

How do domain-specific chatbots work?

An overview of Retrieval Augmented Generation (RAG)



Cory Zue | @czue
Founder of SaaS Pegasus and Scriv.ai

What are we talking about?

The last six months...

Custom
for you

Just connect your data source
for your data. Then add it as a
with it through our integration

POWERED BY OPENAI

ChatGPT
Answers

Instantly answer your cl
with a chatbot trained o

Create Your Chatbot Now

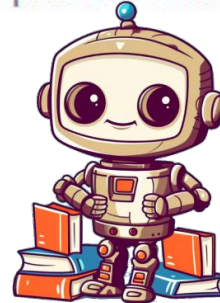
A private chatbot for your company data

Create a chatbot with deep knowledge of your private data
in just minutes.

Get started

Sign in

scriv.ai



er your
wledge

tbot

ed with your website.

Start for Free – No Credit Card required


LangChain



```
1 from langchain.document_loaders import WebBaseLoader
2 from langchain.indexes import VectorstoreIndexCreator
3 loader = WebBaseLoader("http://www.paulgraham.com/greatwork.html")
4 index = VectorstoreIndexCreator().from_loaders([loader])
5 index.query("What should I work on?")
```

A RAG bot

 What should I work on?

 Deciding what to work on can be a complex process, especially when you're young and might not know what you're good at or what different kinds of work are like. However, there are a few guidelines you can follow. First, work on something that you have a natural aptitude for and a deep interest in. Second, develop a habit of working on your own projects, especially those that seem excitingly ambitious to you. Lastly, when in doubt, optimize for interestingness. Try lots of things, meet lots of people, read lots of books, ask lots of questions. It's also important to remember that it's okay to guess and be wrong sometimes. Some of the biggest discoveries come from noticing connections between different fields.

Sources

- [How to Do Great Work](#)

PAUL GRAHAM

HOW TO DO GREAT WORK

July 2023

If you collected lists of techniques for doing great work in a lot of different fields, what would the intersection look like? I decided to find out by making it.

Partly my goal was to create a guide that could be used by someone working in any field. But I was also curious about the shape of the intersection. And one thing this exercise shows is that it does have a definite shape; it's not just a point labelled "work hard."

The following recipe assumes you're very ambitious.

The first step is to decide what to work on. The work you choose needs to have three qualities: it has to be something you have a natural aptitude for, that you have a deep interest in, and that offers scope to do great work.

In practice you don't have to worry much about the third criterion. Ambitious people are if anything already too conservative about it. So all you need to do is find something you have an aptitude for and great interest in. [1]

That sounds straightforward, but it's often quite difficult. When you're young you don't know what you're good at or what different kinds of work are like. Some kinds of work you end up doing may not even exist yet. So while some people know what they want to do at 14, most have to figure it out.

The way to figure out what to work on is by working. If you're not sure what to work on, guess. But pick something and get going. You'll probably guess wrong some of the time, but that's fine. It's good to know about multiple things; some of the biggest discoveries come from noticing connections between different fields.

Develop a habit of working on your own projects. Don't let "work" mean something other people tell you to do. If you do manage to do great work one day, it will probably be on a project of your own. It may be within some bigger project, but you'll be driving your part of it.

You can try this here if you want: <https://scriv.ai/a/scriv/bots/b/pg-bot/chat/>

For comparison with normal GPT4



What should I work on?



As an AI, I don't have personal knowledge about your specific circumstances, skills, or interests. However, here are a few general suggestions:

1. **Personal Development:** Focus on improving your skills or learning new ones.
2. **Health and Fitness:** Adopt a healthy lifestyle, exercise regularly, and maintain a balanced diet.
3. **Relationships:** Spend quality time with family and friends.
4. **Career Goals:** Work towards advancing in your career or starting a new one.
5. **Hobbies:** Pursue activities that you enjoy and are passionate about.

How does this work?

What it isn't

Chatting with a document

Model training

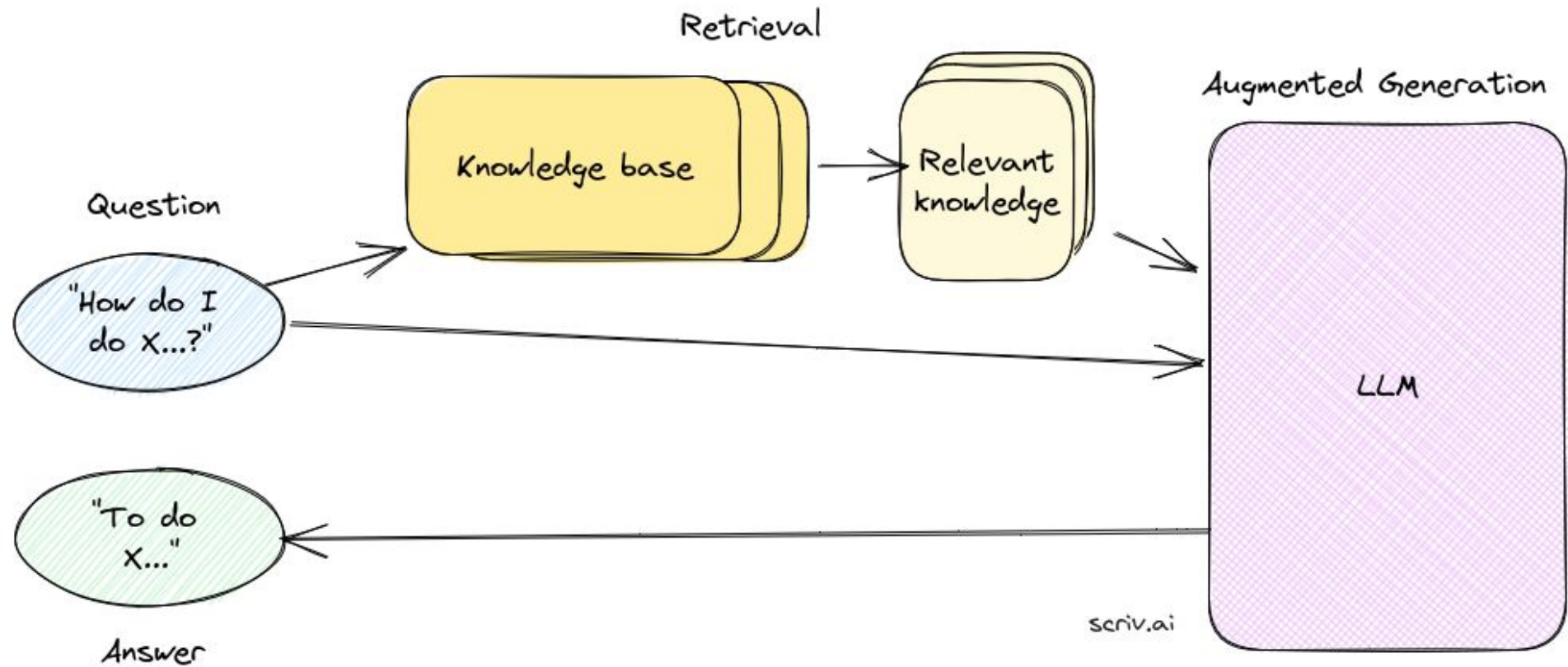
Fine-tuning

(These things are all good and useful, just not what this talk is about).

Retrieval Augmented Generation

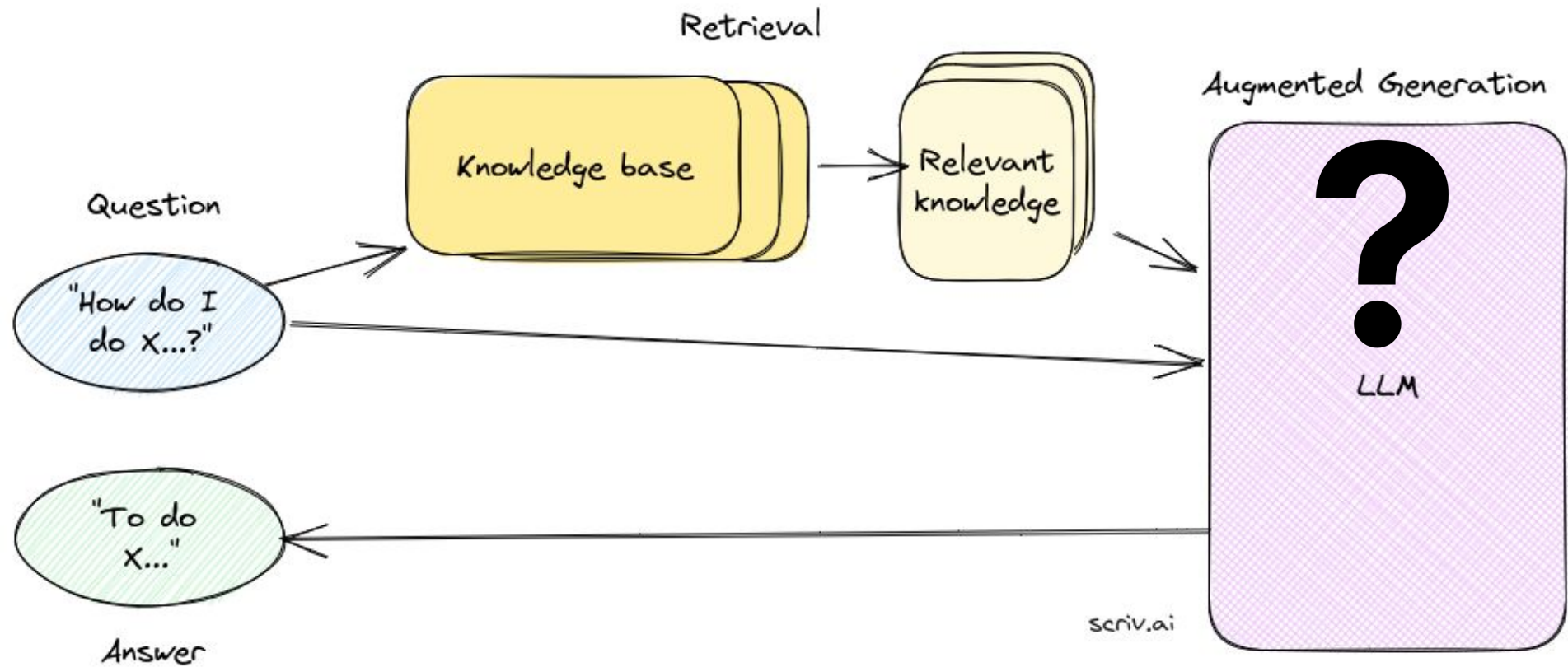
Retrieval augmented generation, or **RAG** is the process of supplementing a user's input to a large language model (LLM) like ChatGPT with additional information that you have *retrieved* from somewhere else. The LLM can then use that information to *augment* the response that it *generates*.

