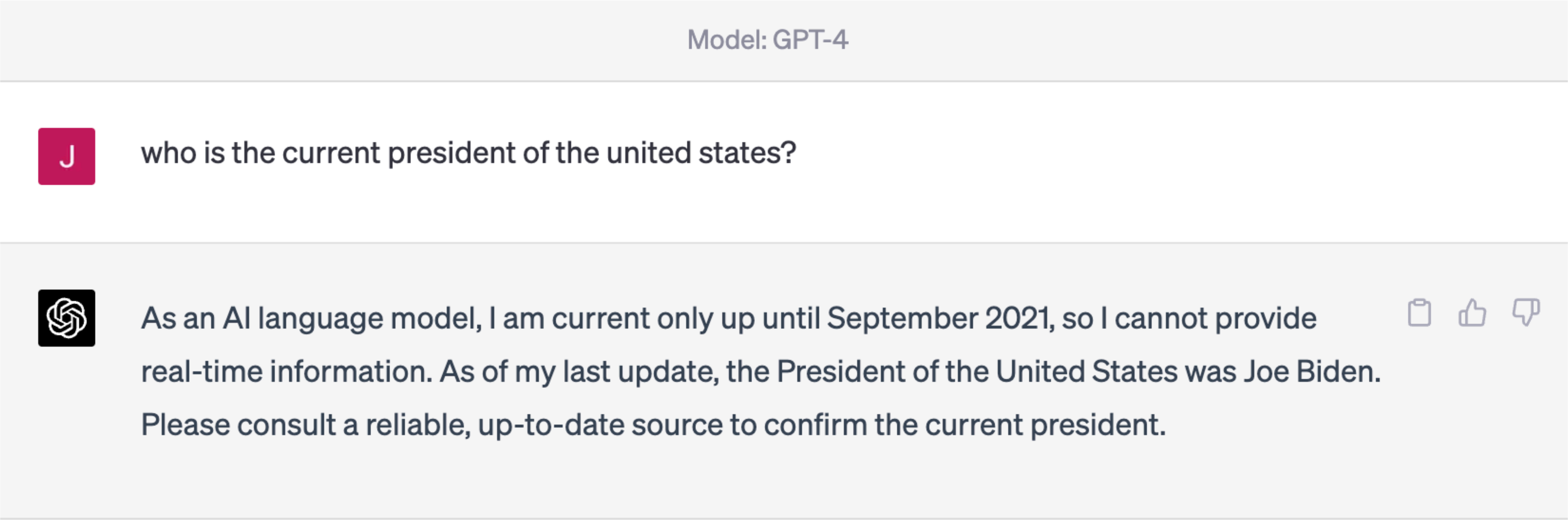
LLM Bootcamp 2023

Augmented language models

Josh Tobin

April 20, 2023

There’s a lot language models don’t know



Model: GPT-4

J

who is the current president of the united states?

As an AI language model, I am current only up until September 2021, so I cannot provide real-time information. As of my last update, the President of the United States was Joe Biden. Please consult a reliable, up-to-date source to confirm the current president.

