibaba Cloud MaxCompute
- Amazon Textract
- Apify Dataset
- ArcGIS
- Arxiv
- AssemblyAI Audio Transcripts
- AstraDB
- Async Chromium
- AsyncHtml
- Athena
- AWS S3 Directory
- AWS S3 File
- AZLyrics
- Azure AI Data
- Azure Blob Storage Container
- Azure Blob Storage File
- Azure AI Document Intelligence
- BibTeX
- BiliBili
- Blackboard
- Blockchain
- Brave Search
- Browserless
- Cassandra
- ChatGPT Data
- College Confidential
- Concurrent Loader
- Confluence
- CoNLL-U
- Copy Paste
- Couchbase
- CSV
- Cube Semantic Layer
- Datadog Logs
- Diffbot
- Discord
- Docugami
- Docusaurus
- Dropbox
- DuckDB
- Email
- EPub
- Etherscan
- EverNote
- Facebook Chat
- Fauna
- Figma
- Geopandas
- Git
- GitBook
- GitHub
- Google AlloyDB for PostgreSQL
- Google BigQuery
- Google Bigtable
- Google Cloud SQL for SQL serve
- Google Cloud SQL for MySQL
- Google Cloud SQL for PostgreSQL
- Google Cloud Storage Directory
- Google Cloud Storage File
- Google Firestore in Datastore Mode
- Google Drive
- Google El Carro for Oracle Workloads
- Google Firestore (Native Mode)
- Google Memorystore for Redis
- Google Spanner
- Google Speech-to-Text Audio Transcripts
- Grobid
- Gutenberg
- Hacker News
- Huawei OBS Directory
- Huawei OBS File
- HuggingFace dataset
- iFixit
- Images
- Image captions
- IMSDb
- Iugu
- Joplin
- Jupyter Notebook
- lakeFS
- LarkSuite (FeiShu)
- Mastodon
- MediaWiki Dump
- Merge Documents Loader
- mhtml
- Microsoft Excel
- Microsoft OneDrive
- Microsoft OneNote
- Microsoft PowerPoint
- Microsoft SharePoint
- Microsoft Word
- Modern Treasury
- MongoDB
- News URL
- Notion DB 1/2
- Notion DB 2/2
- Nuclia
- Obsidian
- Open Document Format (
- Open City Data
- Org-mode
- Pandas DataFrame
- Pebblo Safe DocumentLoa
- Polars DataFrame
- Psychic
- PubMed
- PySpark
- Quip
- ReadTheDocs Documentat
- Recursive URL
- Reddit
- Roam
- Rockset
- rspace
- RSS Feeds
- RST
- Sitemap
- Slack
- Snowflake
- Source Code
- Spreedly
- Stripe
- Subtitle
- SurrealDB
- Telegram
- Tencent COS Directory
- Tencent COS File
- TensorFlow Datasets
- TiDB
- 2Markdown
- TOML
- Trello
- TSV
- Twitter
- Unstructured File
- URL
- Vsdx
- Weather
- WebBaseLoader
- WhatsApp Chat
- Wikipedia
- XML
- Xorbits Pandas DataFra
- YouTube audio
- YouTube transcripts
- Yuque