Retrieval Augmented Generation Overview



Augmented Answer Generation

Retrieval Augmented Generation Overview



Anatomy of ChatGPT

Playground

Your presets

Save

View code

Share

...

SYSTEM

You are a helpful assistant.

System Prompt

USER

Enter a user message here.

+ Add message

Messages

Submit



Mode

Chat

Model

gpt-3.5-turbo

Temperature

1

Maximum length

256

(Model Params)

Stop sequences

Enter sequence and press Tab

Top P

1

Frequency penalty

0

Presence penalty

0

API and Playground requests will not be used to train our models. [Learn more](#)

Anatomy of Augmented Answers

Playground

Your presets

Save

View code

Share

...

SYSTEM

You are a Knowledge Bot. You will be given the extracted parts of a knowledge base (labeled with DOCUMENT) and a question. Answer the question using information from the knowledge base.

Explain the Task

USER

----- DOCUMENT 1 -----

This document describes the blah blah blah...

----- DOCUMENT 2 -----

This document is another example of using x, y and z...

----- DOCUMENT 3 -----

[more documents here...]

Documents

USER

How do I do X?

Question

⊕ Add message

Submit



Mode

Chat

Model

gpt-3.5-turbo

Temperature

1

Maximum length

256

Stop sequences

Enter sequence and press Tab

Top P

1

Frequency penalty

0

Presence penalty

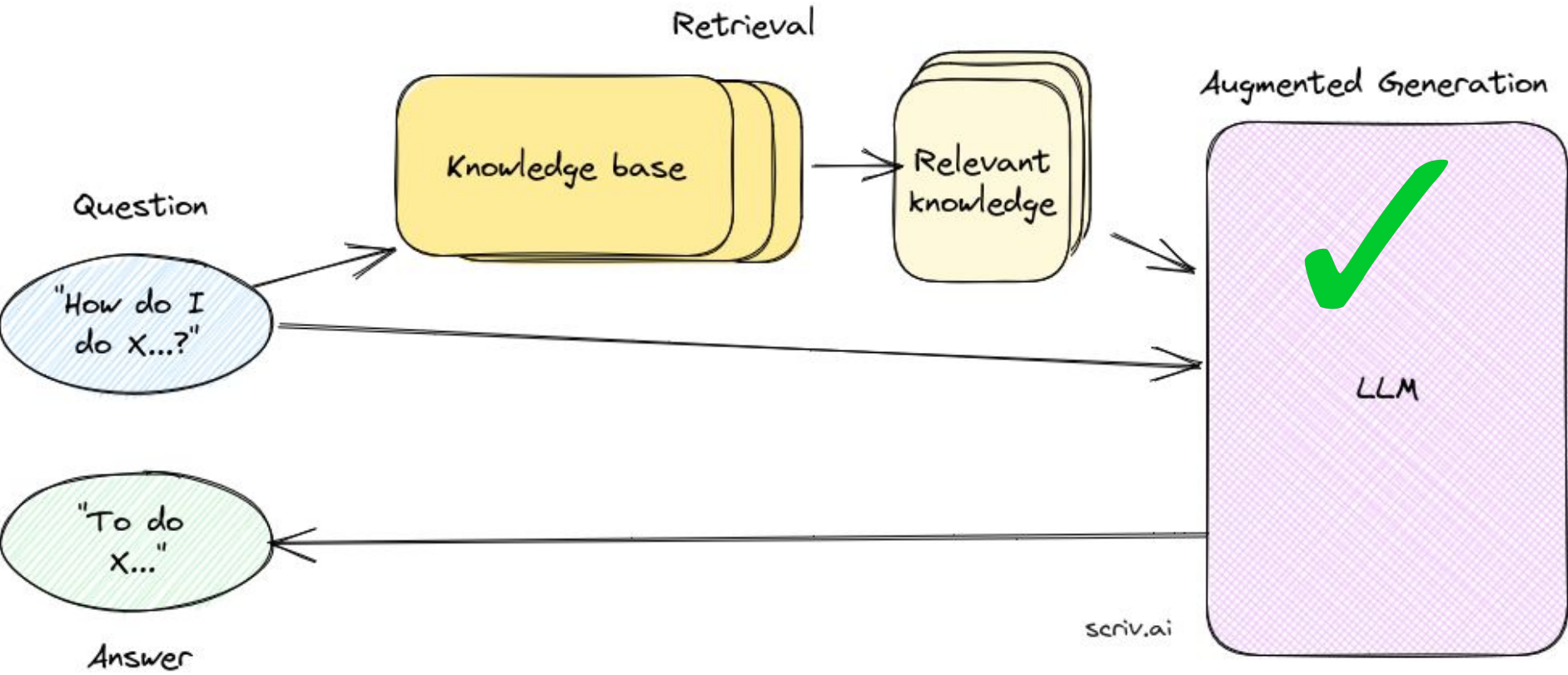
0

🔒 API and Playground requests will not be used to train our models. [Learn more](#)

Or in code

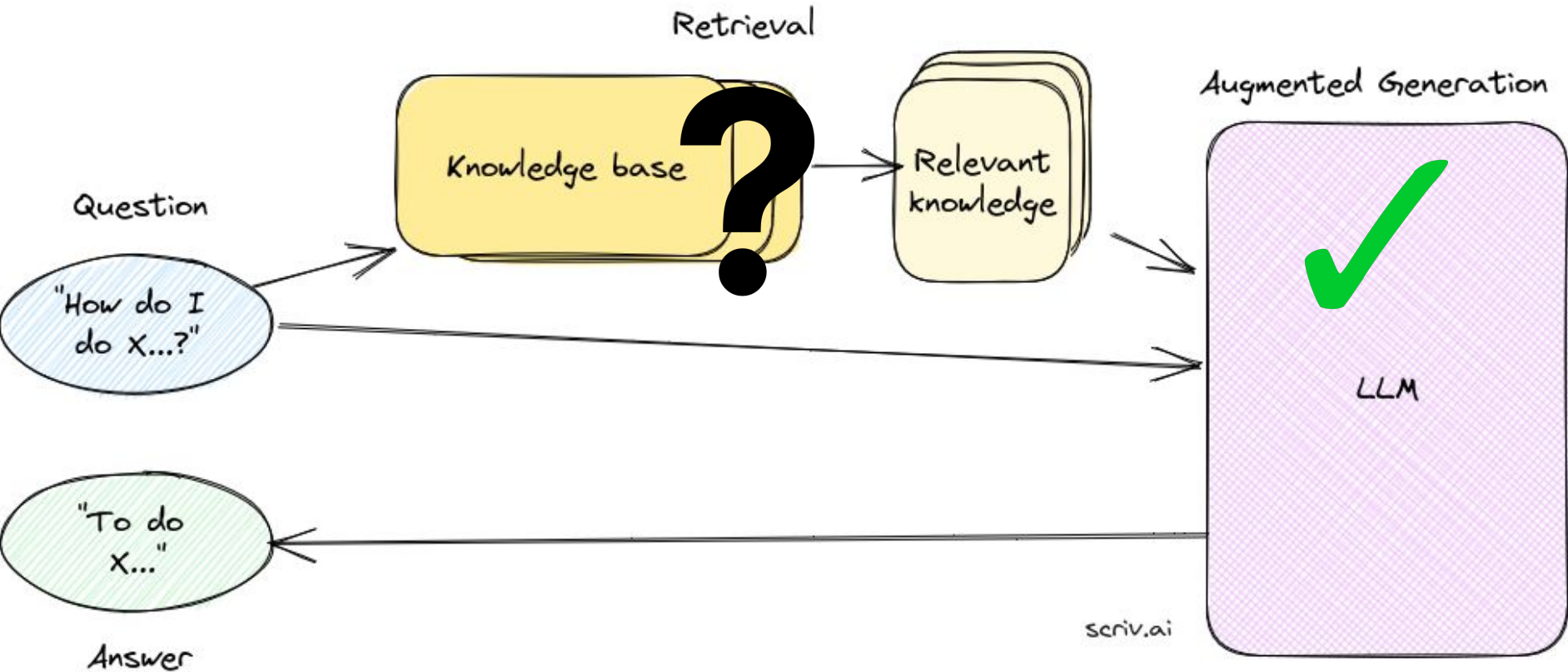
```
1  openai_response = openai.ChatCompletion.create(
2      model="gpt-3.5-turbo",
3      messages=[
4          {
5              "role": "system",
6              "content": get_system_prompt(), # the system prompt as per above
7          },
8          {
9              "role": "system",
10             "content": get_sources_prompt(), # the formatted documents as per above
11         },
12         {
13             "role": "user",
14             "content": user_question, # the question we want to answer
15         },
16     ],
17 )
```

Back to the overview

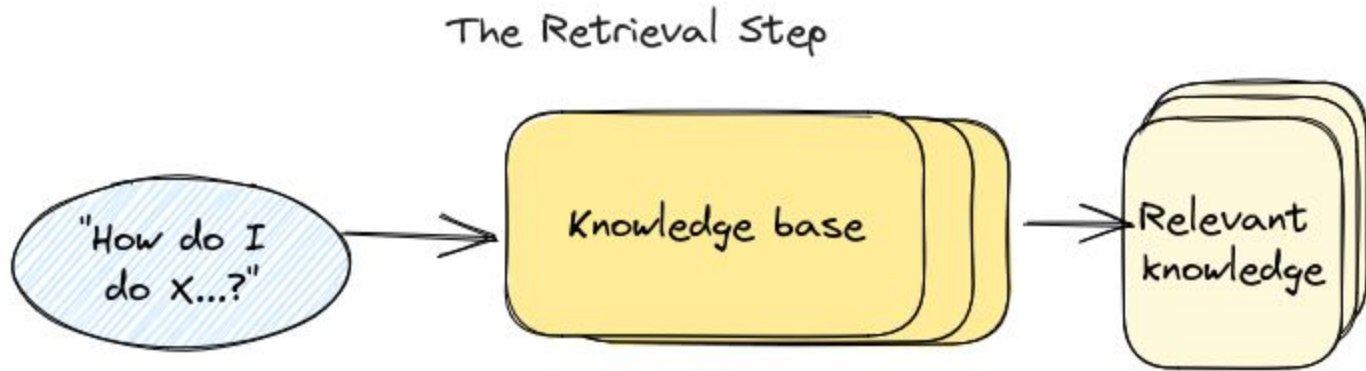


Retrieval

Back to the overview



The Retrieval Step



Retrieval is Search

Two big picture pieces:

1. **Indexing:** Turning your knowledge base into something that can be searched/queried.
2. **Querying:** Pulling out the most relevant bits of knowledge from a search term.