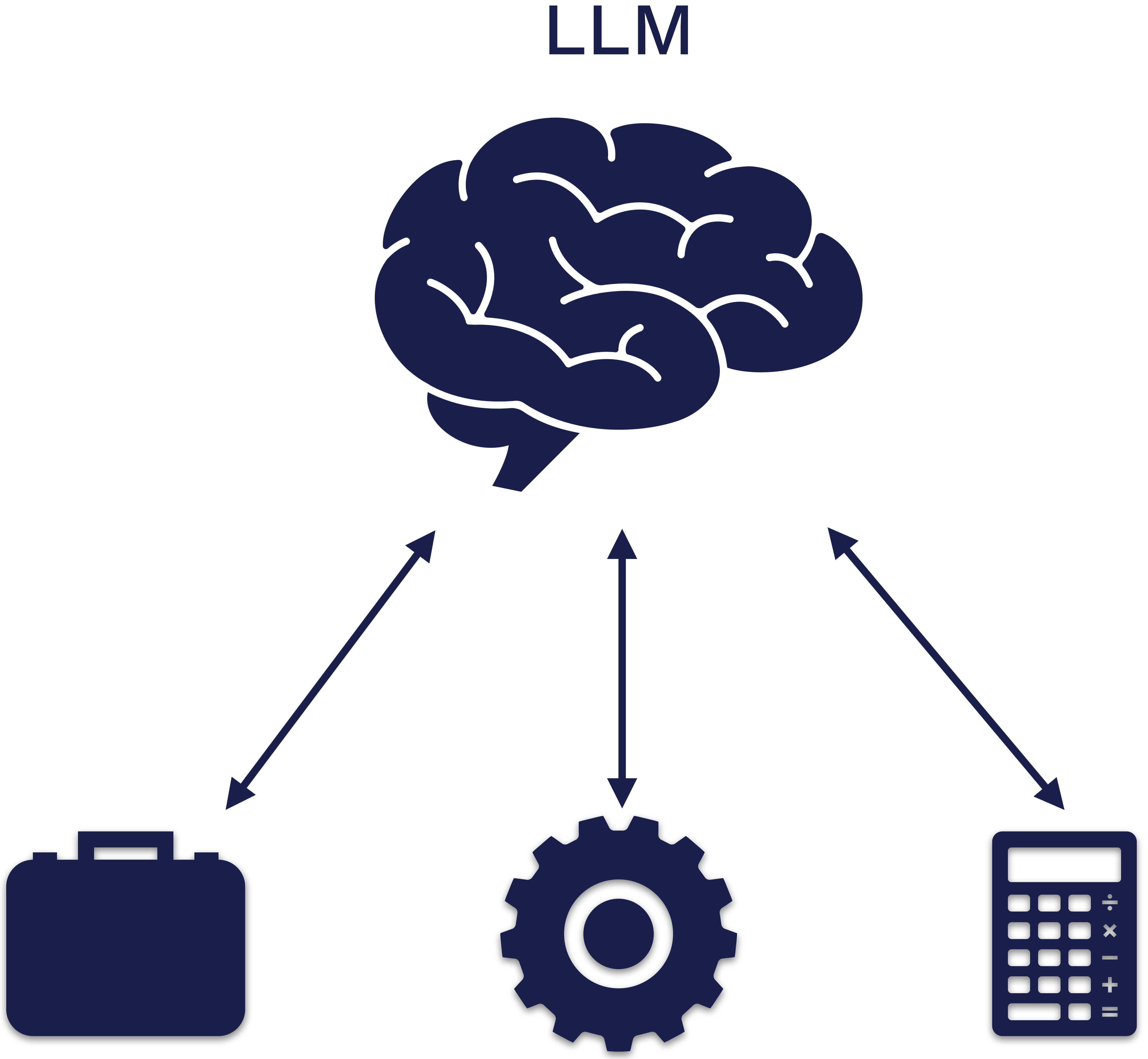
What (base) LLMs are good at

- Language understanding
- Instruction following
- Basic reasoning
- Code understanding

