Hands on lab: Cómo crear aplicaciones de IA Generativa con LLMs

26 octubre 2023

diverger

Índice

03 ejemplo práctico implementación sistemas RAG

RAG

Intro

La **Retrieval-Augmented Generation** se refiere a un enfoque de procesamiento de lenguaje natural (NLP) y aprendizaje automático que combina elementos de modelos de recuperación y modelos de generación para mejorar la calidad y relevancia del texto generado.

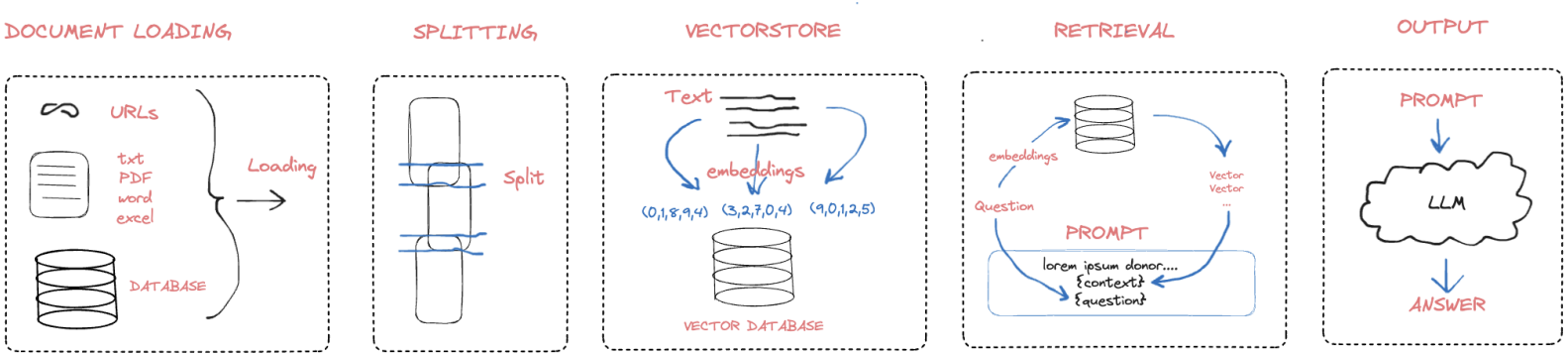
Los modelos de recuperación se centran en encontrar información relevante en un conjunto predefinido de documentos o fuentes de datos.

Los modelos de generación son capaces de generar texto parecido al humano desde cero o completar frases basadas en un contexto dado

Esta técnica se utiliza a menudo para tareas como generación de texto, respuesta a preguntas y resumen de contenido. Combina las fortalezas de los métodos basados en recuperación y los métodos basados en generación.

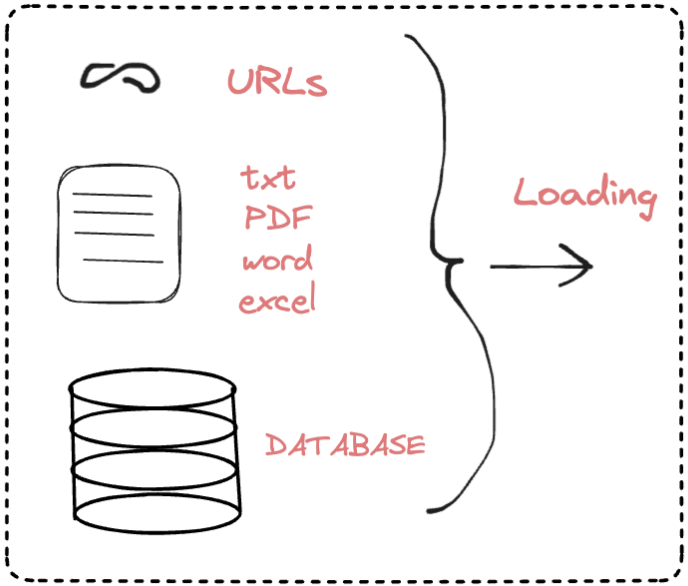
RAG

Steps


DOCUMENT LOADING
URLs
txt
PDF
word
excel
Loading
DATABASE
SPLITTING
Split
VECTORSTORE
Text
embeddings
(0,1,8,9,4) (3,2,7,0,4) (9,0,1,2,5)
VECTOR DATABASE
RETRIEVAL
embeddings
Vector
Vector
...
Question
PROMPT
lorem ipsum donor....
{context}
{question}
OUTPUT
PROMPT
LLM
ANSWER