Any search operation works, but most RAG systems use something called semantic search, which uses another core piece of AI technology: *embeddings*.

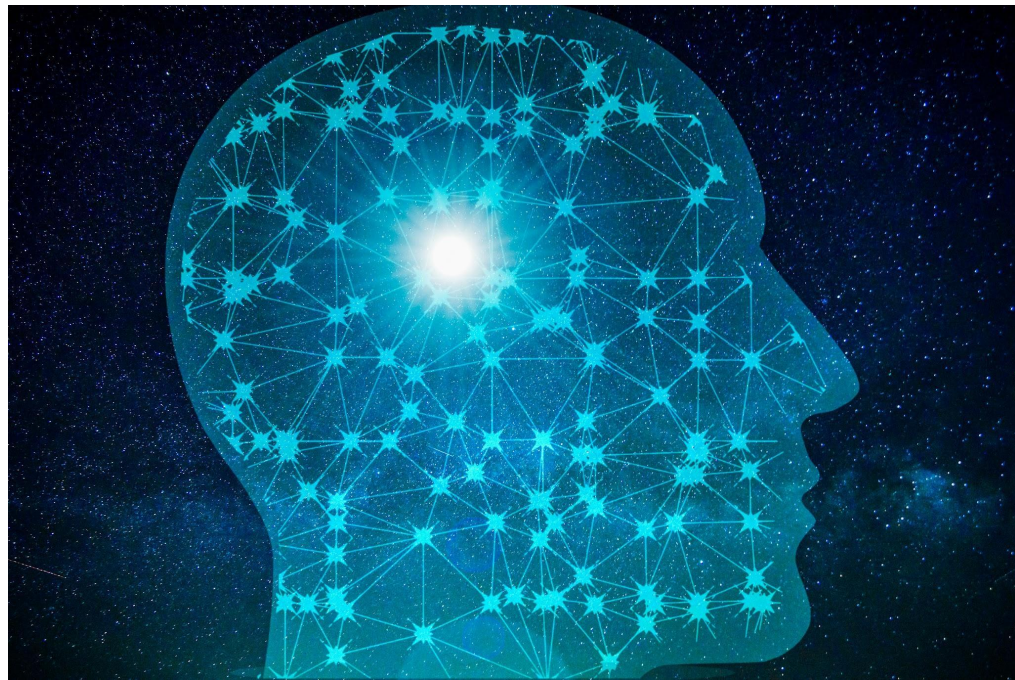
A Detour: Embeddings

How do we know what words mean?

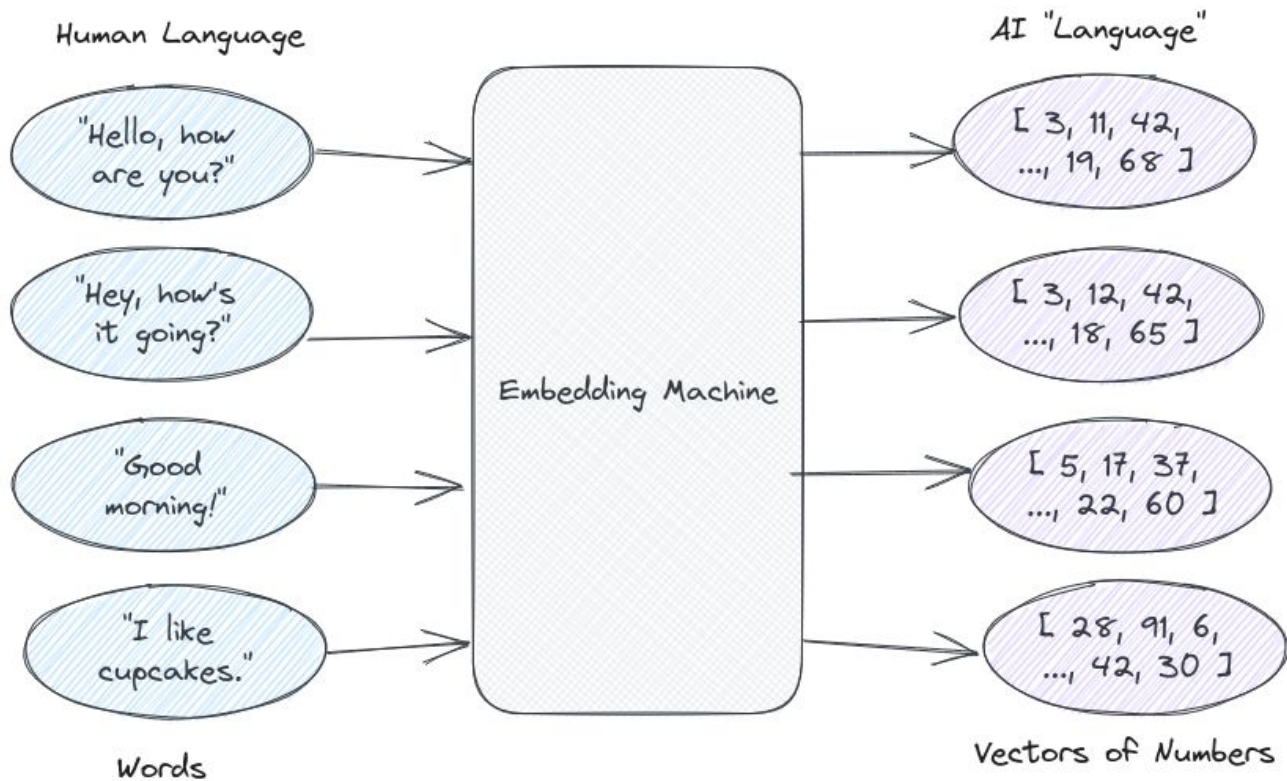
“Child” == “Kid”

“Red”, “Green”, “Blue” \in {“Colors”}

“Pleased” < “Happy” < “Elated”



Embeddings: a computer's version of "meaning"



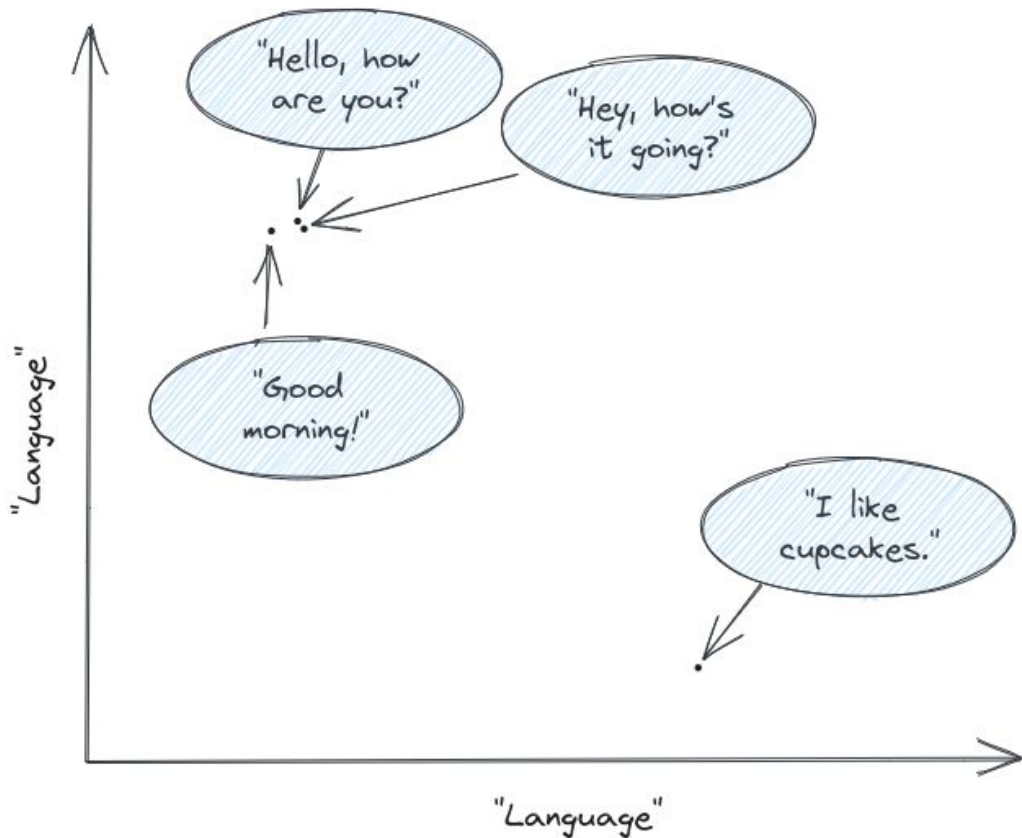
Embeddings: a computer's version of "meaning"

We don't know what these numbers mean.

But we can plot them.

And the closer they are the more "similar" they are.

