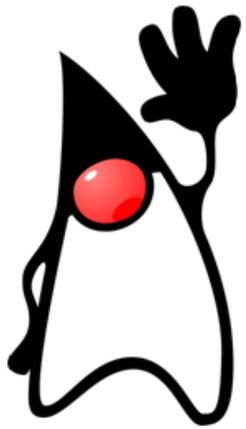
Developing LLM-Driven Java Applications with Langchain4j



by Jansen Ang

Meet
JANSEN ANG

FREELANCE JAVA DEVELOPER

- Jansen Ang is a one of the leaders of the Java User Group Philippines.
- He is currently a freelance Java developer and has been developing Java applications with various companies throughout the years.
- He dedicates his time to contributing to the Java community and organizing JUG PH meetups.

SPONSORED BY:

O&B

JUG ph
Java User Group

https://www.linkedin.com/in/jansen-ang/

https://github.com/jmgang

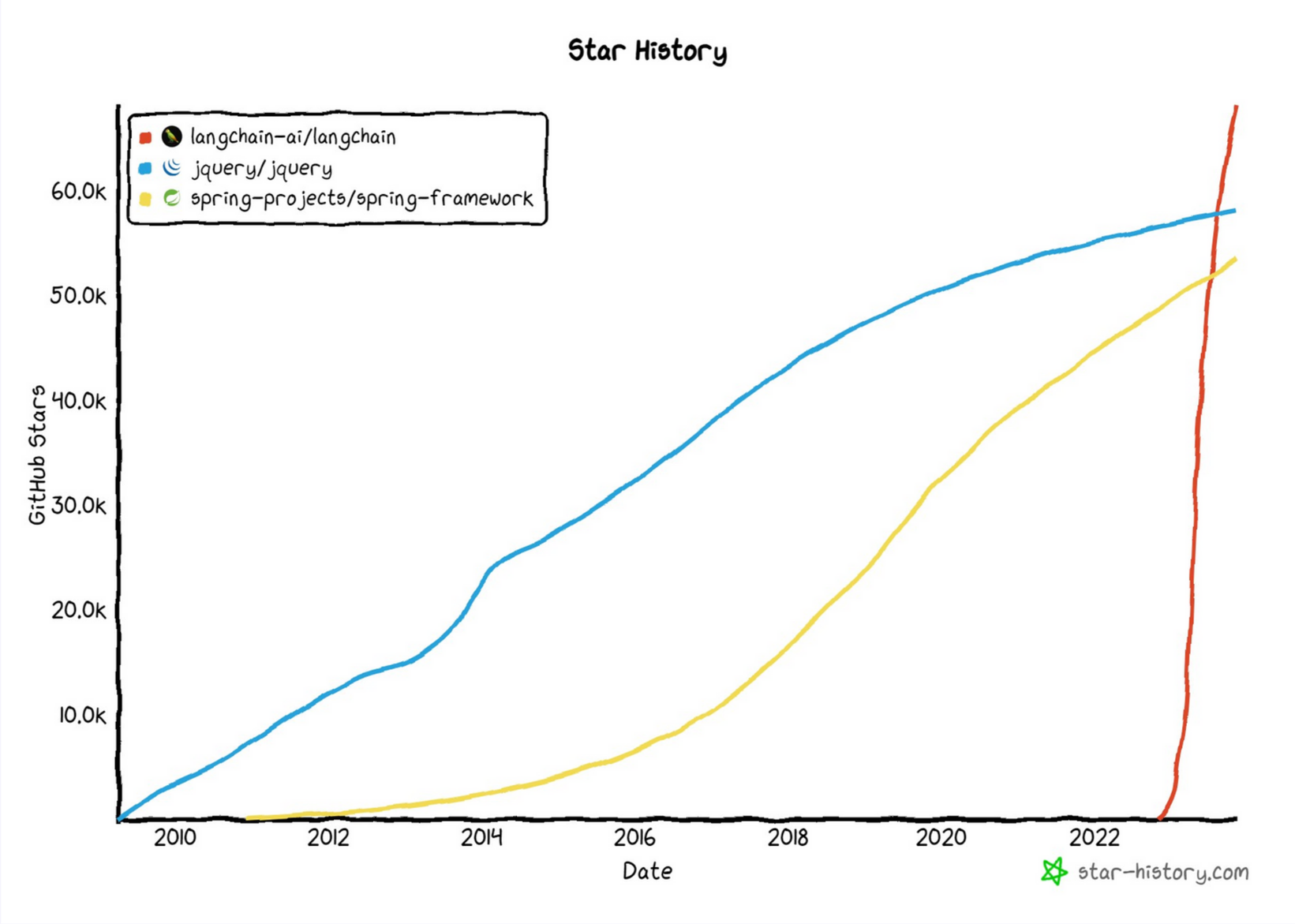
Roadmap

01 Langchain, Langchain4j and its Rise

02 Components of Langchain4j
- Concepts
- Use-Cases

03 Other and Future features

Langchain has achieved more stars in less than a year compared to other popular frameworks