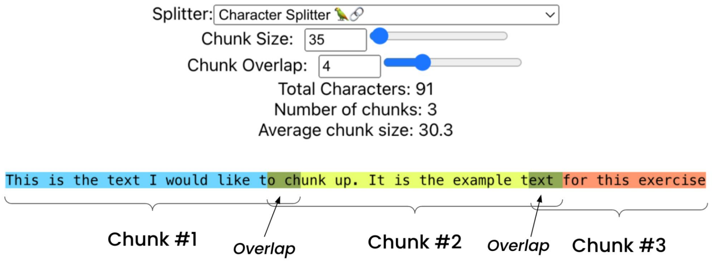
Text Splitting

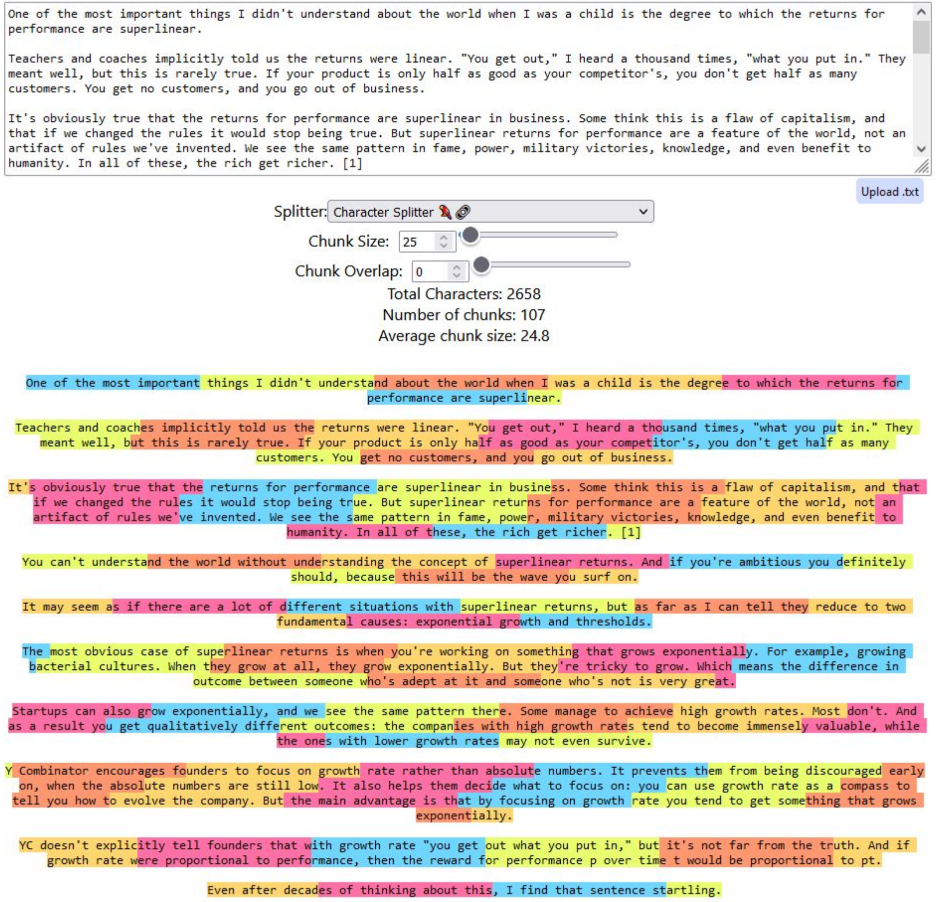
We will need to split the documents into chunks that facilitate the retrieval of a sufficiently large context to help answer the question but can also fit into the llm context window.

Split by:

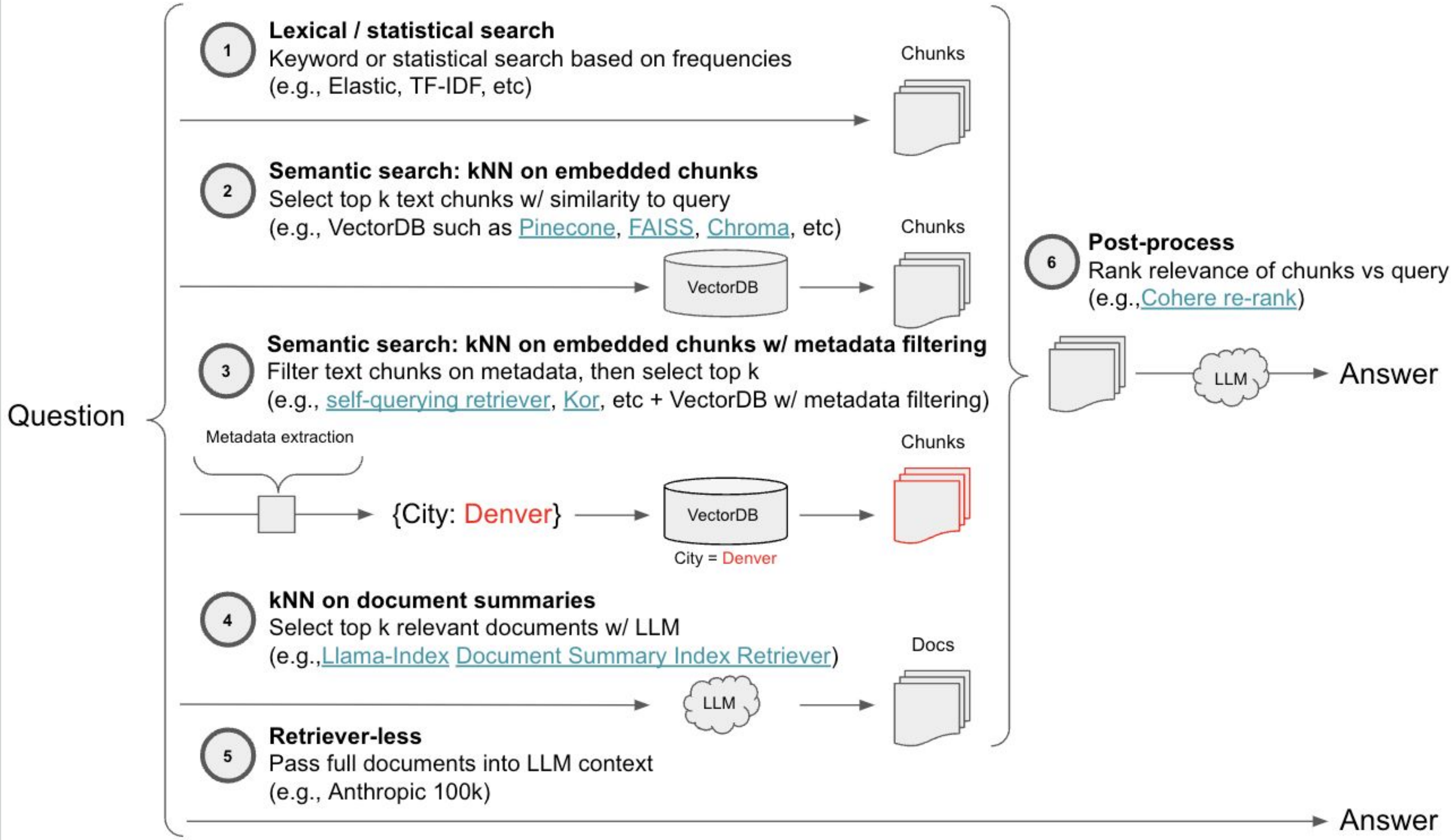
- # of tokens, character, sentence, paragraph
- document structure (e.g. HTML headers)
- semantic structure
- multi-vector indexing

see notebook





https://chunkviz.up.railway.app



Maastricht University

Institute of Data Science

https://github.com/langchain-ai/auto-evaluator/tree/main/streamlit

AgenticChunker

1. Get propositions
2. For each proposition, ask the LLM, should this be in an existing chunk or a make a new one?

{ } JSON
{ } d6613
chunk_id : "d6613"
[ ] propositions
0 : "The month is October."
1 : "The year is 2023."
summary : "This chunk contains information about specific dates and years."
{ } 51c07
chunk_id : "51c07"
[ ] propositions
0 : "One of the most important things that I didn't understand about the world as a child was the degree to which the returr
1 : "Teachers and coaches implicitly told us that the returns were linear."
2 : "The returns for performance are superlinear in business."
3 : "Some people think the superlinear returns for performance are a flaw of capitalism."
4 : "Some people think that changing the rules of capitalism would stop the superlinear returns for performance from being
5 : "Superlinear returns for performance are a feature of the world."
6 : "Superlinear returns for performance are not an artifact of rules that humans have invented."
7 : "The same pattern of superlinear returns is observed in fame."
8 : "The same pattern of superlinear returns is observed in power."
9 : "The same pattern of superlinear returns is observed in military victories."
10 : "The same pattern of superlinear returns is observed in knowledge."
11 : "The same pattern of superlinear returns is observed in benefit to humanity."
summary : "This chunk explores the concept of superlinear returns across various aspects of life and performance, examining
{ } 11c9b
chunk_id : "11c9b"
[ ] propositions
0 : "I heard a thousand times that 'You get out what you put in.'"
1 : "The statement that 'You get out what you put in' is rarely true."
2 : "You get no customers if your product is only half as good as your competitor's product."
summary : "This chunk explores the principles of effort and reward, and critiques the saying 'You get out what you put in.'"