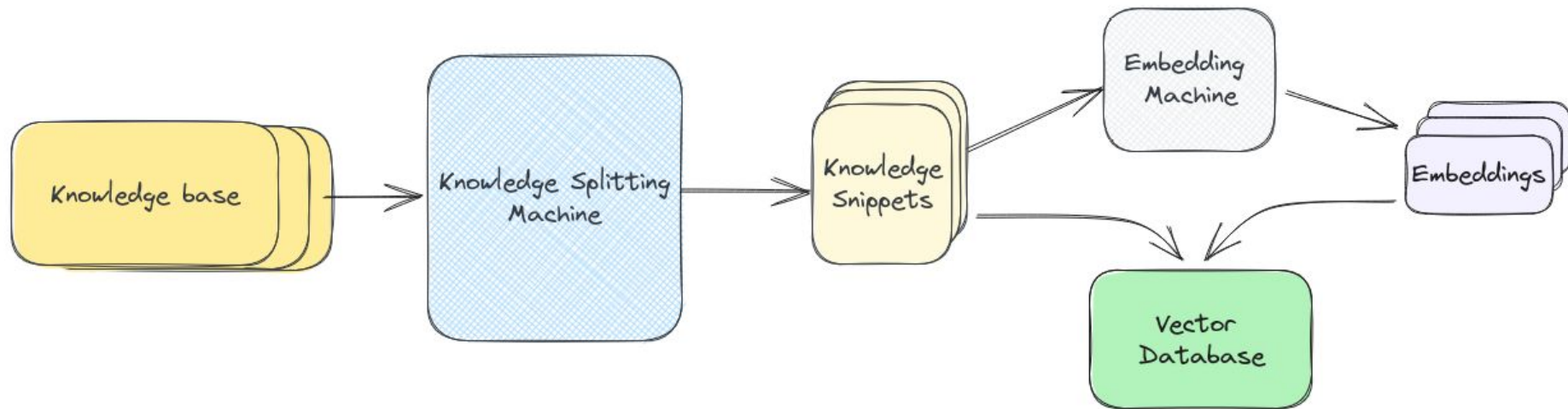
This is "semantic search"



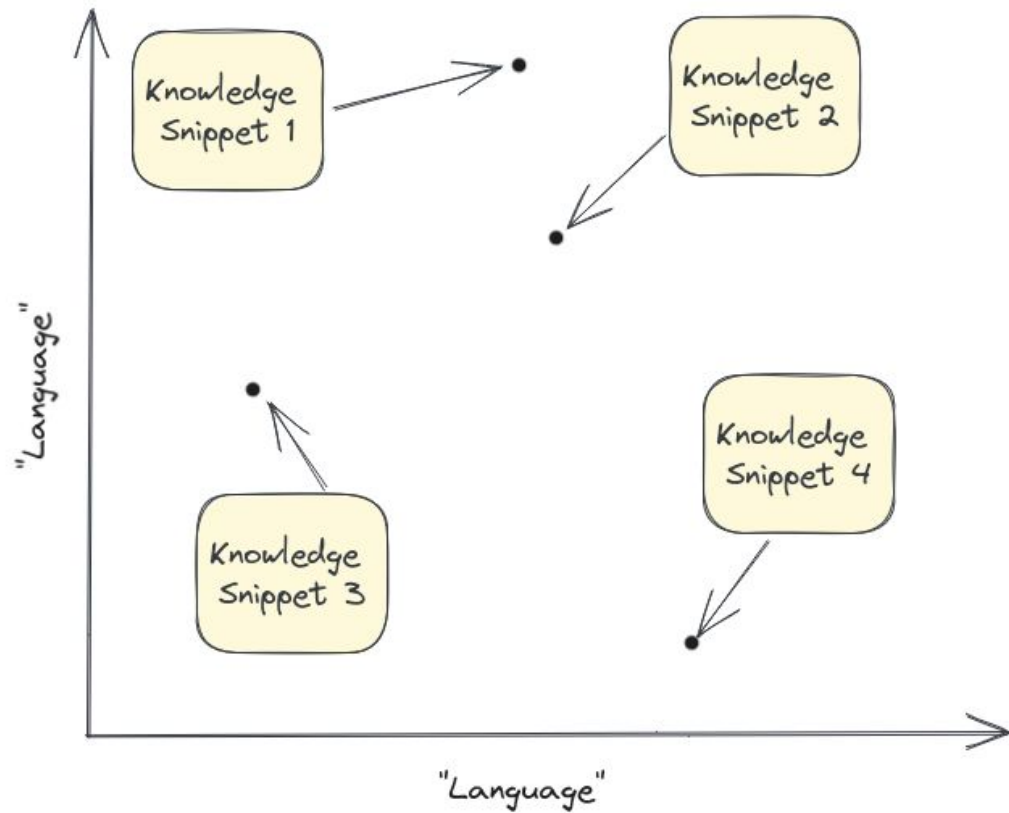
Retrieval with Embeddings (Indexing)

Assume we have a way to split up our knowledge base (we'll come back to this)

Then pass each snippet through our embedding machine and save it, with its embedding in a *vector database*.

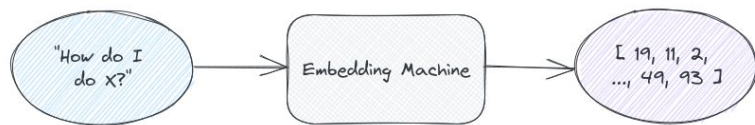


Our Knowledge Base in Embedding/Vector Space

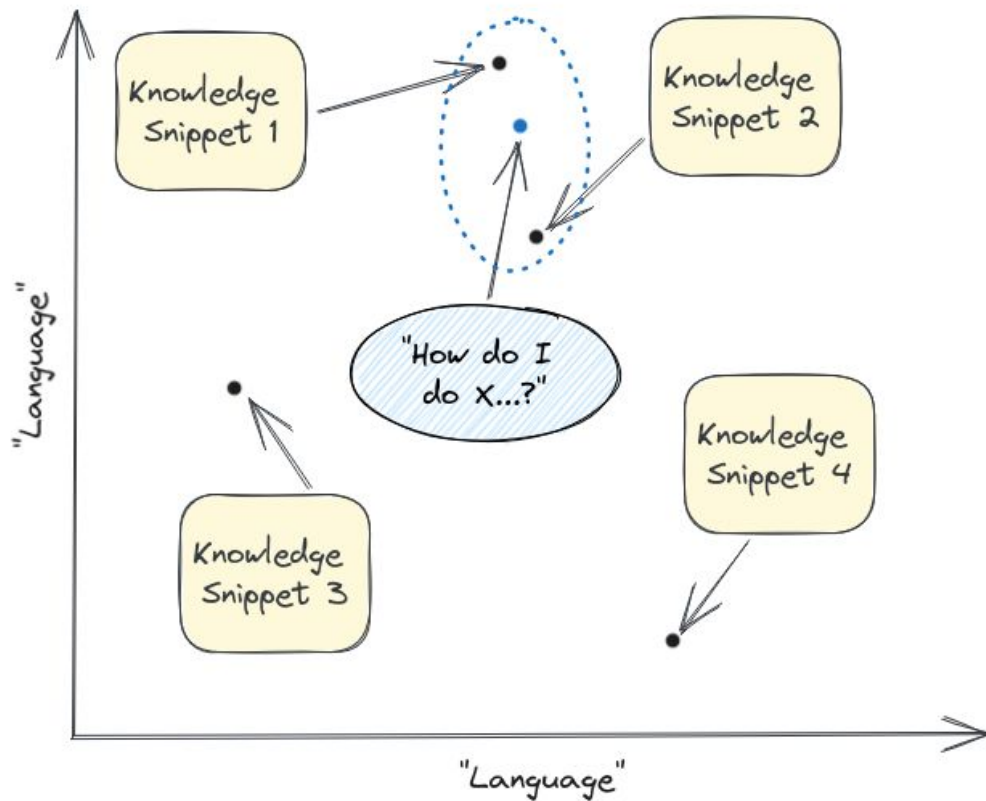


Retrieval with Embeddings (Querying)

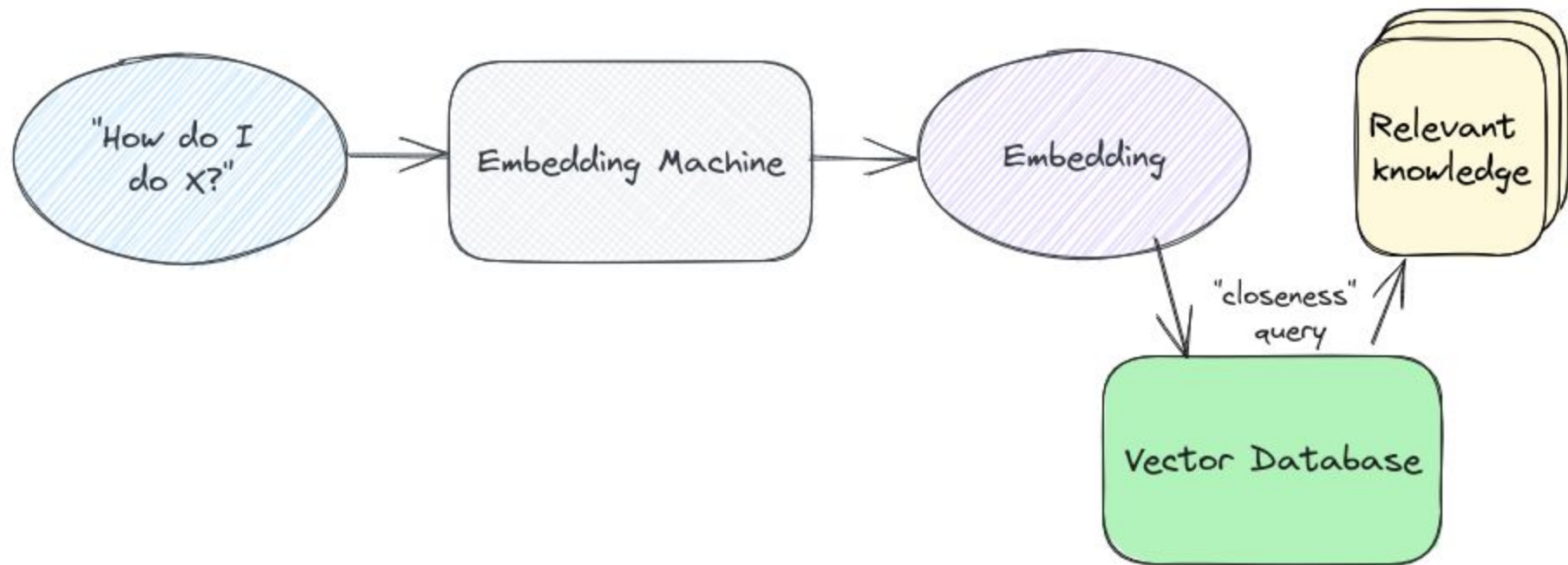
First get the embedding for our question.