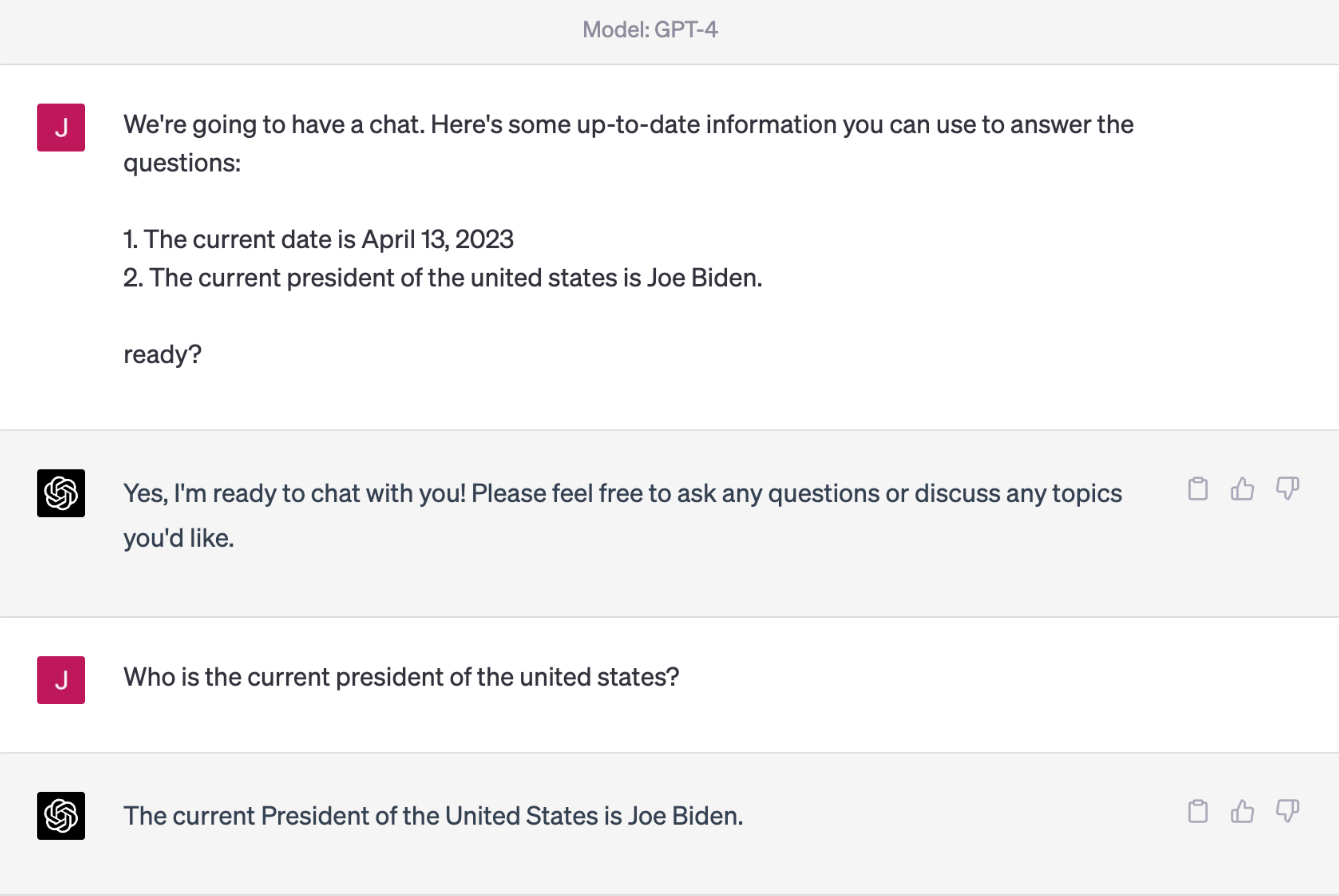
What they need help with

- Up-to-date knowledge
- Knowledge of your data
- More challenging reasoning
- Interacting with the world

LLMs are for general reasoning, not specific knowledge



A baseline: using the context window



Model: GPT-4

J

We're going to have a chat. Here's some up-to-date information you can use to answer the questions:

1. The current date is April 13, 2023
2. The current president of the united states is Joe Biden.

ready?

Yes, I'm ready to chat with you! Please feel free to ask any questions or discuss any topics you'd like.

J

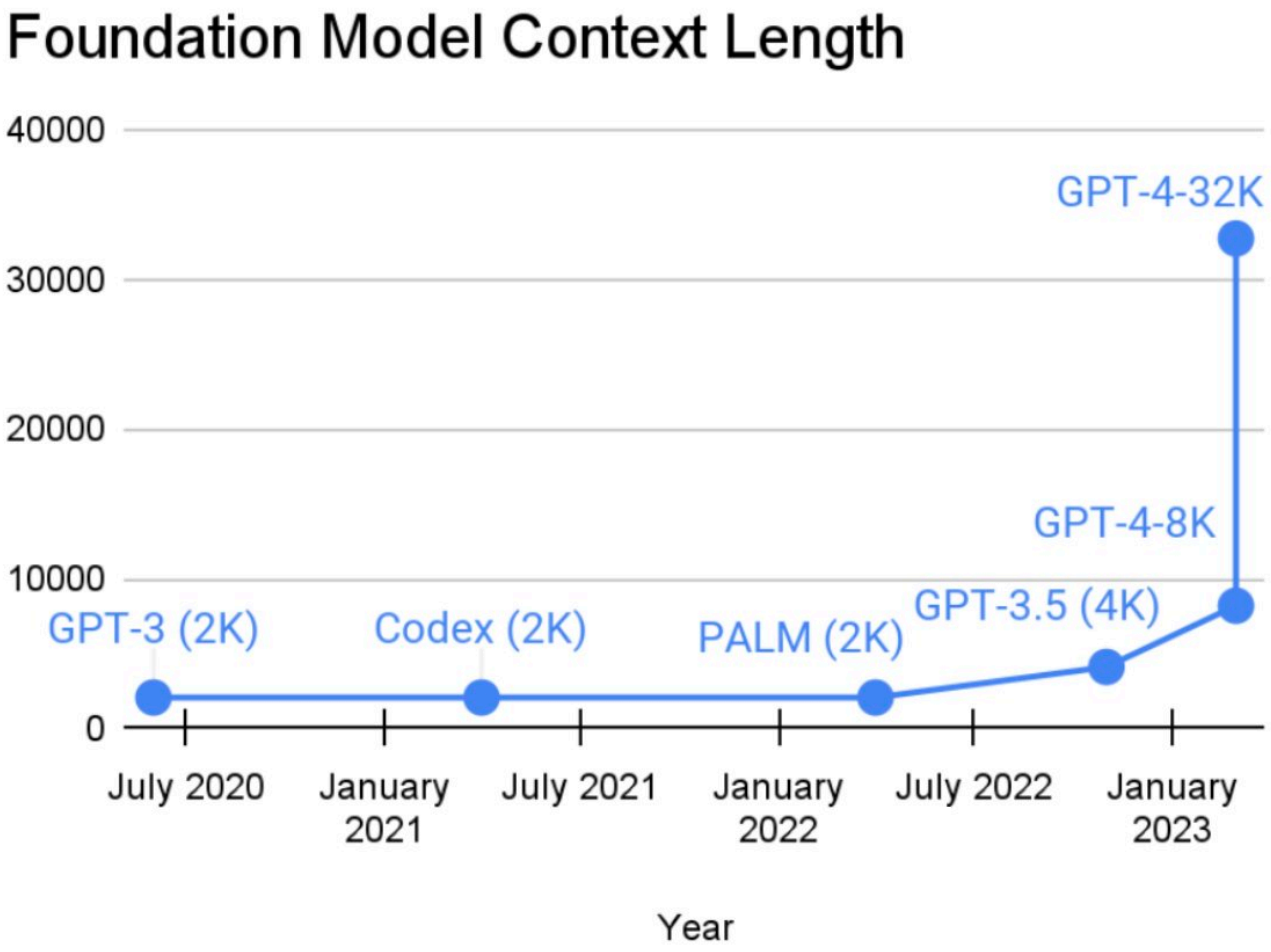
Who is the current president of the united states?