Multi-vector Indexing

semantic search for a vector that is derived from something other than the raw text

summary

use LLM to generate a summary of the full document, which can be indexed for retrieval.

"The document discusses the concept of superlinear returns for performance, where the rewards for performance are not proportional to the effort put in. It explains that this concept is present in various aspects of life, such as business, fame, power, military victories, and benefit to humanity....

hypothetical questions

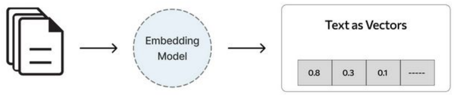
use LLM to generate hypothetical questions that would best match the document.

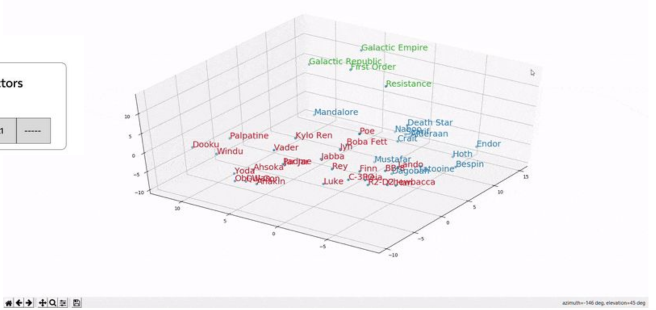
["What was the author's first experience with programming like?",
'Why did the author switch their focus from AI to Lisp during their graduate studies?',
'What led the author to contemplate a career in art instead of computer science?']

parent document

do search against smaller document fragments, and retrieve the parent document section for fuller context.

Text Embedding





The next step is to embed the text. Use sentence and document embedding models.

[Sentence Transformers library](#) is a Python framework for sentence, text and image embeddings. Gives you access to embedding models on hugging face.

["sentence-transformers/all-MiniLM-L6-v2"](#) maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search. It is very small (0.09GB), currently #75 on leaderboard

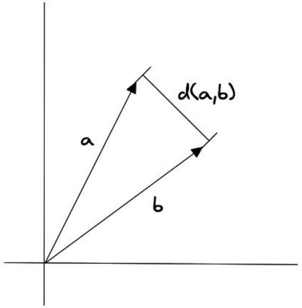
bge-small-en-v1.5 (0.13GB) currently #29, supported in [FastEmbed python library](#).

Rank	Model	Model Size (GB)	Embedding Dimensions	Max Tokens	Average (56 datasets)
1	SFR-Embedding-Mistral	14.22	4096	32768	67.56
2	voyage-lite-02-instruct		1024	4000	67.13
3	GritLM-7B	14.48	4096	32768	66.76
4	e5-mistral-7b-instruct	14.22	4096	32768	66.63
5	GritLM-8x7B	93.41	4096	32768	65.66
6	echo-mistral-7b-instruct-last	14.22	4096	32768	64.68
7	mxbai-embed-large-v1	0.67	1024	512	64.68
8	UAE-Large-V1	1.34	1024	512	64.64
9	text-embedding-3-large		3072	8191	64.59
10	voyage-lite-01-instruct		1024	4000	64.49
11	Cohere-embed-english-v3.0		1024	512	64.47

Massive Text Embedding Benchmark (MTEB)
https://huggingface.co/spaces/mteb/leaderboard