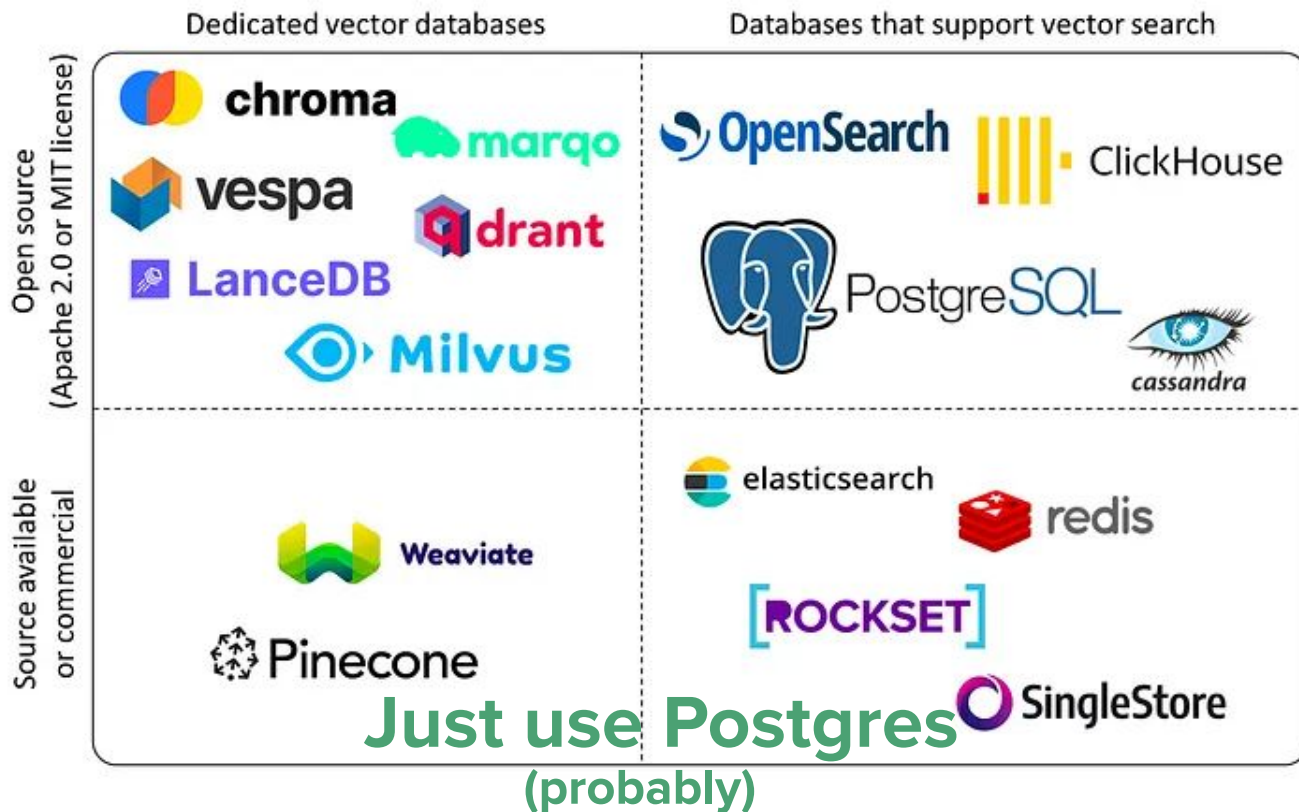
Then plot it.



Retrieval with Embeddings (Querying)

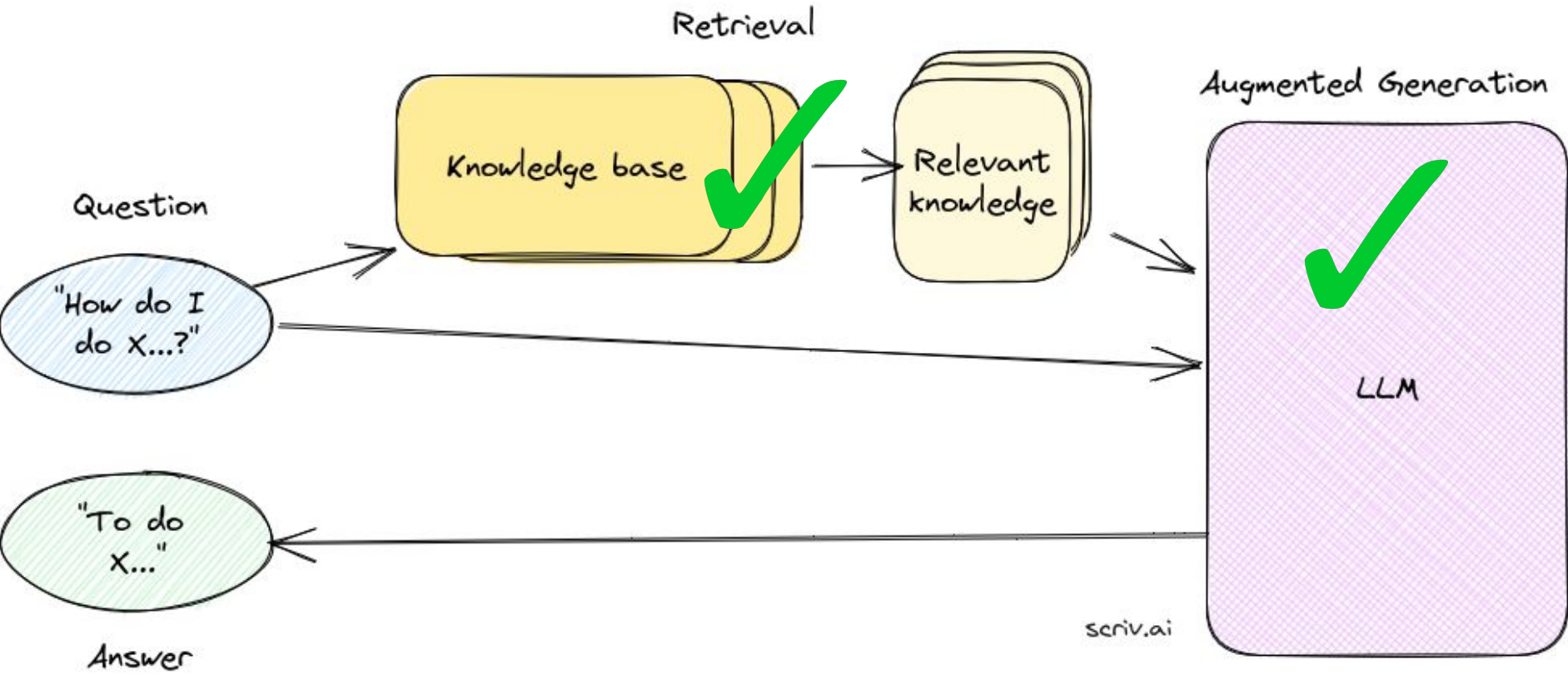


Vector Databases



Source: <https://blog.det.life/why-you-shouldnt-invest-in-vector-databases-c0cd3f59d23c>

Back to the Overview



Back to LangChain

But what about these ones?

```
1 from langchain.document_loaders import WebBaseLoader
2 from langchain.indexes import VectorstoreIndexCreator
3 loader = WebBaseLoader("http://www.paulgraham.com/greatwork.html")
4 index = VectorstoreIndexCreator().from_loaders([loader])
5 index.query("What should I work on?")
```

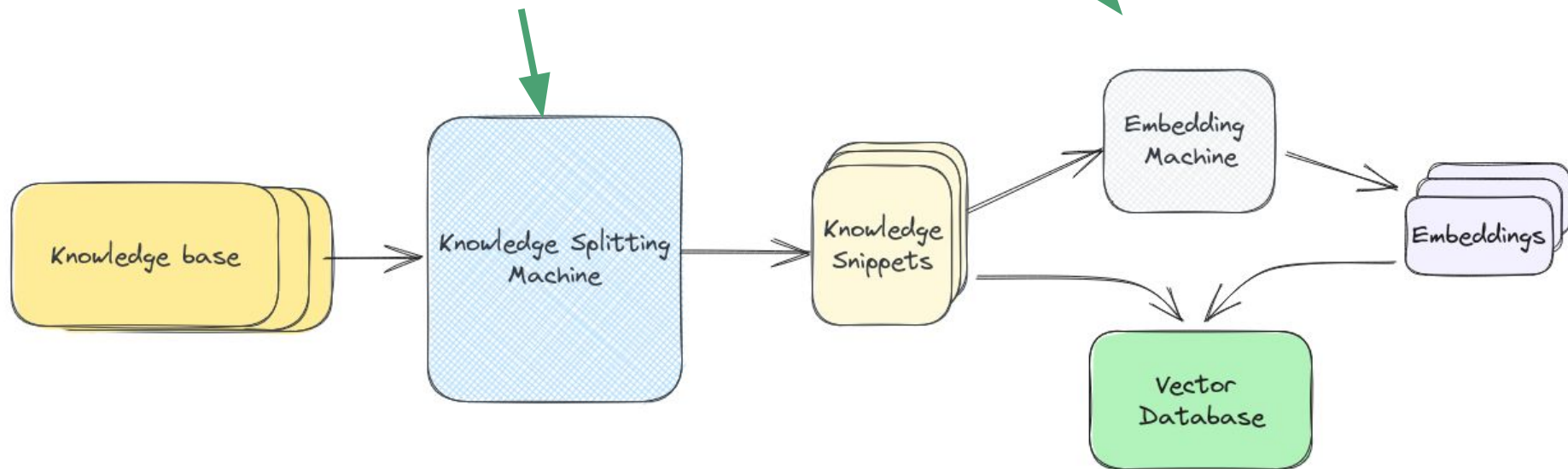
We've now covered this line

Building the knowledge base

Back to Indexing

But what's happening here?

We covered this



Building the knowledge base is the most important part of the whole thing.

It's probably where **95%** of the work happens.