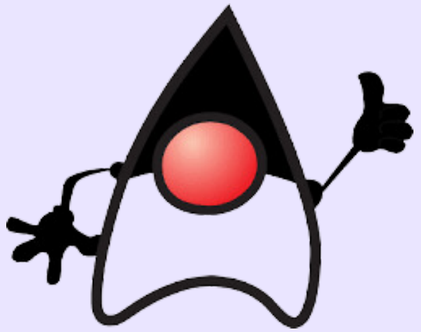


- Open-source framework designed for developers to integrate large language models for application development
- Simplifies the process of creating Generative AI application interfaces
- Q&A over documents, analyzing structured data, interacting with apis, code understanding, agents, chatbots, code writing, extraction, summarization, etc.
- Currently only supports Python and JS/TS

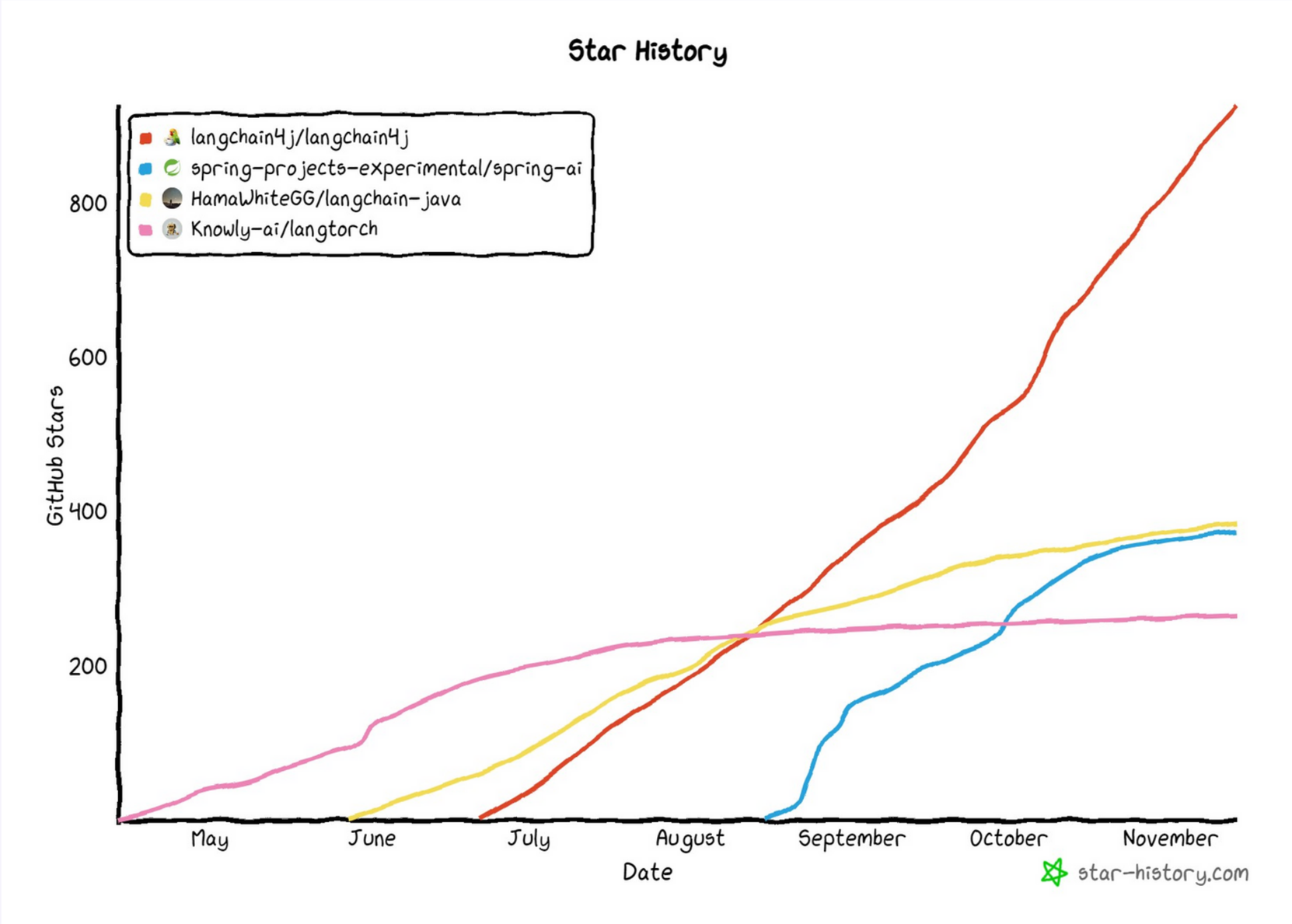
Langchain4j

Java version of Langchain, created by the Java community

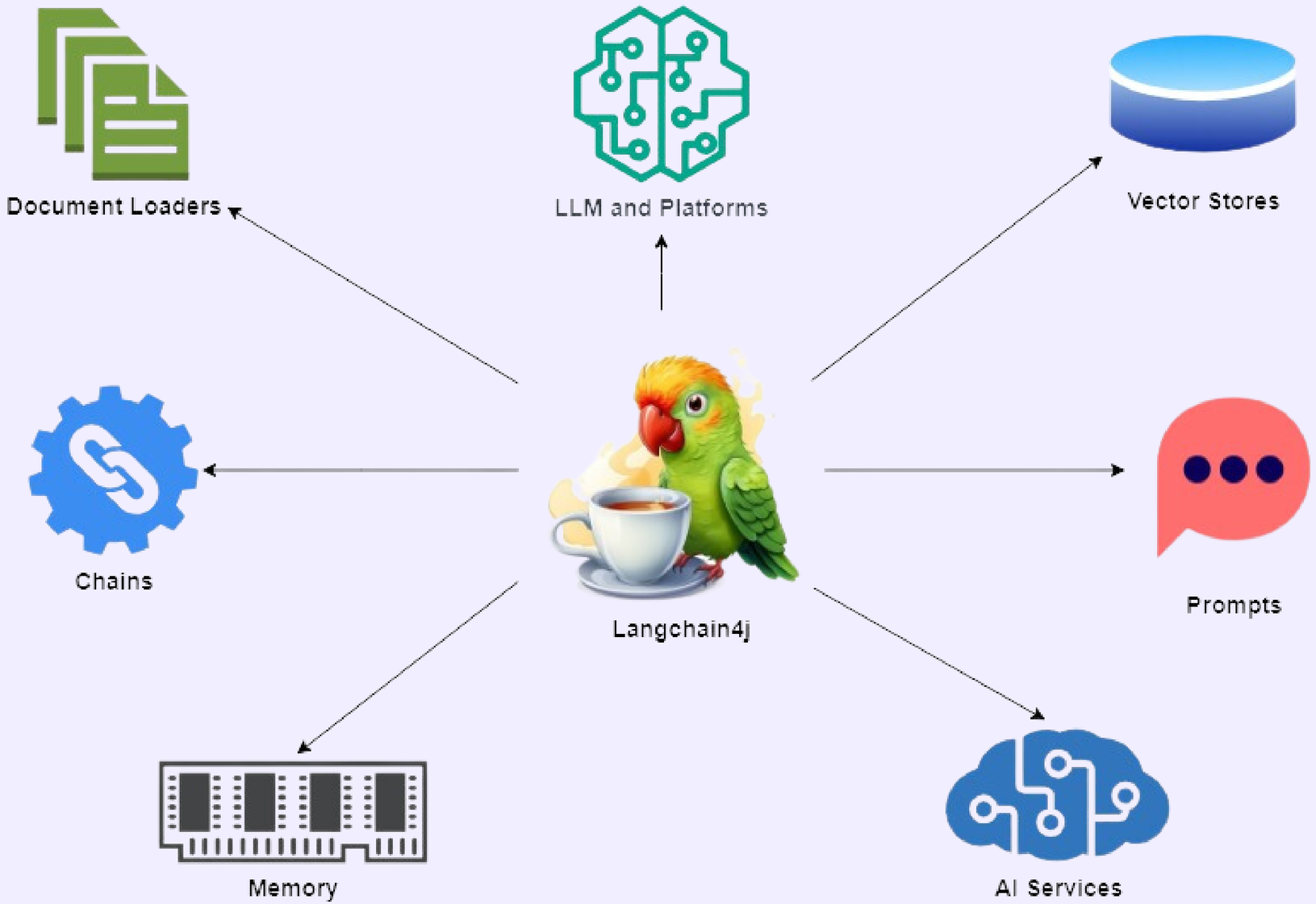
Has the same capability with Langchain but with additional and powerful features



Langchain4j is currently the leading standalone Java-based LLM framework on GitHub, as measured by the number of stars.