RAG

Document loading

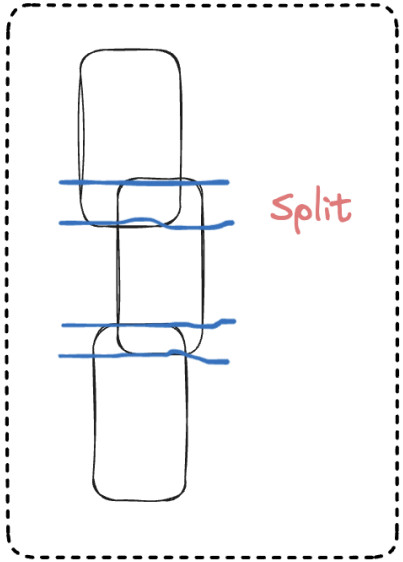



```
import { DirectoryLoader } from "langchain/document_loaders/fs/directory";
import { TextLoader } from "langchain/document_loaders/fs/text"
import { PDFLoader } from "langchain/document_loaders/fs/pdf";

const loader = new DirectoryLoader(
  'documents',
  {
    ".txt": (path) => new TextLoader(path),
    ".pdf": (path) => new PDFLoader(path)
  }
)
```

RAG

Splitting

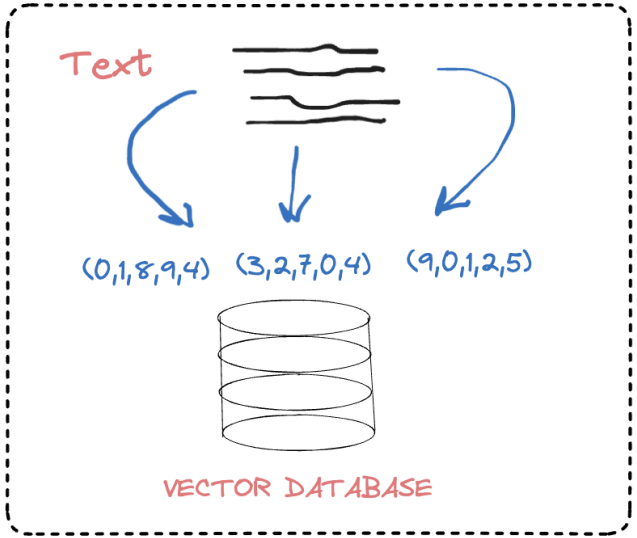



```
import { RecursiveCharacterTextSplitter } from "langchain/text_splitter";

const splitter = new RecursiveCharacterTextSplitter({
  chunkSize: 1000,
  chunkOverlap: 200,
});
```

RAG

Vectorstore

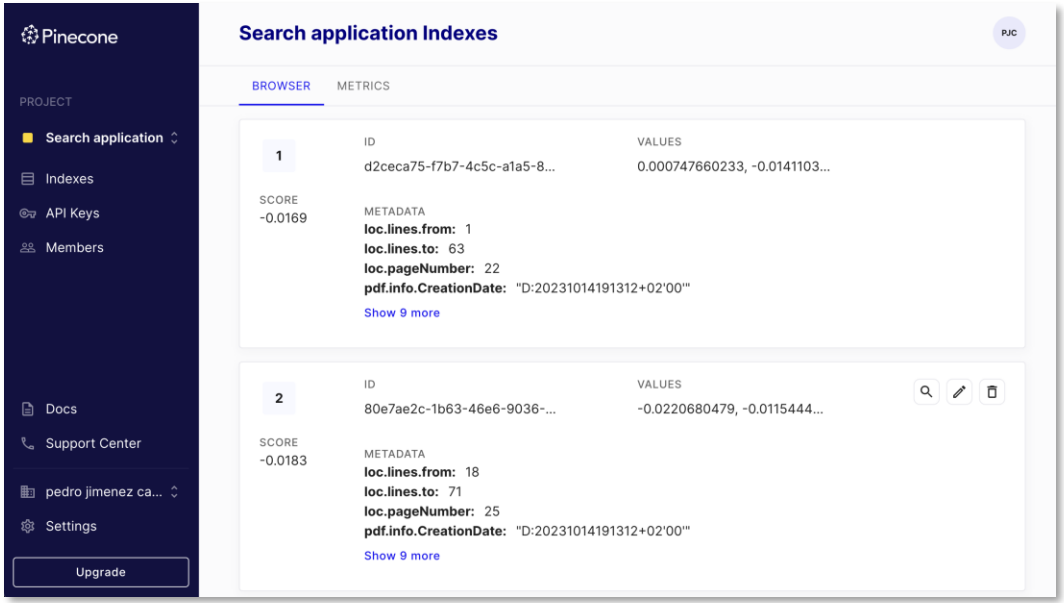



```
import { PineconeClient } from "@pinecone-database/pinecone";
import { PineconeStore } from "langchain/vectorstores/pinecone";
import { OpenAIEmbeddings } from "langchain/embeddings/openai";

await PineconeStore.fromDocuments(document, new OpenAIEmbeddings(), {
  pineconeIndex,
  // namespace: <namespace-name>,
})
```