Two conceptual pieces

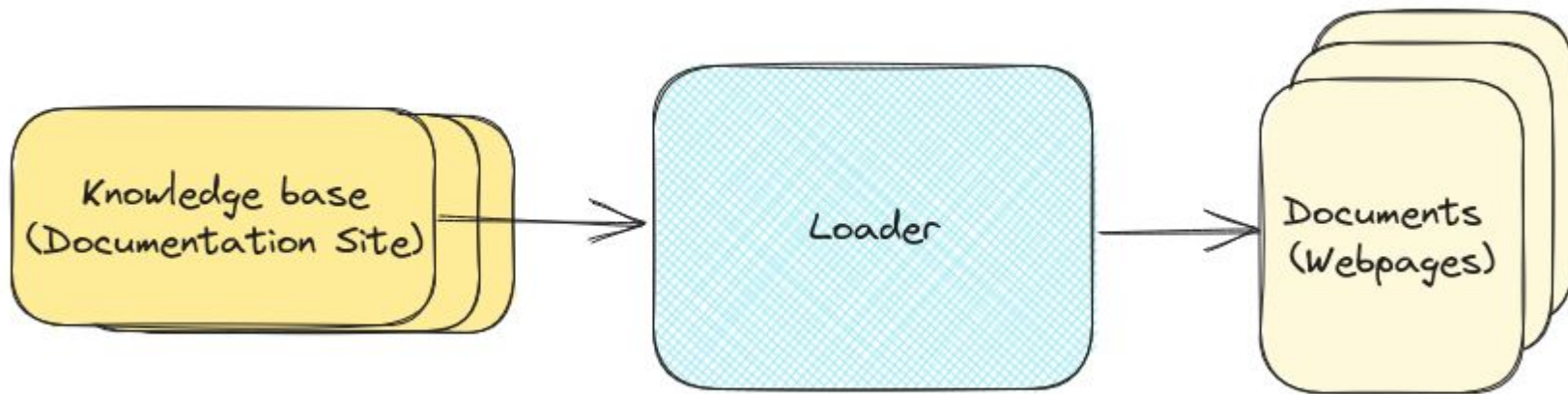
Loading: Getting the contents of your knowledge base out of wherever it is normally stored.

Splitting: Splitting up the knowledge into snippet-sized chunks that work well with embedding searches.

In LangChain these steps are handled by tools called *loaders* and *splitters*.

Loading is plumbing

Examples: scraping websites, extracting text from documents, consuming APIs, etc.



LangChain loaders

Mastodon

Mastodon is a federated social media and social networking service.

MediaWiki Dump

MediaWiki XML Dumps contain the content of a wiki (wiki pages wi...

Trello

Trello is a web-based project management and collaboration tool t...

TSV

A tab-separated values (TSV) file is a simple, text-based file format ...

Merge Documents Loader

Merge the documents returned from a set of specified data loaders.

mhtml

MHTML is a is used both for emails but also for archived webpages....

Twitter

Twitter is an online social media and social networking service.

Unstructured File

This notebook covers how to use Unstructured package to load file...

Microsoft OneDrive

Microsoft OneDrive (formerly SkyDrive) is a file hosting service op...

Microsoft PowerPoint

Microsoft PowerPoint is a presentation program by Microsoft.

URL

This covers how to load HTML documents from a list of URLs into a ...

Weather

OpenWeatherMap is an open source weather service provider

Microsoft SharePoint

Microsoft SharePoint is a website-based collaboration system that ...

Microsoft Word

Microsoft Word is a word processor developed by Microsoft.

WebBaseLoader

This covers how to use WebBaseLoader to load all text from HTML ...

WhatsApp Chat

WhatsApp (also called WhatsApp Messenger) is a freeware, cross-p...

Modern Treasury

Modern Treasury simplifies complex payment operations. It is a uni...

MongoDB

MongoDB is a NoSQL, document-oriented database that supports ...

Wikipedia

Wikipedia is a multilingual free online encyclopedia written and ma...

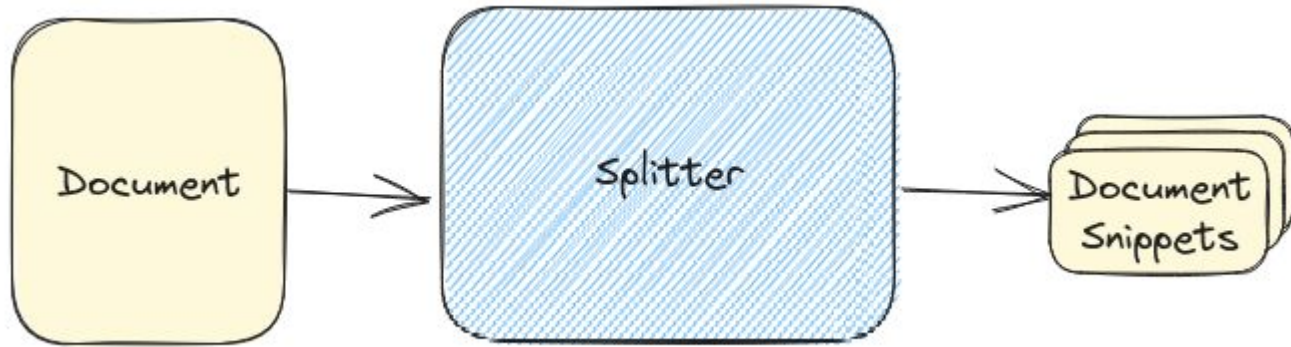
XML

The UnstructuredXMLLoader is used to load XML files. The loader ...

https://python.langchain.com/docs/integrations/document_loaders/

Splitting is also plumbing, but it's harder to get right

Examples: break on pages, break on section headings, break after a certain number of words, etc.

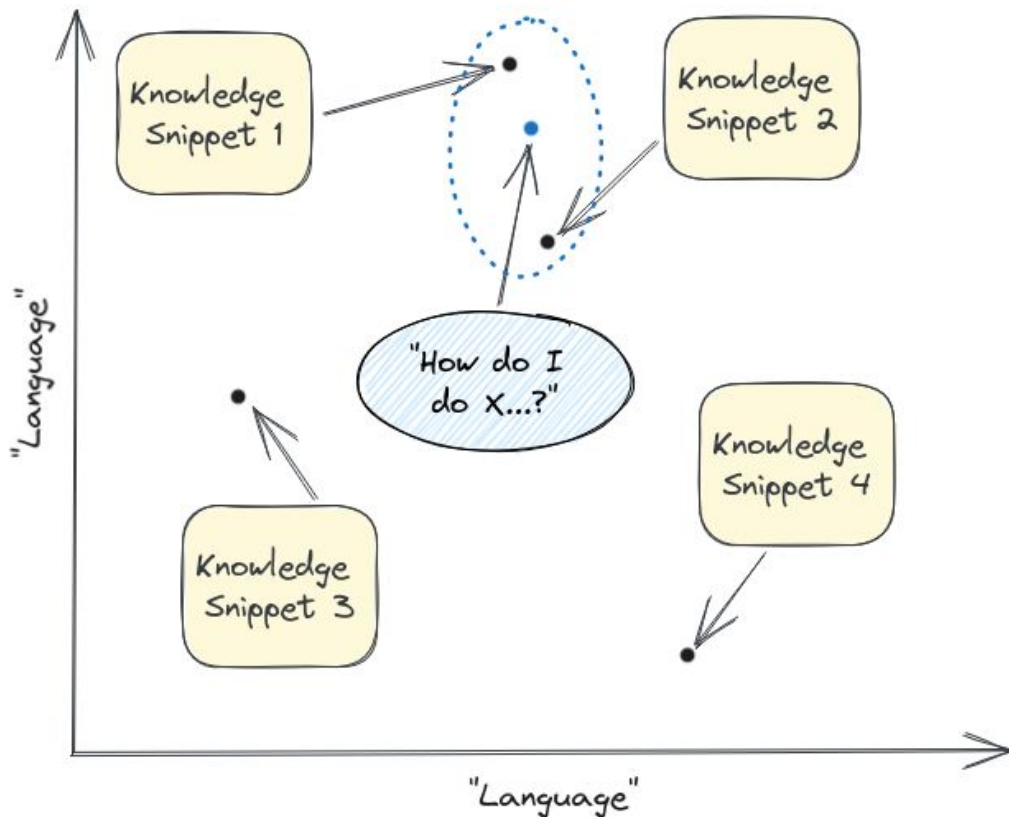


Why do we need to split our documents?

Recall how we are doing retrieval.

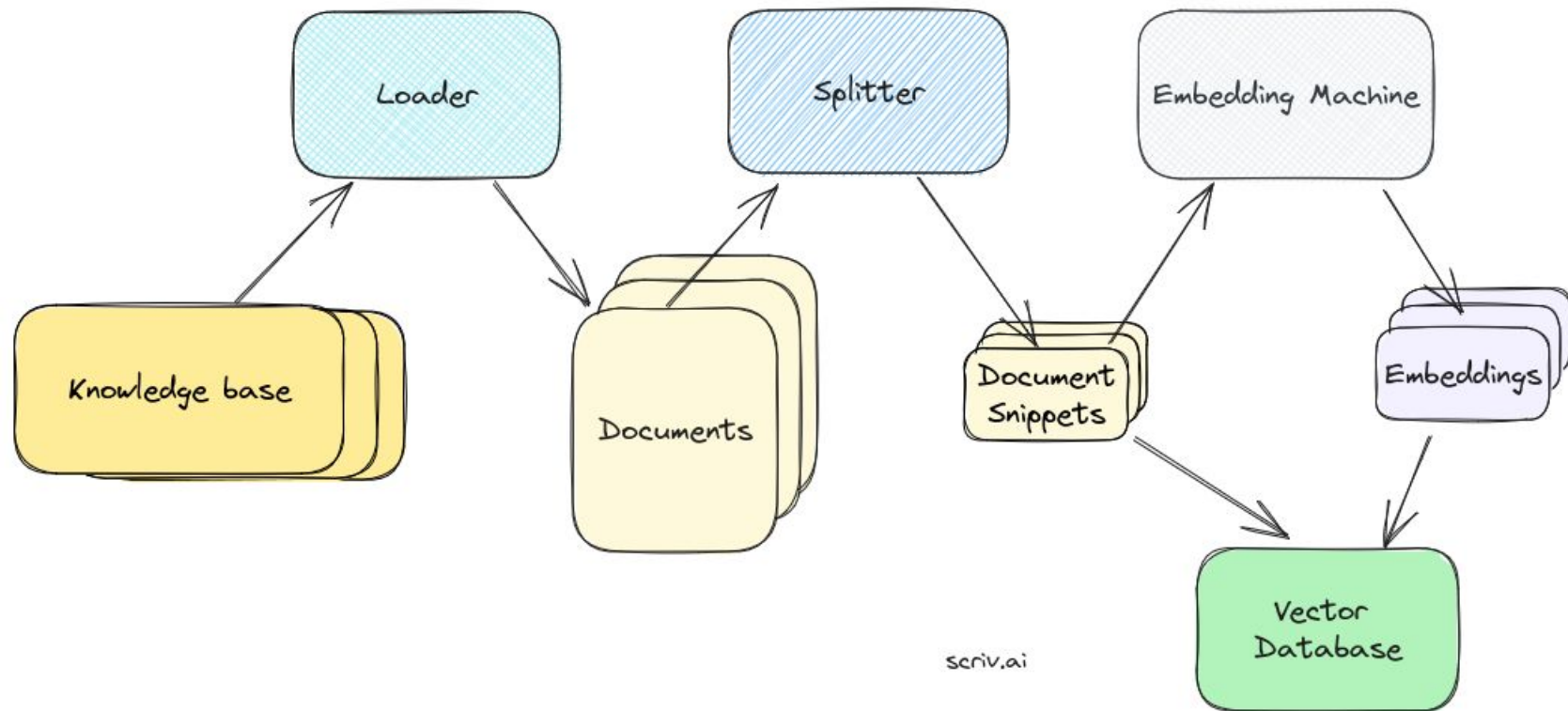
Too big and the location of the embedding gets less meaningful.