Components of Langchain4j



LLMs and Platforms

- Core component of Langchain4j
- Provides a standard interface for interacting with many different LLMs and LLM providers

OpenAI

Azure OpenAI

Hugging Face

Vertex AI

Qwen

Amazon Bedrock

LocalAI

Ollama

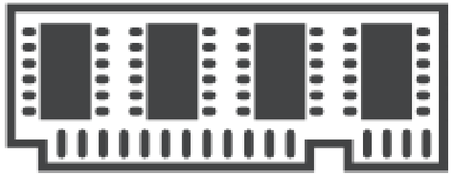
Prompts

- Act as the input or instruction that guides the model to generate a specific output or perform a particular task
- Supports application of user-input variables
- Supports annotations such as @UserMessage, @SystemMessage and @StructuredPrompt

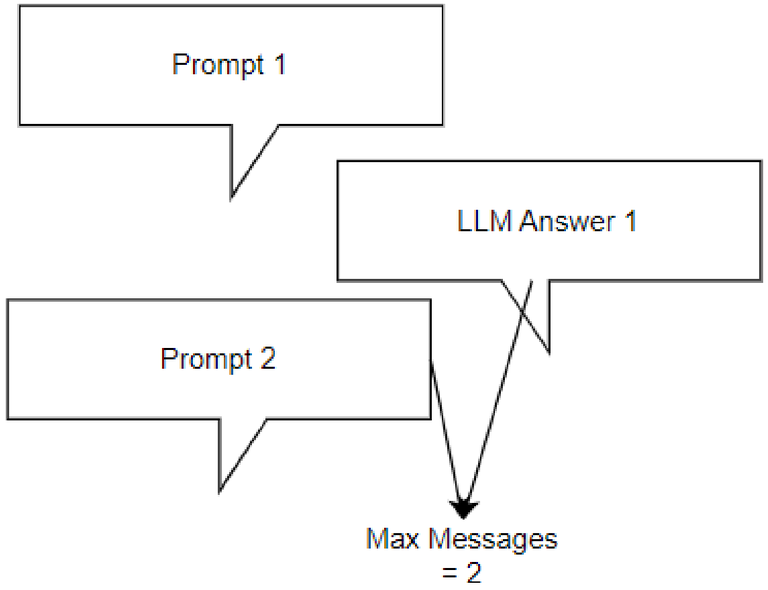
What are the features of Java version {{version}}?

Memory

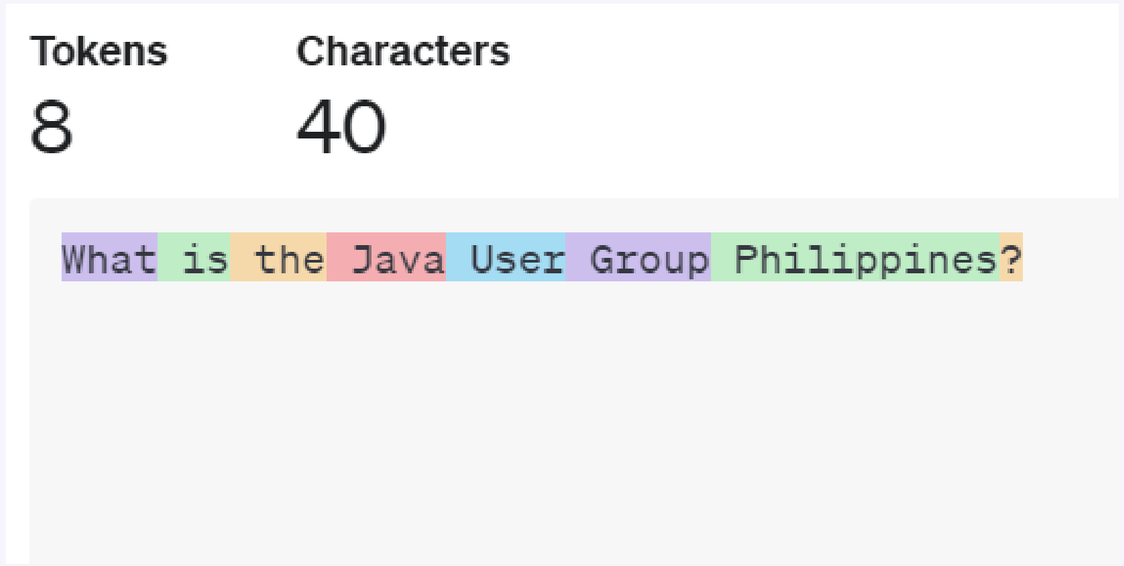
- Storage for previous prompts and responses
- Can be In-Memory or Persistent