The current President of the United States is Joe Biden.

Context is the way to give LLM unique, up-to-date information…

…But it only fits a limited amount of information

How much information can you fit in the context window?



Source: Dan Fu (https://twitter.com/realDanFu/status/1640762299408601089?s=20)

How much information can you fit in the context window?

Number of tokens	50	500	4,000	32,000	256,000	2,048,000	8,192,000	65,536,000
Example model		GPT-1	GPT-3.5	GPT-4-32K				
How much is it?	A sentence	~4 paragraphs of writing	New Yorker article	College thesis	Novel	~7,500 emails (or, about 1 year's worth for a productive office worker)	30 seconds worth of tweets (At 40 tokens per tweet, ~400000 / minute)	~500Mb of unicode text data (a single ElasticSearch node can store 50gb)

Context windows are growing fast,
but won't fit everything for a while

(Plus, more context = more $$$)

This lecture: how to make the most of a limited context by augmenting the language model

Augmented language models

Retrieval

Augment with
a bigger corpus

Chains

Augment with
more LLM calls

Tools

Augment with
outside sources

01

Retrieval augmentation

Retrieval augmentation: outline

A. Why retrieval augmentation?

B. Traditional information retrieval

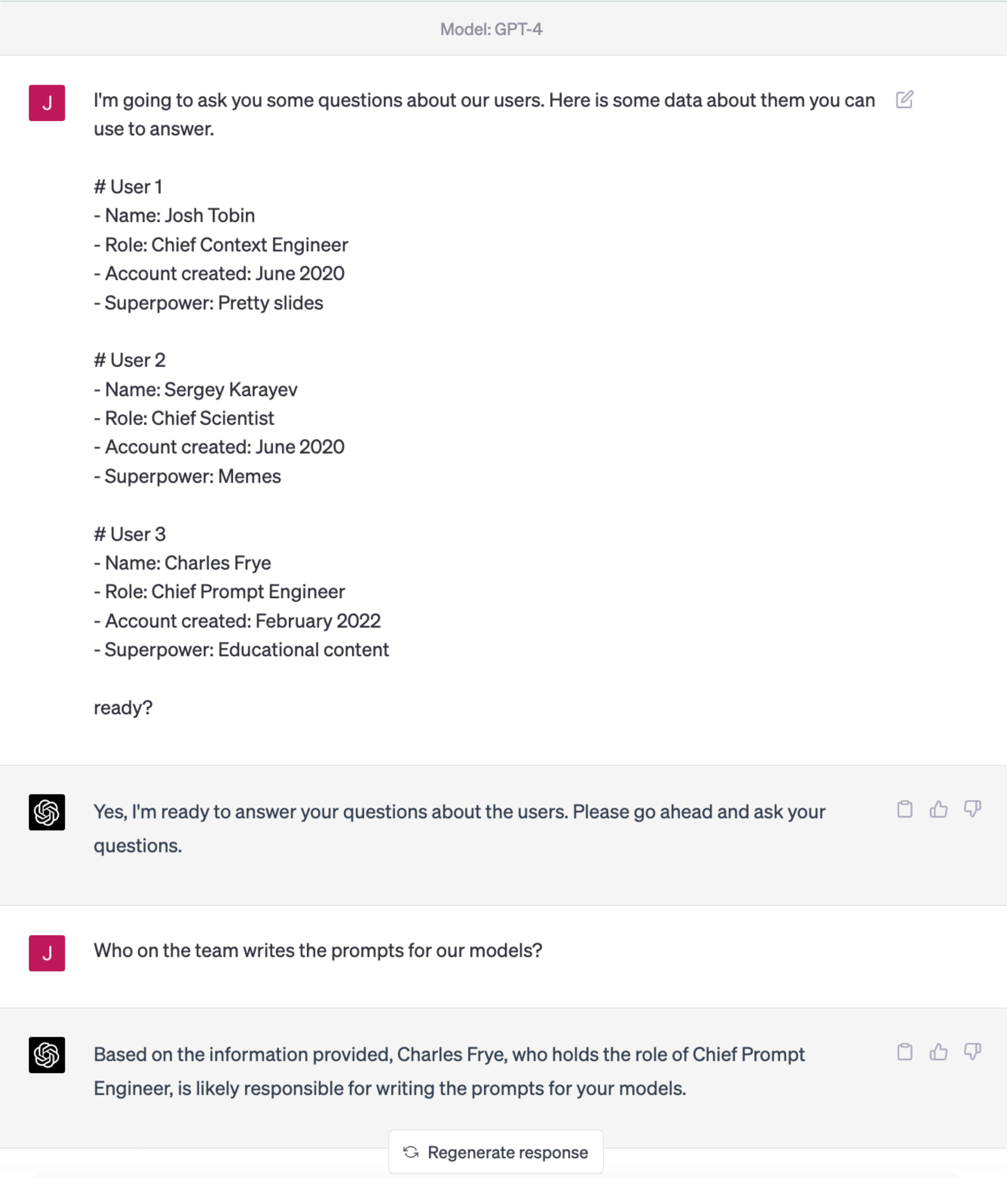
C. Information retrieval via embeddings

D. Patterns and case studies

A. Why retrieval augmentation?

Say we want our model to have access to user data

Approach 1: put it in the context



I'm going to ask you some questions about our users. Here is some data about them you can use to answer.

User 1
- Name: Josh Tobin
- Role: Chief Context Engineer
- Account created: June 2020
- Superpower: Pretty slides

User 2
- Name: Sergey Karayev
- Role: Chief Scientist
- Account created: June 2020
- Superpower: Memes

User 3
- Name: Charles Frye
- Role: Chief Prompt Engineer
- Account created: February 2022
- Superpower: Educational content

ready?

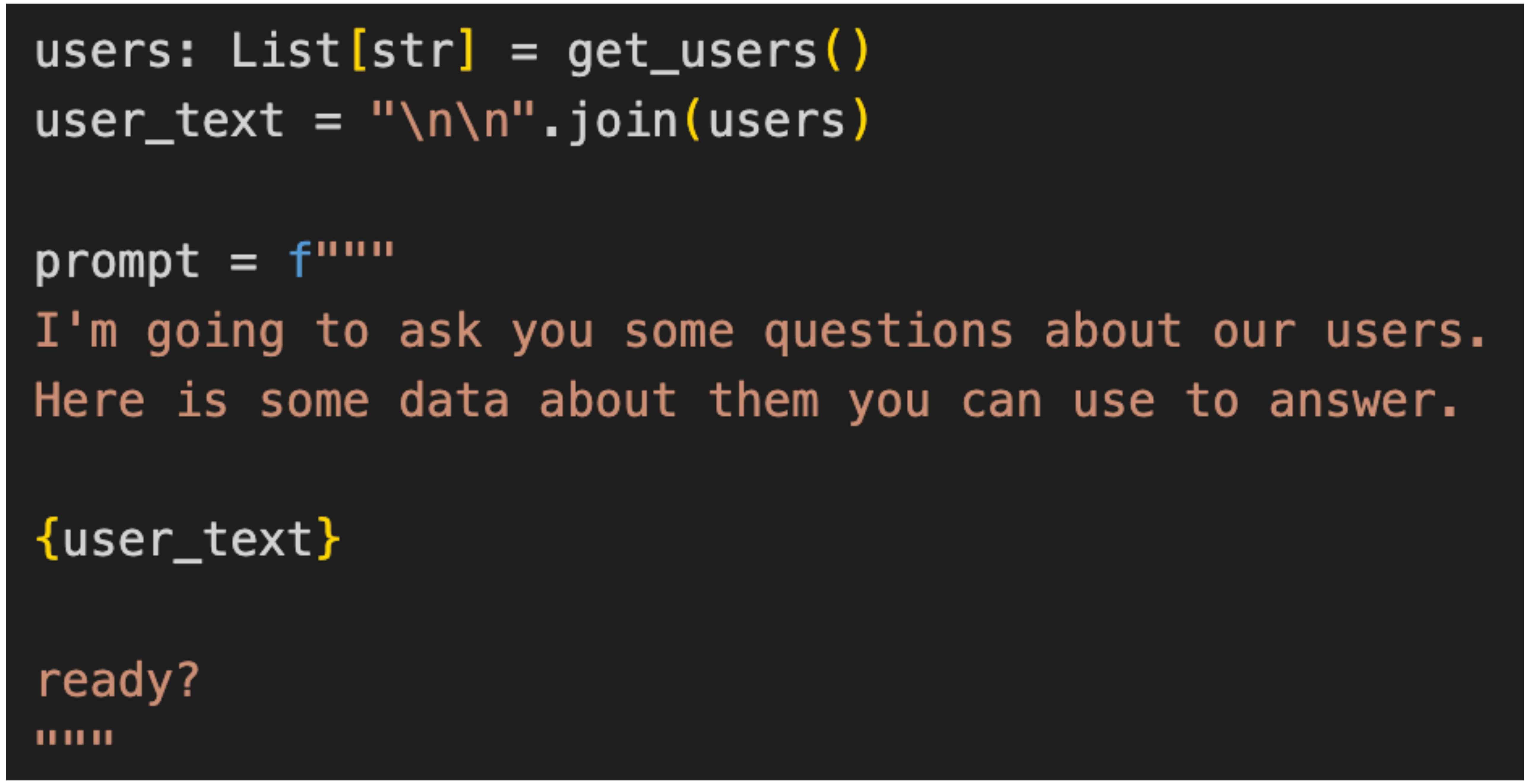
Yes, I'm ready to answer your questions about the users. Please go ahead and ask your questions.