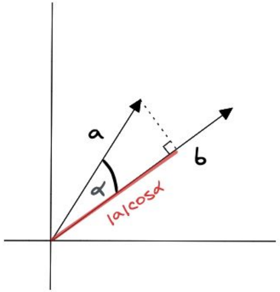
similarity measures

Euclidean distance



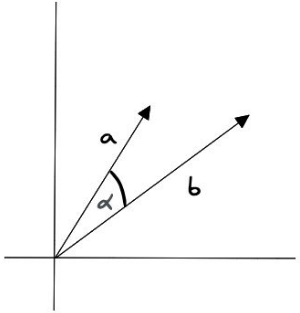
$$d(\mathbf{a},\mathbf{b}) = \sqrt{(\mathbf{a_1} - \mathbf{b_1})^2 + (\mathbf{a_2} - \mathbf{b_2})^2 + \ldots + (\mathbf{a_n} - \mathbf{b_n})^2}$$

dot product



$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

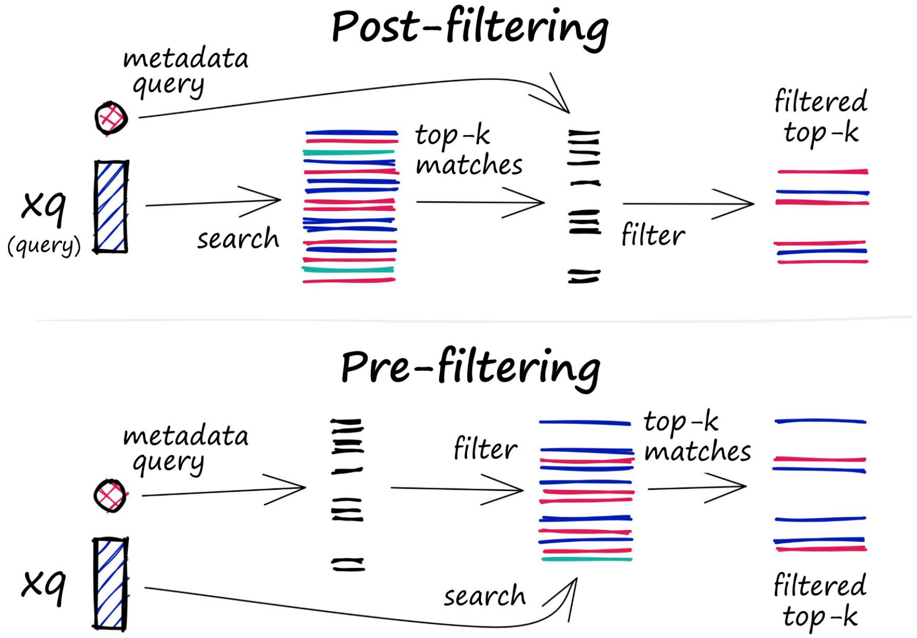
cosine similarity



$$sim(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||}$$

metadata filtering (aka self-querying)

The retriever can query the knowledge source and filter document metadata



RAG over RDF Graphs

Generate labels with URIs for indexing & retrieval.
Further customization is done by code.

- LangChain (cf. docs: RAG with memory, streaming RAG)
- FastEmbed embeddings
- Qdrant vectorstore
- LlamaCpp inference library
- Mixtral 8x7B LLM

https://github.com/vemonet/langchain-rdf/blob/main/tests/rag_ontology.ipynb

KG retriever (Ontotext graphdb)

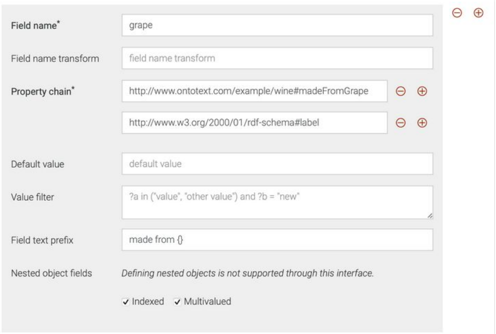
Generates a text representation from the RDF graph.