Message-Window based



Token-Window based



Chains

- Reduces the need for extensive boilerplate code in common chat use-cases
- Best used for chat conversation use-cases
- Supports 2 conversation chats with memory and with RAG

Use Case: AI Chatbot

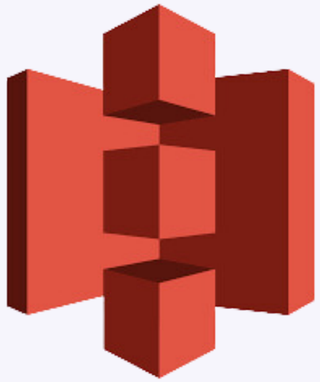
Document Loaders and Utils

- Allows you to load documents from numerous data sources
- Supports many document types such as PDF and MS Word

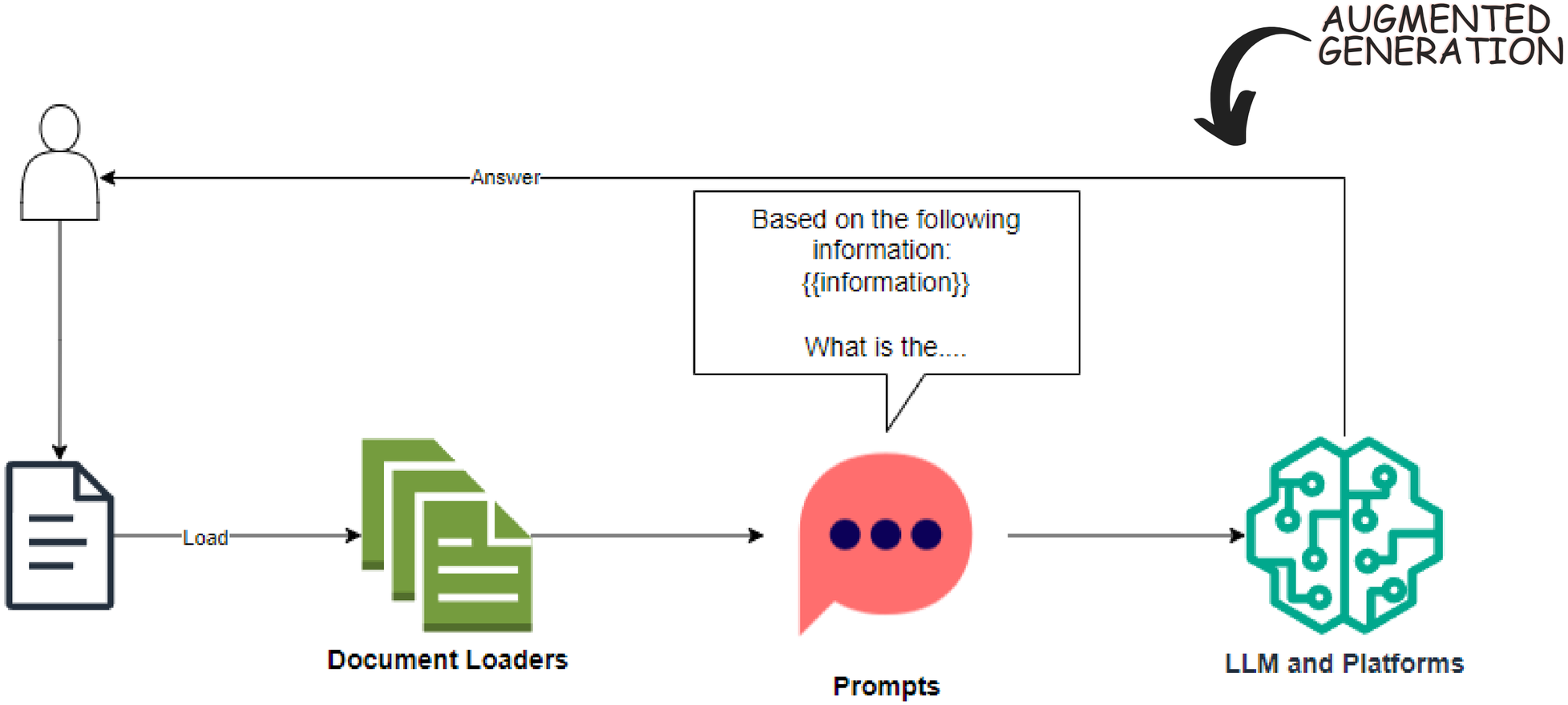
File System

URL

AWS S3



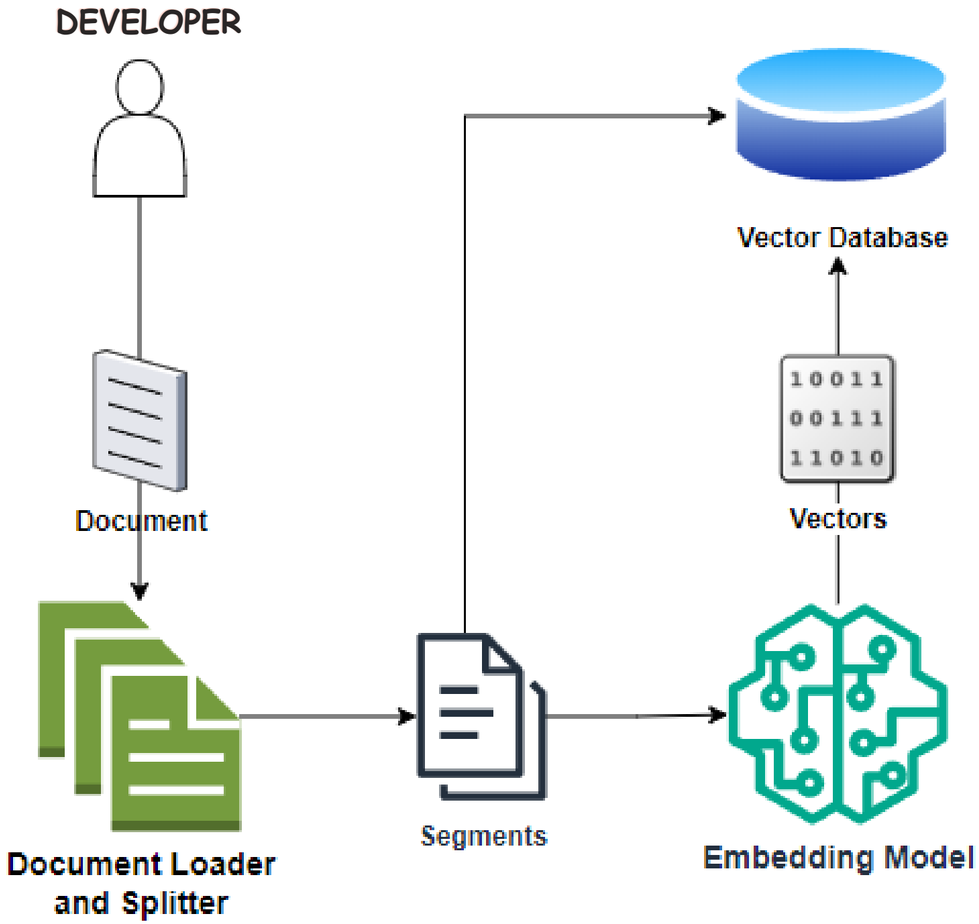
Use Case: Question-Answering with your own documents