Who on the team writes the prompts for our models?

Based on the information provided, Charles Frye, who holds the role of Chief Prompt Engineer, is likely responsible for writing the prompts for your models.

What if we have thousands of users?

- Use rules to figure out which users should go in the context
  - Most recent users?
  - Users mentioned in the query?
  - Most viewed users?
- What happens if the relationship is hard to write rules for?



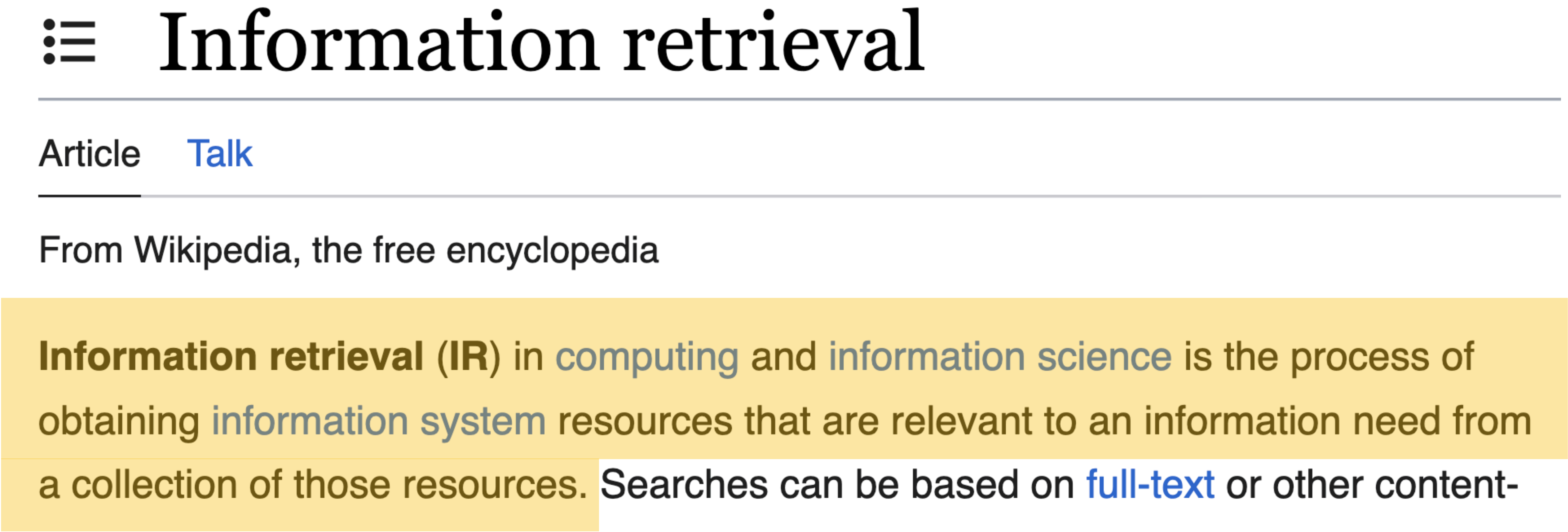
```
users: List[str] = get_users()
user_text = "\n\n".join(users)

prompt = f"""
I'm going to ask you some questions about our users.
Here is some data about them you can use to answer.

{user_text}

ready?
"""
```