Create rules for the retriever that are inserted and executed from the KG.

```
wine:Franvino
    rdf:type wine:RedWine ;
    wine:madeFromGrape wine:Merlo ;
    wine:madeFromGrape wine:CabernetFranc ;
    wine:hasSugar "dry" ;
    wine:hasYear "2012"^^xsd:integer ;
    wine:hasWinery "Semantinos" .
```

```
Franvino:
- is a RedWine.
- made from grape Merlo.
- made from grape Cabernet Franc.
- has sugar dry.
- has year 2012.
```

https://graphdb.ontotext.com/documentation/10.6/retrieval-graphdb-connector.html



```
...
INSERT DATA {
        retr-index:starwars retr:createConnector '''
{
  "retrievalUrl": "http://localhost:8000",
  "retrievalBearerToken": "<your-bearer-token>",
...
```

Talk to Your Graph

who is taller, luke or han solo?

Han Solo is taller than Luke. Han Solo has a height of 180.0, while Luke has a height of 172.0.

where is han solo from?

Han Solo is from Corellia.

who are the pilots of the millenium falcon?

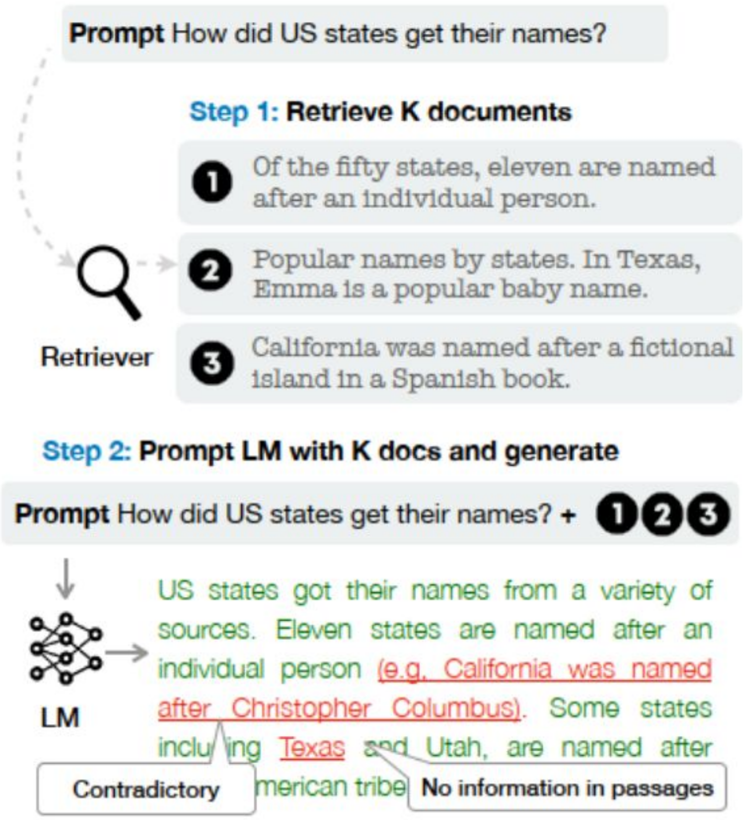
The pilots of the Millennium Falcon are Chewbacca, Han Solo, Lando Calrissian, and Nien Nunb.