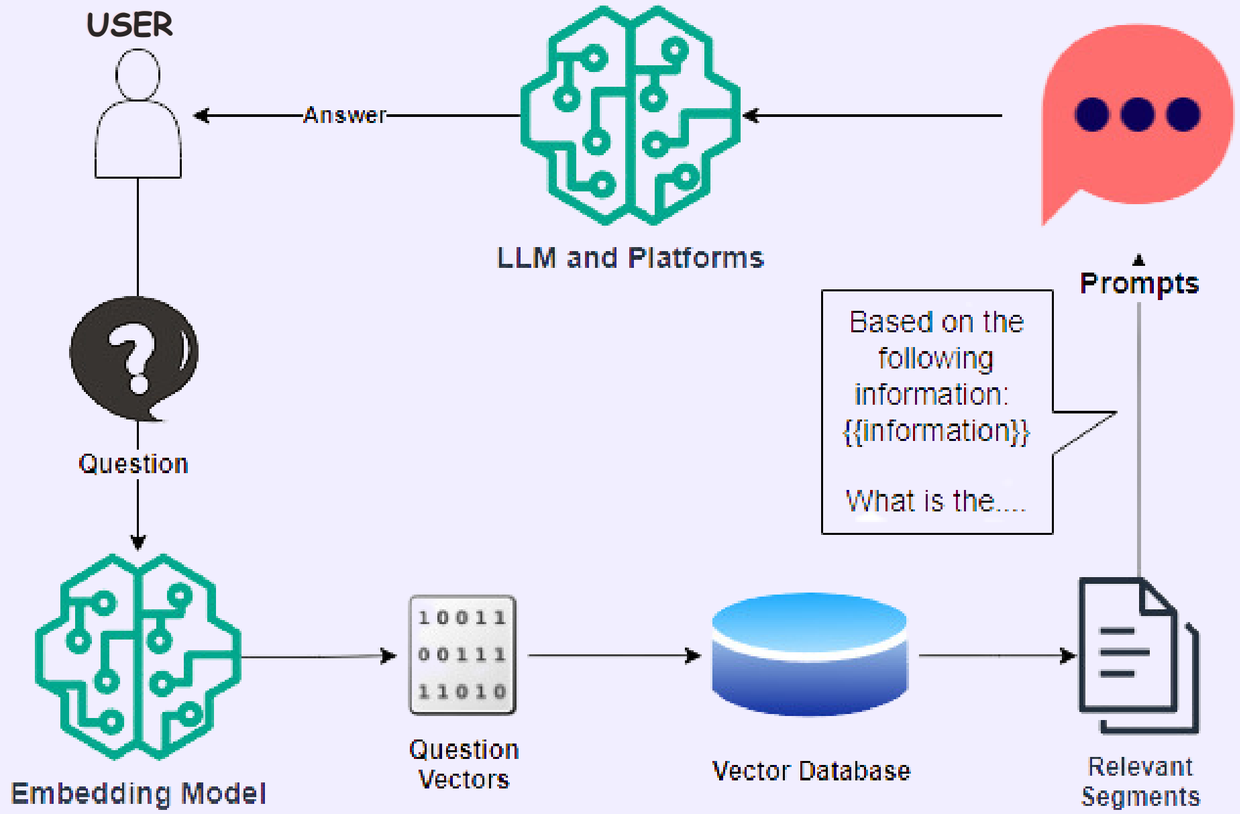
Question

How can all my documents fit in one prompt?

