

[Home](#) / [Blog](#) / [Langchain and Pathway: RAG Apps with always-up-to-date knowledge](#)

showcase

llm

Langchain and Pathway: RAG Apps with always-up-to-date knowledge

 Your email address**Szymon Dudycz**

Published May 18, 2024

Updated May 18, 2024

4 min read

[Table of Contents](#) >

You can now use Pathway in your RAG applications which enables always up-to-date knowledge from your documents to LLMs with Langchain integration.

Pathway is now available on [Langchain](#), a framework for developing applications powered by large language models (LLMs). You can now query Pathway and access up-to-date documents for your RAG applications from LangChain using [PathwayVectorClient](#).

With this new integration, you will be able to use Pathway Vector Store natively in LangChain. In this guide, you will have a quick dive into Pathway + LangChain to learn how to create a simple, yet powerful RAG solution.

Prerequisites

To work with LangChain you need to install `langchain` package, as it is not a dependence of Pathway. In the example in this guide you will also use `OpenAIEmbeddings` class for which you need `langchain_openai` package.

```
!pip install langchain
!pip install langchain_community
!pip install langchain_openai
```

Using LangChain components in Pathway Vector Store

When using Pathway [VectorStoreServer](#), you can use LangChain embedder and splitter for processing documents. To do that, use [from_langchain_components](#) class method.

To start, you need to create a folder Pathway will listen to. Feel free to skip this if you already have a folder on which you want to build your RAG application. You can also use Google Drive, Sharepoint, or any other source from [pathway-io](#).

```
!mkdir -p 'data/'
```

To run this example you also need to set OpenAI API key, or change the embedder.

```
import os
import getpass

# Set OpenAI API Key
if "OPENAI_API_KEY" in os.environ:
    api_key = os.environ["OPENAI_API_KEY"]
else:
    api_key = getpass.getpass("OpenAI API Key:")
```

To run the server use Pathway filesystem connector to read files from the `data` folder.

```
import pathway as pw

from pathway.xpacks.llm.vector_store import VectorStoreServer
from langchain_openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter

data = pw.io.fs.read(
    "./data",
    format="binary",
    mode="streaming",
    with_metadata=True,
)
```

And then pass them to the server, which will split them using `CharacterTextSplitter` and embed them using `OpenAIEmbeddings`, both from `LangChain`.

```
embeddings = OpenAIEmbeddings(api_key=api_key)
splitter = CharacterTextSplitter()

host = "127.0.0.1"
port = 8666

server = VectorStoreServer.from_langchain_components(
    data, embedder=embeddings, splitter=splitter
)
server.run_server(host, port=port, with_cache=True, cache_backend=pw.persis
```

The server is now running and ready for querying with a `VectorStoreServer` or with a `PathwayVectorClient` from `langchain-community` described in the next Section.

Using Pathway as a Vector Store in LangChain pipelines

Once you have a `VectorStoreServer` running you can access it from `LangChain` pipeline by using `PathwayVectorClient`.

To do that you need to provide either the `url` or `host` and `port` of the running `VectorStoreServer`. In the code example below, you will connect to the `VectorStoreServer` defined in the previous Section, so make sure it's running before making queries. Alternatively, you can also use a publicly available [demo pipeline](#) to test your client. Its REST API you can access at <https://demo-document-indexing.pathway.stream>. This demo ingests documents from [Google Drive](#) and [Sharepoint](#).

```
from langchain_community.vectorstores import PathwayVectorClient

client = PathwayVectorClient(host=host, port=port)

query = "What is Pathway?"
docs = client.similarity_search(query)
print(docs)
```

As you can see, the LLM cannot respond clearly as it lacks current knowledge, but this is where Pathway shines. Add new data to the folder Pathway is listening to, then ask our agent again to see how it responds. To do that, you can download the repo readme of Pathway into our data folder:

```
!wget 'https://raw.githubusercontent.com/pathwaycom/pathway/main/README.md'
```

Try again to query with the new data:

```
docs = client.similarity_search(query)
print(docs)
```

RAG pipeline in LangChain

The next step is to write a chain in LangChain. The next example implements a simple RAG, that given a question, retrieves documents from Pathway Vector Store. These are then used as a context for the given question in a prompt sent to the OpenAI chat.

```
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.runnables import RunnablePassthrough
from langchain_openai import ChatOpenAI

retriever = client.as_retriever()

template = """
You are smart assistant that helps users with their documents on Google Drive.
Given a context, respond to the user question.
CONTEXT:
{context}
QUESTION: {question}
YOUR ANSWER: """

prompt = ChatPromptTemplate.from_template(template)
llm = ChatOpenAI()
chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
```

Now you have a RAG chain written in LangChain that uses Pathway as its Vector Store. Test it by asking some question.

```
chain.invoke("What is Pathway?")
```

Vector Store statistics

Just like `VectorStoreClient` from the Pathway LLM xpack, `PathwayVectorClient` gives you two methods for getting information about indexed documents.

The first one is `get_vectorstore_statistics` and gives essential statistics on the state of the vector store, like the number of indexed files and the timestamp of the last updated one. The second one is `get_input_files`, which gets the list of indexed files along with the associated metadata.

```
print(client.get_vectorstore_statistics())  
print(client.get_input_files())
```



Szymon Dudycz

Algorithm and Data Processing Magician



Power your RAG and ETL pipelines with Live Data

Get started for free

Related Articles

Building In-House RAG for Scale

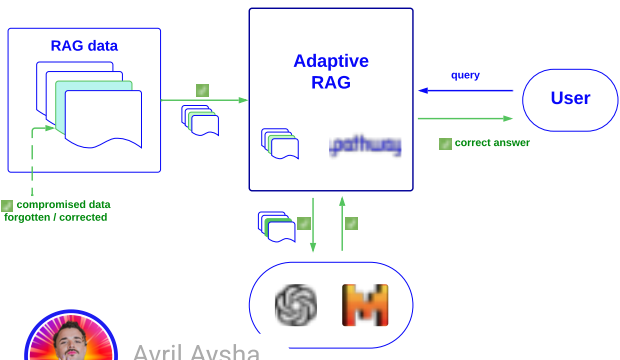
Jayprasad Hegde
Head Data Science and AI Initiatives – NPCI

1 HOUR MASTERCLASS

 Pathway Team

BLOG · CASE-STUDY APR 29, 2024

Building End-to-End RAG with NPCI’s AI Leader



Avril Aysha

BLOG APR 19, 2024

Machine Unlearning for LLMs: Build Apps that Self-Correct in Real-Time

Power and Deploy RAG Agent Tools with

 Saksham Goel

BLOG JAN 16, 2025

Power and Deploy RAG Agent Tools with Pathway

Blog

← Multimodal RAG with ...

Blog

LlamaIndex and Pathway: RAG Apps with always-up-to-dat... →



40k

Product

Pathway Framework

RAG Templates

[Why Pathway?](#)

[Pricing](#)

[Get Free Enterprise Features](#)

[Pathway Templates](#)

Resources

[Developer Docs](#)

[Success Stories](#)

[Bootcamps](#)

[Blog](#)

[Events](#)

Company

[Team](#)

[Careers](#)

[Newsroom](#)

[Media kit](#)

[Licensing Terms](#)

[Policies](#)

Contact

Let's talk 

Chat with us on Discord 

Pathway

418 Florence Street

Palo Alto, CA 94301

© 2021-2025 Pathway