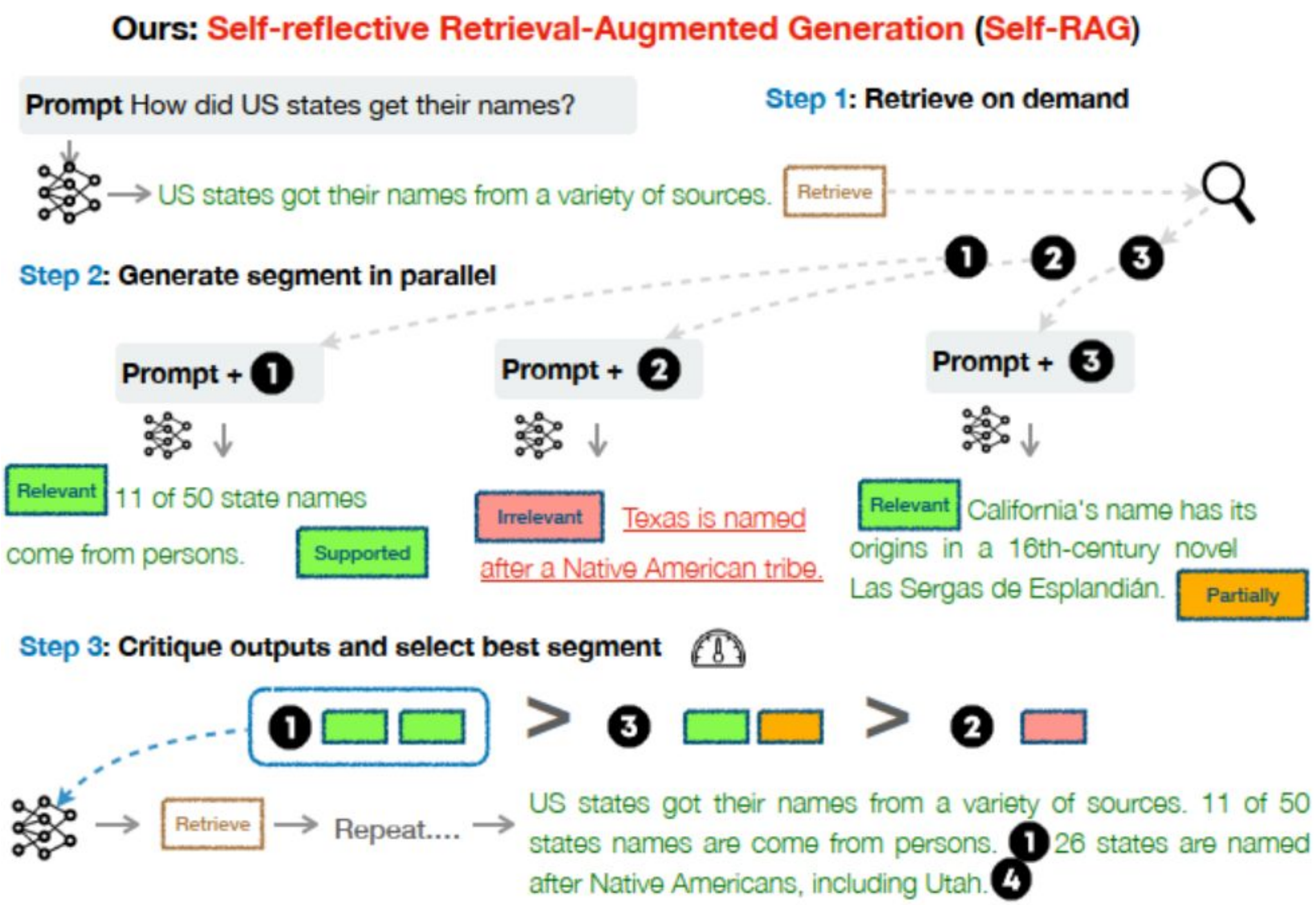
who is the tallest of them?

Chewbacca is the tallest among them with a height of 228.0.

https://graphdb.ontotext.com/documentation/10.6/talk-to-graph.html

Retrieval-Augmented Generation (RAG)





https://arxiv.org/abs/2310.11511 [langchain notebook]()

Corrective Retrieval Augmented Generation (CRAG)