Ingestion

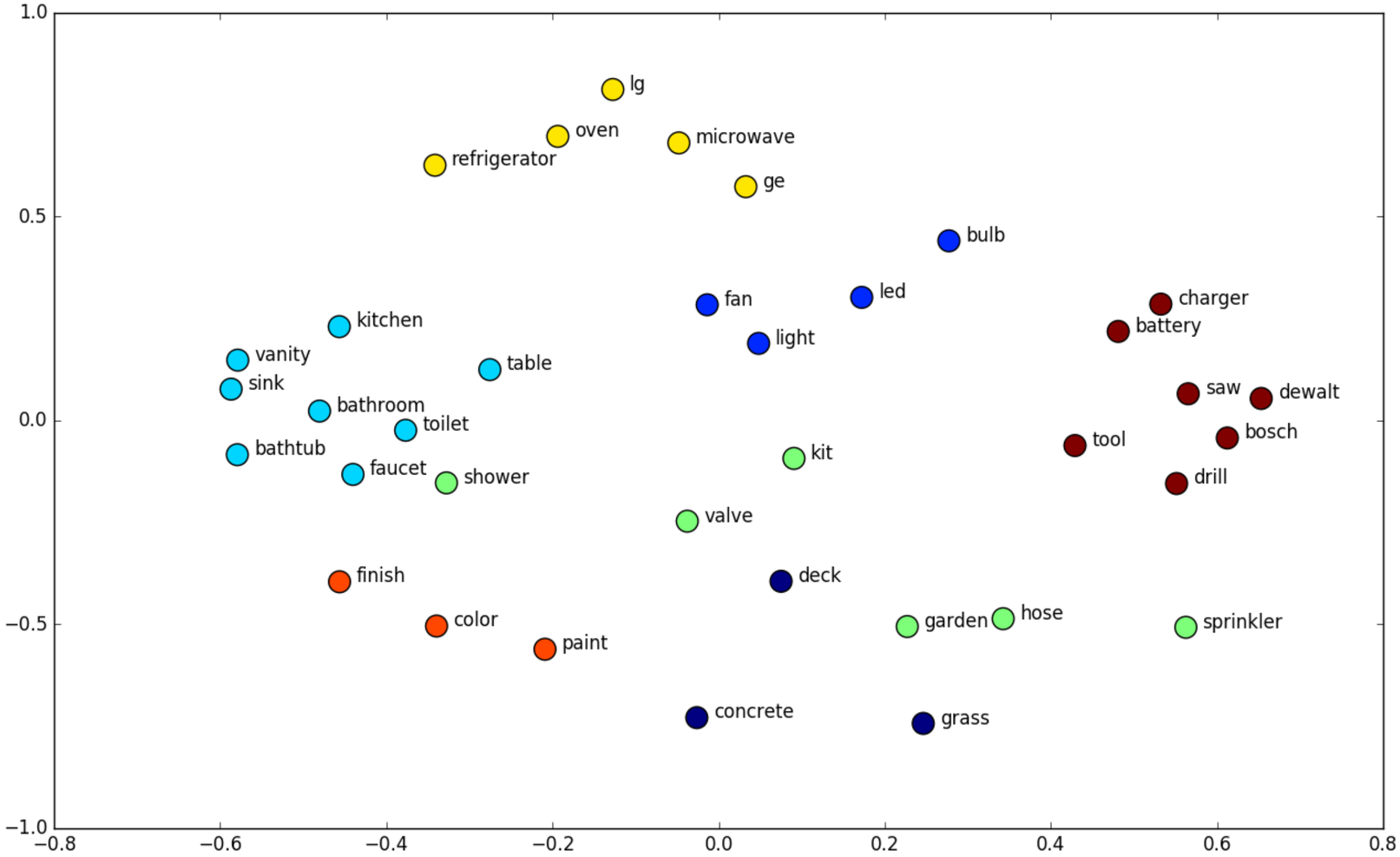


Retrieval



RETRIEVAL AUGMENTED GENERATION (RAG)

Sample Embedding Space



Taken from https://neptune.ai/blog/word-embeddings-guide

Supported Vector Stores

Weaviate

Chroma

AI Services

- Provide a simpler and more flexible alternative to chains
- Allows you to create complex, customizable AI interactions with an interface
- Data Type versatility
- Declarative and Annotation-based Syntax
- Can integrate with prompts, models, memory and tools.

Use Case: Member Extractor and Book Finder

AI Services Tools

- Delegate your own implemented tasks to the LLM
- Methods annotated with @Tool are considered tools that a language model can use.
- OpenAI Function Calls

Use Case:
Java News Retriever

Other Features

- Support for Java Frameworks
  - Spring Boot Starter
  - Quarkus Extension (quarkus-langchain4j-core)
- Moderation models
- Dynamic Tools
- Streaming
- Output Parsers

Features Coming Soon

- More Vector Stores
  - Neo4j
  - MongoDB
- More LLMs
- More Document Loaders
- Support for OpenAI Assistants API
- many more

News

12 November:

- Integration with OpenSearch by @riferrei
- Add support for loading documents from S3 by @jmgang
- Integration with PGVector by @kevin-wu-os
- Integration with Ollama by @Martin7-1
- Integration with Amazon Bedrock by @pascalconfluent
- Adding Memory Id to Tool Method Call by @benedictstrube
- And more

29 September:

- Updates to models API: return `Response<T>` instead of `T`. `Response<T>` contains token usage and finish reasor
- All model and embedding store integrations now live in their own modules
- Integration with Vespa by @Heezer
- Integration with Elasticsearch by @Martin7-1
- Integration with Redis by @Martin7-1
- Integration with Milvus by @IuriiKoval
- Integration with Astra DB and Cassandra by @clun
- Added support for overlap in document splitters

References

1. LangChain4j. (2023). LangChain4j. Retrieved from GitHub: https://github.com/langchain4j/langchain4j
2. Neptune AI. (2023). Word Embeddings Guide. Retrieved from Neptune AI Blog: https://github.com/langchain4j/langchain4j
3. Raes, L. (2023). Java Meets AI: A Hands On Guide to Building LLM Powered Applications with LangChain4j. Retrieved from YouTube: https://github.com/langchain4j/langchain4j

Do you have any questions?