Too small and there's not enough information to answer the question.



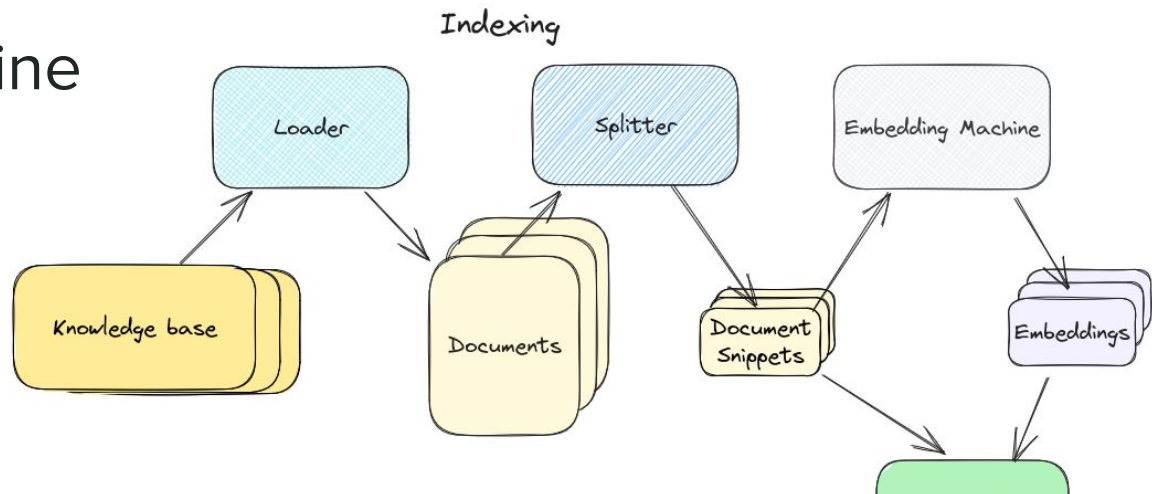
The whole indexing process

Knowledge Indexing

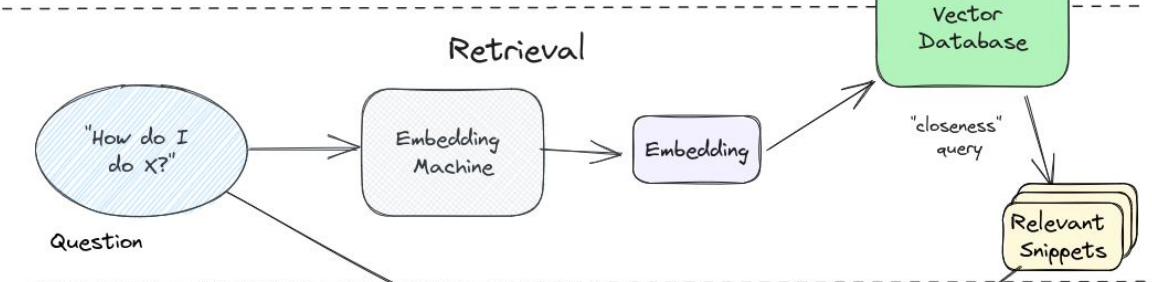


The whole RAG Pipeline

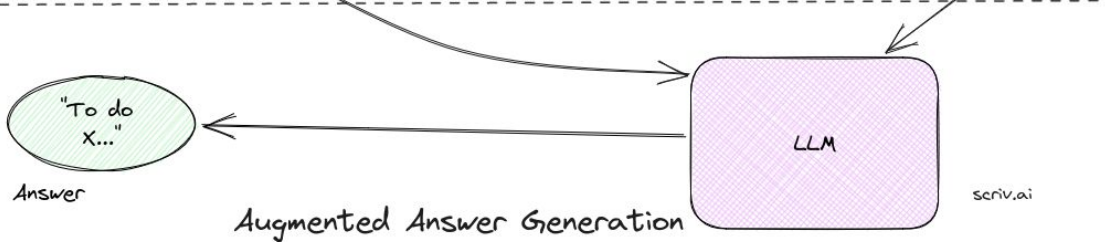
Indexing: We first load, split, embed, and save our knowledge base and embeddings in our vector DB.



Retrieval: A question is embedded and the closest matching knowledge snippets are pulled out.



Augmented Generation: The question and snippets are formatted and sent to the LLM to get our context-specific answer



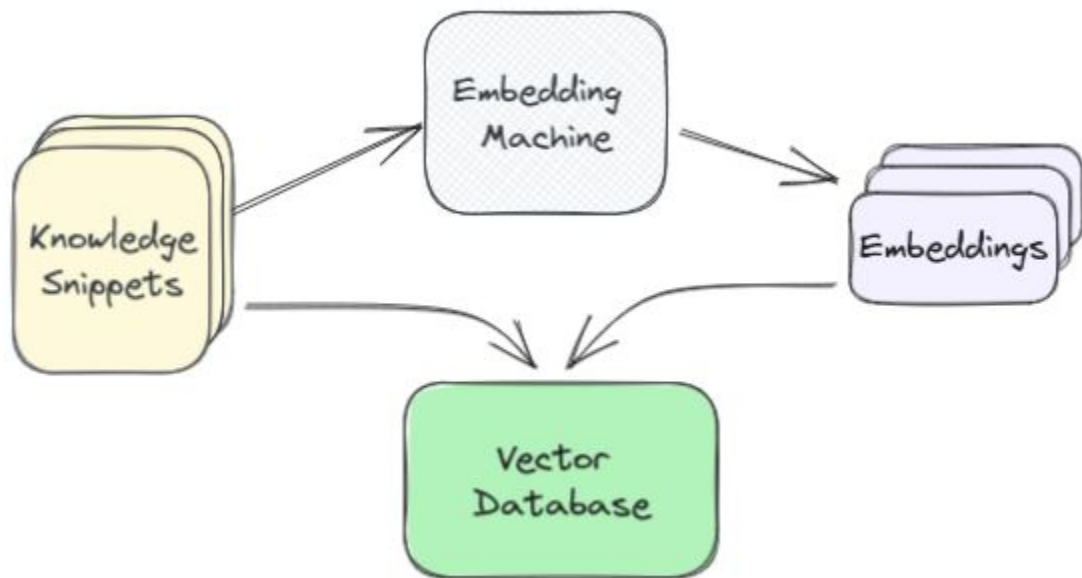
Congratulations, you now understand this code!

```
1 from langchain.document_loaders import WebBaseLoader
2 from langchain.indexes import VectorstoreIndexCreator
3 loader = WebBaseLoader("http://www.paulgraham.com/greatwork.html")
4 index = VectorstoreIndexCreator().from_loaders([loader])
5 index.query("What should I work on?")
```

Optimizations

More complex embeddings

There is no rule that says the embeddings have to exactly match the knowledge snippets, or even be 1:1 with them.



Adding metadata

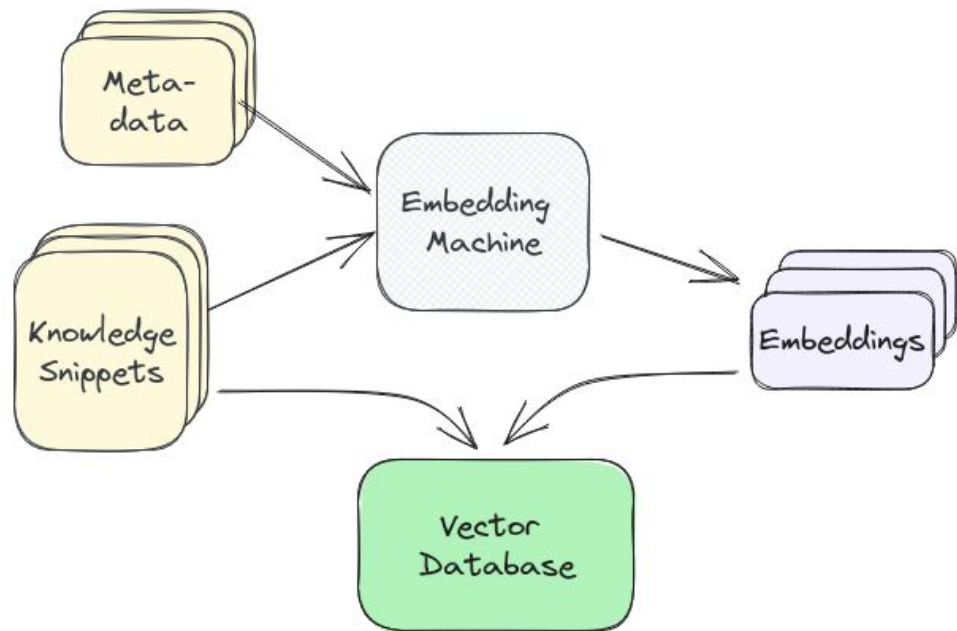
Idea: Instead of just embedding the snippet, also attach metadata.

Imagine a bot to help navigate visa paperwork.

You could add more context to every document snippet. E.g:

- Country: South Africa
- Visa Type: Critical Skills

Now the bot will do much better at answering questions like “do I need police clearance for a critical skills visa in South Africa?”