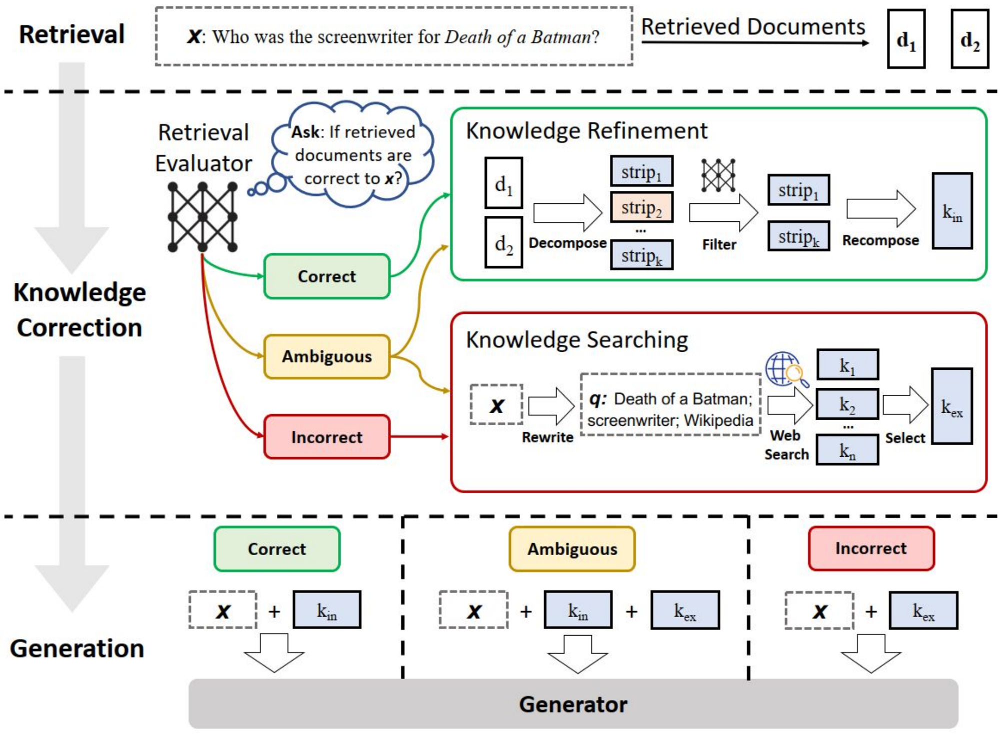


Figure 2: An overview of CRAG at inference. A retrieval evaluator is constructed to evaluate the relevance of the retrieved documents to the input, and estimate a confidence degree based on which different knowledge retrieval actions of {Correct, Incorrect, Ambiguous} can be triggered.

[langchain notebook]

https://arxiv.org/abs/2401.15884

GraphDB Connectors

Translate user input into structured queries using the LLM

Integrations

- Diffbot Graph Transformer
- ArangoDB QA chain
- Neo4j DB QA chain
- FalkorDBQAChain
- Gremlin (with CosmosDB) QA chain
- HugeGraph QA Chain
- KuzuQAChain
- Memgraph QA chain
- NebulaGraphQAChain
- NetworkX
- Ontotext GraphDB QA Chain
- GraphSparqlQAChain
- Neptune Open Cypher QA Chain
- Neptune SPARQL QA Chain

```
chain = GraphSparqlQAChain.from_llm(
    ChatOpenAI(temperature=0), graph=graph, verbose=True
)
```

```
chain.run("What is Tim Berners-Lee's work homepage?")
```

```
> Entering new GraphSparqlQAChain chain...
Identified intent:
SELECT
Generated SPARQL:
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?homepage
WHERE {
    ?person foaf:name "Tim Berners-Lee" .
    ?person foaf:workplaceHomepage ?homepage .
}
Full Context:
[]

> Finished chain.
```

```
"Tim Berners-Lee's work homepage is http://www.w3.org/People/Berners-Lee/."
```

```
from langchain.chains import GraphQAChain
```

```
chain = GraphQAChain.from_llm(OpenAI(temperature=0), graph=graph, verbose=True)
```

```
chain.run("what is Intel going to build?")
```

```
> Entering new GraphQAChain chain...
Entities Extracted:
 Intel
Full Context:
Intel is going to build $20 billion semiconductor "mega site"
Intel is building state-of-the-art factories
Intel is creating 10,000 new good-paying jobs
Intel is helping build Silicon Valley

> Finished chain.
```

```
' Intel is going to build a $20 billion semiconductor "mega site" with state-of-the-art factories, creating 10,00
```

Summary

Retrieval Augmented Generation (RAG) is a powerful technique that uses external knowledge sources to augment the capabilities of a Large-Large Model. There are many factors to consider in the development of the pipeline including

- document chunking strategies
- embedding model selection
- vector database selection
- relevancy and corrective techniques

KGs can also be embedded into vectorDBs, with strategies needed to retrieve fragments across the graph span.