Context-building is information retrieval



Information retrieval

From Wikipedia, the free encyclopedia

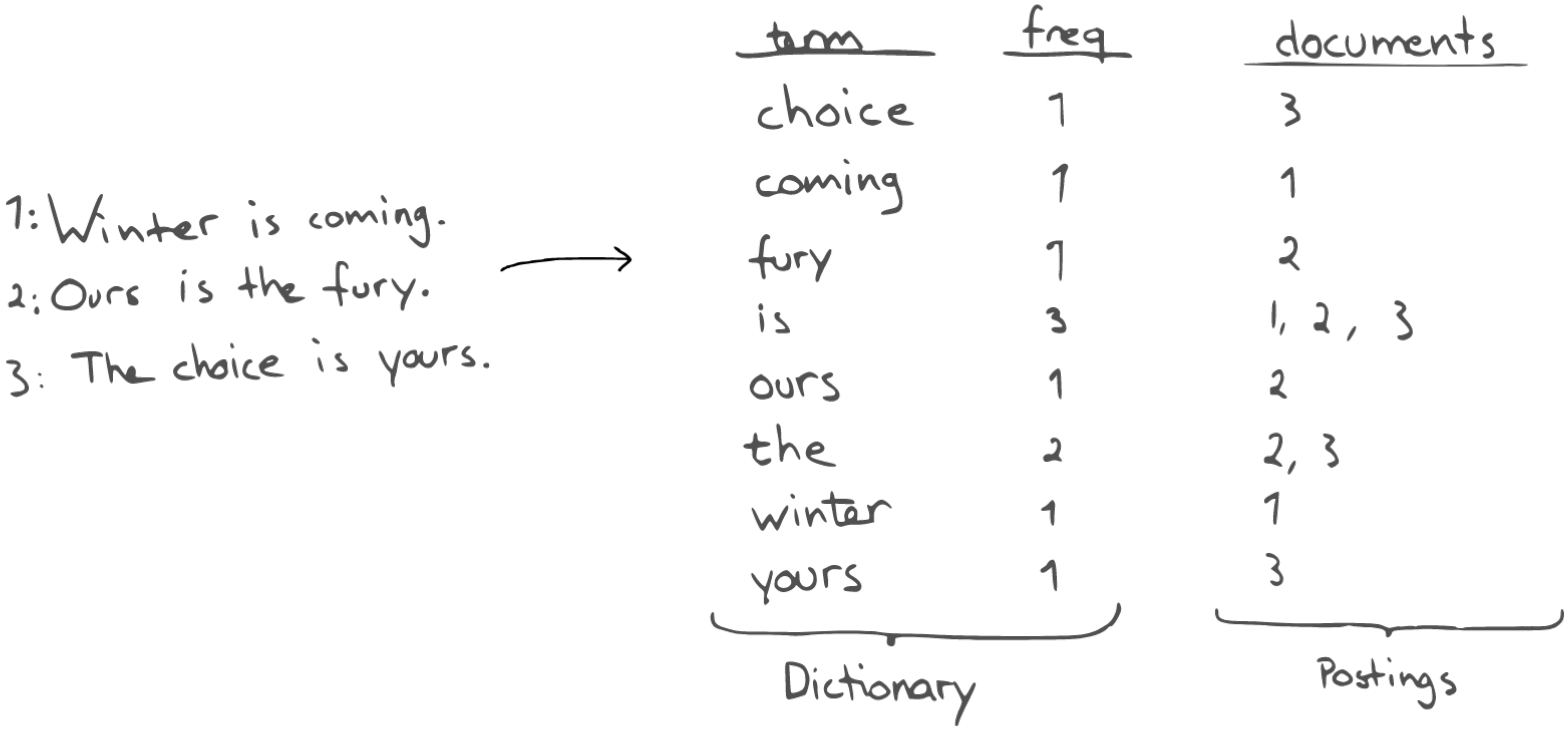
Information retrieval (**IR**) in computing and information science is the process of obtaining information system resources that are relevant to an information need from a collection of those resources. Searches can be based on full-text or other content-

B. Traditional information retrieval

Information retrieval basics

- **Query**. Formal statement of your information need. E.g., a search string.
- **Object.** Entity inside your content collection. E.g., a document.
- **Relevance.** Measure of how well an object satisfies the information need
- **Ranking.** Ordering of relevant results based on desirability

Traditional information retrieval: search via inverted indexes



1: Winter is coming.
2: Ours is the fury.
3: The choice is yours.

term	freq	documents
choice	1	3
coming	1	1
fury	1	2
is	3	1, 2, 3
ours	1	2
the	2	2, 3
winter	1	1
yours	1	3

https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up

Ranking & relevance in traditional search

- **Relevance** via boolean search
  - E.g., only return the docs that contain: *simple AND rest AND apis AND distributed AND nature*
- **Ranking** via BM25. Affected by 3 factors
  - **Term frequency (TF)** — More appearances of search term = more relevant object
  - **Inverse document frequency (IDF)** — More objects containing search term = less important search term
  - **Field length** — If a document contains a search term in a field that is very short (i.e. has few words), it is more likely relevant than a document that contains a search term in a field that is very long (i.e. has many words).

