Real-time AI made easy: Harnessing Generative AI Functions in Apache Kafka and Apache Pulsar

October, 2023
Rise of the GenAI

- ChatGPT was launched in November 2022, less than a year ago
- 2 talks at ApacheCon 2022: vector search only, not AI
- 16 talks including 2 keynotes at Community over Code 2023
- OpenAI valuation at $29Bn, aiming $90Bn
- Related Apache technologies
GenAI is changing the industry landscape
The barrier to AI

Traditional ML

Training phase

- Training data
- ML engineering
- Model

Inference phase

- Apply model
- Live data
- Result
The training phase costs $$$:
- Gathering, cleaning, storing, processing data
- Highly qualified, hard to find ML engineers / data scientists required
- Long process / TTM
- Not reusable, need 1 model per use-case
Lowering the barrier to AI

GPT (Large Language Model)

Training phase

Public Internet texts → ML engineering

Inference phase

Prompt → LLM → Chat completion → Next word/token
Lowering the barrier to AI

GPT (Large Language Model)

**Training phase**
- Public Internet texts
- ML engineering
- LLM

**Inference phase**
- Apply model
- Result

As of September 2021
Costs $$$$ but done only once by a SaaS company (eg. OpenAI)

Generic Large Language Model, reusable, tailored for human language
Using GenAI is just about building a prompt

- The problem to solve is not anymore in the model, it’s in the prompt.
- The quality of the description of the problem directly impacts the accuracy of the result.
- The LLM is static:
  - The live data must be included into the prompt
  - Information not present in the LLM training data must be included into the prompt
- The size of the prompt is limited
Anatomy of a prompt

Problem to Solve:
Eg: “Should we propose a reduction coupon to the user?”
Anatomy of a prompt

**Problem to Solve:**
Eg: “Should we propose a reduction coupon to the user?”

**Dynamic data:**
Eg: “user payment history, item price, item stock, …”

Aggregate structured data from multiple sources in real-time. Event-Driven Architectures are a perfect match for this.
Anatomy of a prompt

Problem to Solve:
Eg: “Should we propose a reduction coupon to the user?”

Dynamic data:
Eg: “user payment history, item price, item stock, ...”

Domain specific data (RAG):
Eg: “rules to propose a reduction”

Data to solve the problem that was not scraped during the training or that we want to give more focus on. For instance, new data after September 2021 or private data.
Anatomy of a prompt

Problem to Solve:
Eg: “Should we propose a reduction coupon to the user?”

Dynamic data:
Eg: “user payment history, item price, item stock, ...”

Domain specific data:
Eg: “rules to propose a reduction”

Limited size.
For OpenAI's gpt-3.5-turbo: 4097 tokens (token ~= word)
For OpenAI's gpt-3.5-turbo-16k: 16385 tokens
Problem: the domain specific data is huge and doesn’t fit into the LLM prompt.
What is a vector/embedding?

- An embedding model transforms a text into a vector called an embedding.
- The embedding can be N dimensions. For instance, OpenAI’s embeddings are 1536 dimensions.
- Similarity: $v_1$ is more similar to $v_2$ than $v_3$. This is a simple mathematical formula.

2 dimensions normalised vectors
What is a vector/embedding?

- The vector captures the essence of a word or a block of text within its context.
- The dimensions are the result of the LLM training.
Vector stores / vector databases

- Embeddings storage (with or without metadata depending on the DB)
- Built-in algorithms for fast retrieval of so-called “nearest-neighbors” embeddings (eg. HNSW, JVector, ...)
- Vectors are a new type of data supported in established databases (DataStax AstraDB, Cassandra, ...) and new specialized databases (Pinecone, Milvius, ...)
Let’s go back to the problem: our domain specific data is huge and doesn’t fit into the LLM prompt.
Retrieval Augmented Generation (RAG)

1. Domain specific data
2. Compute embeddings
3. Store embeddings
4. Prompt question
5. Compute embedding
6. Get nearest neighbors
7. Add to prompt
8. Get response

Embeddings model (eg OpenAI)
Vector Database (eg Cassandra)
LLM (eg OpenAI)
### Generative AI features