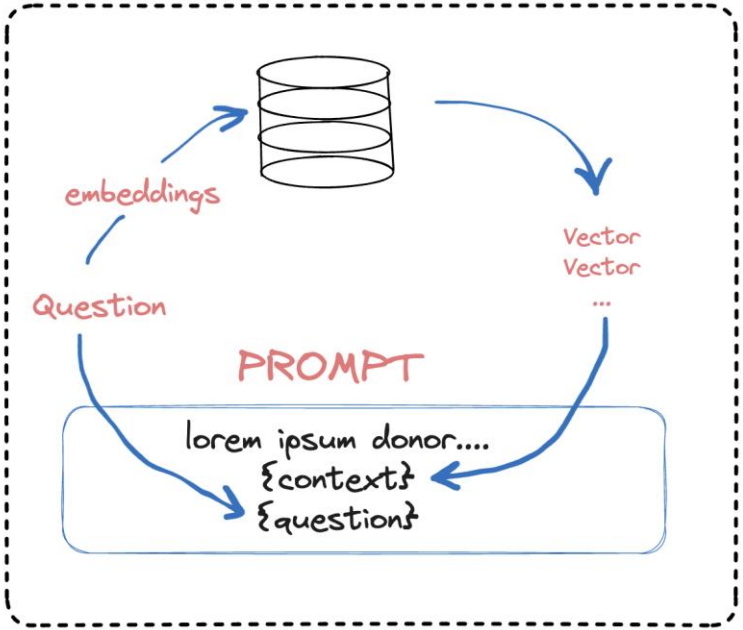
RAG

Vectorstore



RAG

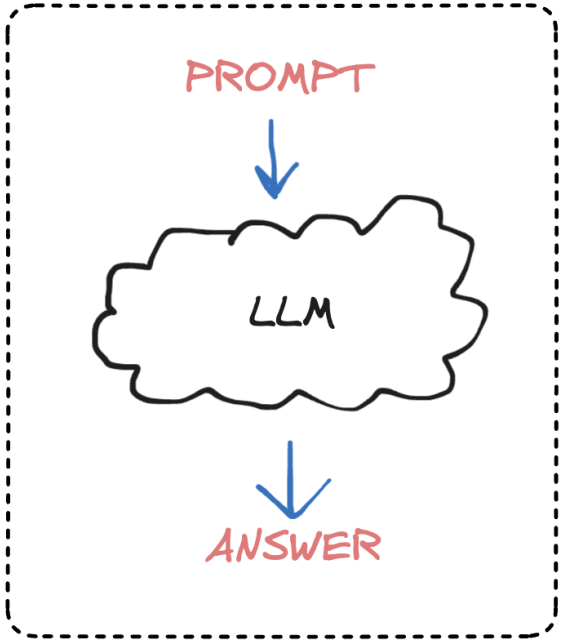
Retrieval




```
const vectorStore = await PineconeStore.fromExistingIndex(
  new OpenAIEmbeddings(),
  {
    pineconeIndex: index,
    filter: {}
  }
);
const retriever = vectorStore.asRetriever();
const chain = await makeChain(retriever);
```

RAG

Output




```
const questionPrompt = buildQuestionPrompt();
const question = await questionPrompt.format({
  question: req.body.question,
})

const answer = await chain.call({ question, chat_history: [] });
```

RAG

Docs

LangChain

NodeJS

Installation | 🦜🔗 Langchain

Python

Introduction | 🦜🔗 Langchain

Pinecone

Pinecone

OpenAI

Introduction - OpenAI API

¡Gracias!

diverger