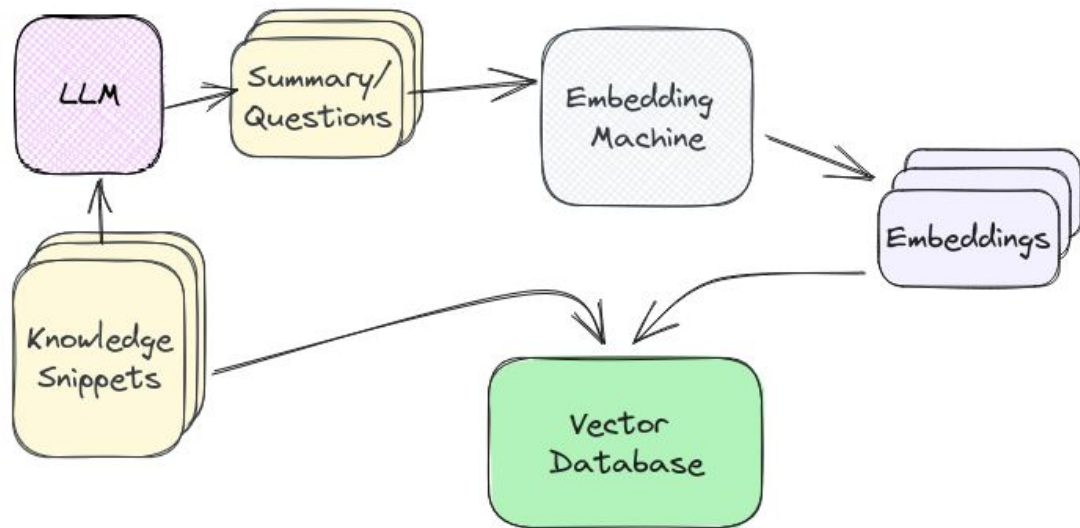
Adding other LLM-generated content

Idea: Instead of just embedding the snippet, use an LLM to generate alternate embeddings.

For example:

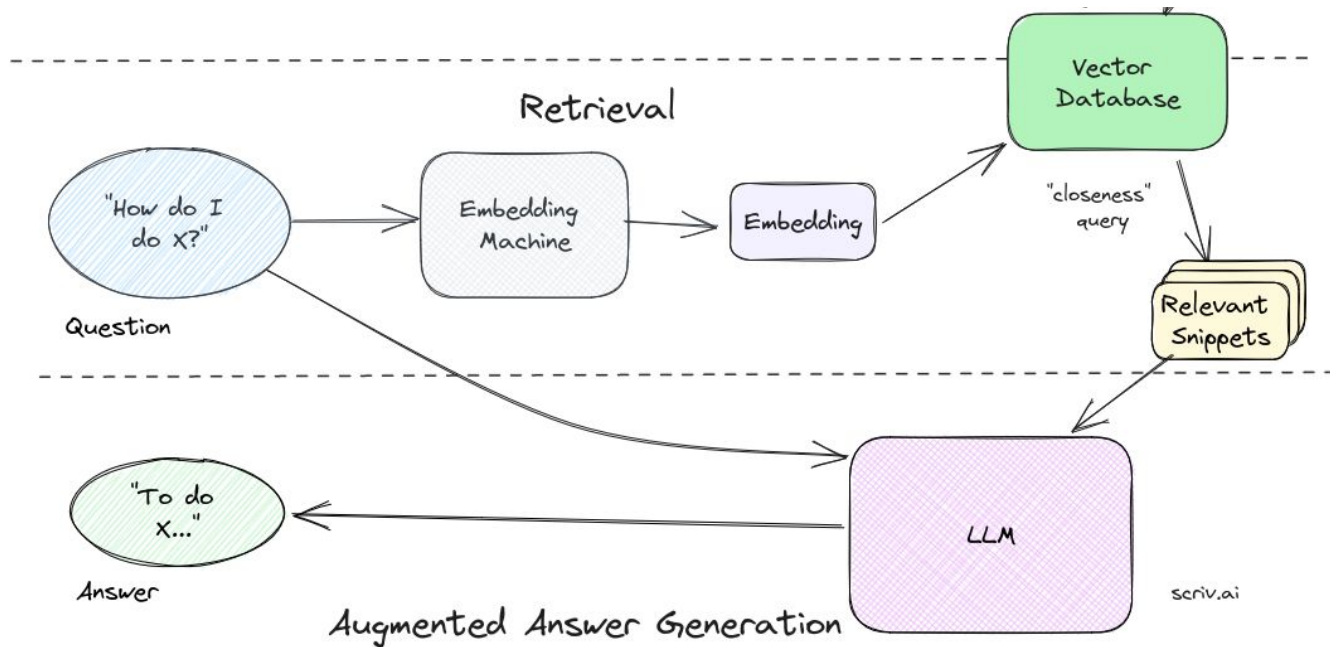
“Summarize this document.”

“List ten questions that this document answers.”



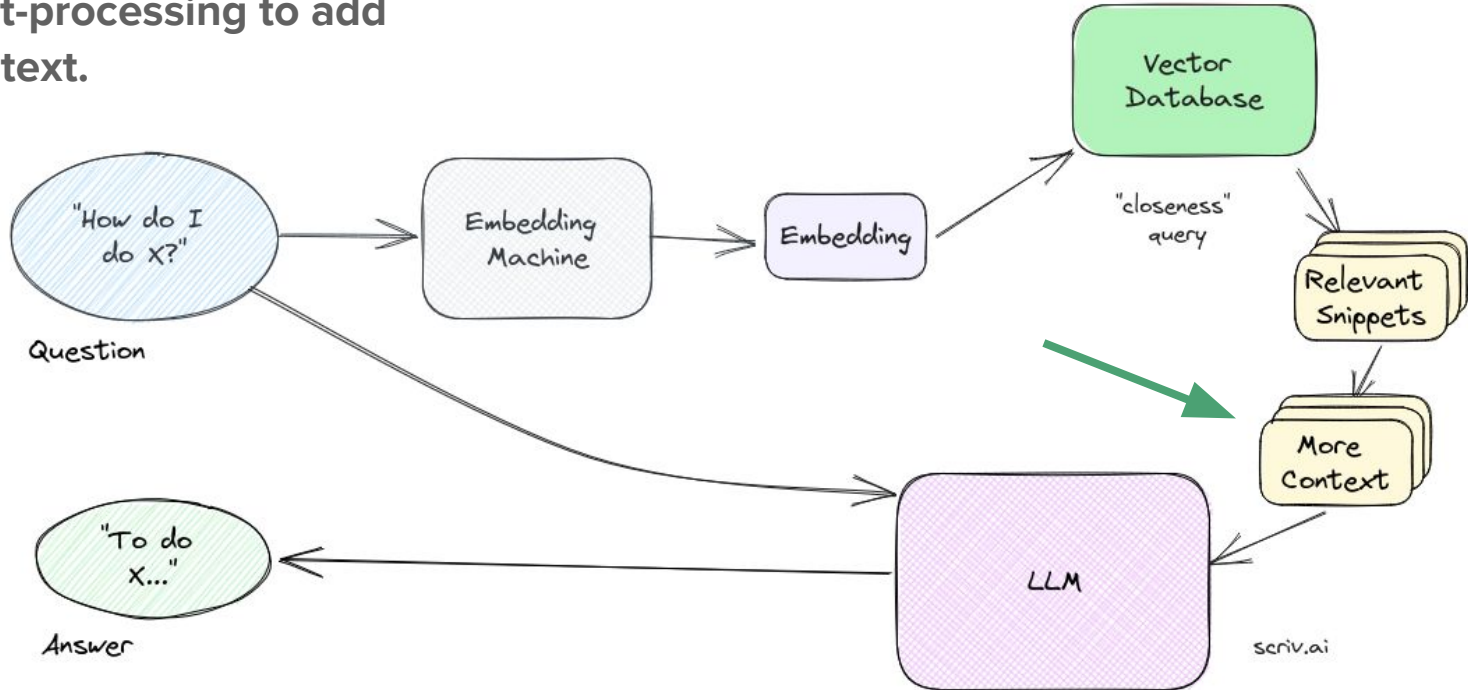
More complex retrievers

There is no rule that says the retrieved snippets are all that you send to the LLM.



Expanding the retrieved snippets

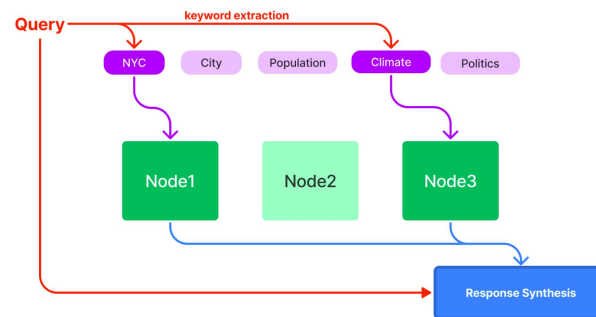
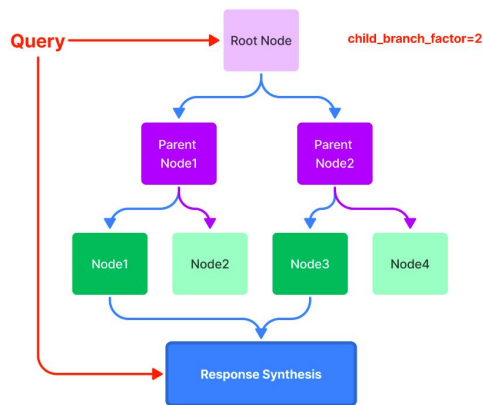
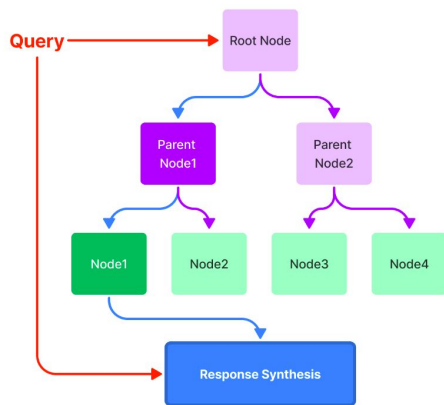
Idea: once you've retrieved knowledge snippets, do post-processing to add surrounding context.



Advanced index/querying

Llama Index has keyword extraction, trees-based querying, and more.

(don't ask me how this works)



Questions?

Text version of this talk: <https://scriv.ai/guides/retrieval-augmented-generation-overview/>