<table>
<thead>
<tr>
<th>Unstructured data</th>
<th>Structured data</th>
<th>AI operations</th>
<th>Source/Sink data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract text (eg. from PDF, HTML, MS doc)</td>
<td>Compute fields</td>
<td>Compute embeddings</td>
<td>Streaming system (Kafka, Pulsar, ...)</td>
</tr>
<tr>
<td>Normalise (lower case, trim spaces)</td>
<td>Drop fields</td>
<td>Store embeddings</td>
<td>Web site crawling</td>
</tr>
<tr>
<td>Detect/filter/tag language</td>
<td>Flatten</td>
<td>Perform vector search</td>
<td>Database</td>
</tr>
<tr>
<td>Split into chunks</td>
<td>Cast to types</td>
<td>Re-rank (MMR)</td>
<td>S3 / block storage</td>
</tr>
<tr>
<td></td>
<td>Merge key and value fields</td>
<td>Get chat completions</td>
<td>Micro-service</td>
</tr>
<tr>
<td></td>
<td>Unwrap key or value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Here comes LangStream
An Ideation for Kubernetes-native Kafka Connect

Posted at Sep 6, 2022

Kafka Connect, part of the Apache Kafka project, is a development framework and runtime for connectors which either ingest data into Kafka clusters (source connectors) or propagate data from Kafka into external systems (sink connectors). A diverse ecosystem of ready-made connectors has come to life on top of Kafka Connect, which lets you connect all kinds of data stores, APIs, and other systems to Kafka in a no-code approach.

With the continued move towards running software in the cloud and on Kubernetes in particular, it’s just natural that many folks also try to run Kafka Connect on Kubernetes. On first thought, this should be simple enough: just take the Connect binary and some connector(s), put them into a container image, and schedule it for execution on Kubernetes. As so often, the devil is in the details though: should you use Connect’s standalone or distributed mode? How can you control the lifecycle of specific connectors via the Kubernetes control plane? How to make sure different connectors don’t compete unfairly on resources such as CPU, RAM, or network bandwidth? In the remainder of this blog post, I’d like to explore running Kafka Connect on Kubernetes, what some of the challenges are for doing so, and how Kafka Connect could potentially be reimagined to become more "Kubernetes-friendly" in the future.
Composable Agents

Sources:
- Kafka
- Pulsar
- Kafka-Connect
- Python custom
- Web crawler
- S3
-...

Processors:
- Python custom
- Compute embeddings
- Chat completions
- Compute fields
- Drop fields
- Extract text
- Normalise text
- Detect language
- Split into chunks
- Query Vector DB
- Re-rank documents
-...

Sinks:
- Kafka
- Pulsar
- Kafka-Connect
- Python custom
- Vector DB
-...

1 pod
Declarative low-code

```yaml
topics:
- name: "input-topic"
  creation-mode: create-if-not-exists
- name: "output-topic"
  creation-mode: create-if-not-exists
- name: "history-topic"
  creation-mode: create-if-not-exists

pipeline:
- name: "convert-to-json"
  type: "document-to-json"
  input: "input-topic"
  configuration:
    text-field: "question"
- name: "ai-chat-completions"
  type: "ai-chat-completions"
  output: "history-topic"
  configuration:
    model: "${secrets.open-ai.chat-completions-model}"
    completion-field: "value.answer"
    log-field: "value.prompt"
    stream-to-topic: "output-topic"
    stream-response-completion-field: "value"
    min-chunks-per-message: 10
    messages:
      - role: user
        content: "You are a helpful assistant. Below you can find a question from the user. Please try to help them the best way you can.\n\n{value.question}"```

Example application:
- Reads from input-topic
- Asks OpenAI for chat completion
- Writes streamed answer chunks to output-topic
- Writes full answers to history-topic
Example application: RAG chatbot

**Pipeline 1:** Crawl Web site, chunk and store embeddings.

**Pipeline 2:** Get questions from Kafka, retrieve relevant chunks to answer the question, answer the question with OpenAI chat completions and output results to Kafka.

**Gateway:** provides Web socket endpoints over the Kafka topics.

Flow chart
Demo
Thank You

https://langstream.ai
https://github.com/LangStream/langstream
https://astra.datastax.com
cbornet_