https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up

Search engines are more than inverted indices

- Document ingestion
- Document processing (e.g., remove stop words, lower case, etc)
- Transaction handling (adding / deleting documents, merging index files)
- Scaling via shards
- Ranking & relevance
- Etc

https://www.elastic.co/blog/found-elasticsearch-from-the-bottom-up

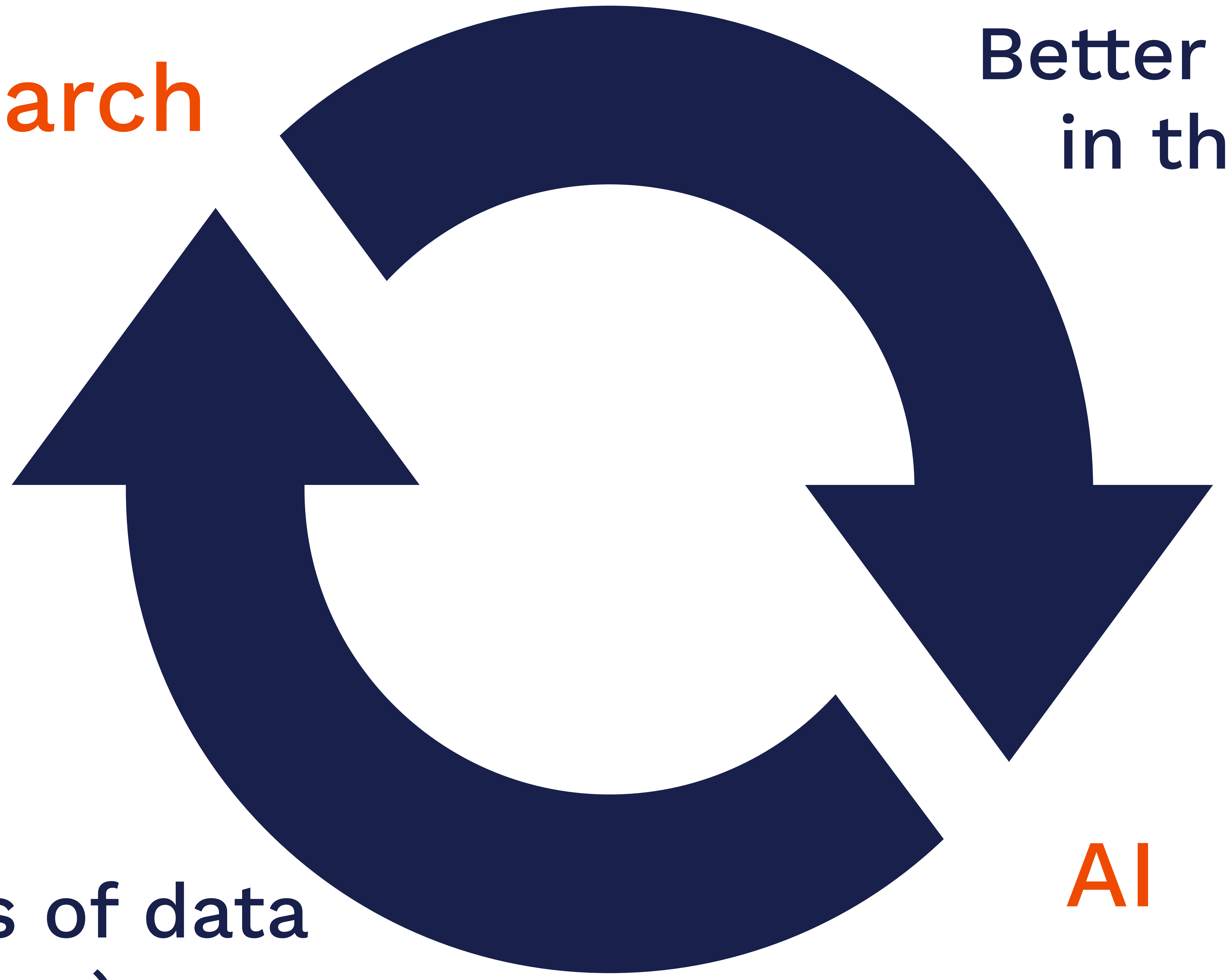
Limitations of “sparse” traditional search

- Only models simple word frequencies
- Doesn’t capture semantic information, correlation information, etc
- E.g., searching for “what is the top hand in bridge” might return documents about 🌉, ♠️, 💸

https://www.microsoft.com/en-us/research/video/research-talk-system-frontiers-for-dense-retrieval/

C. AI-powered information retrieval via embeddings

Search and AI make each other better



Search

Better information
in the context

AI

Better
representations of data
(embeddings)

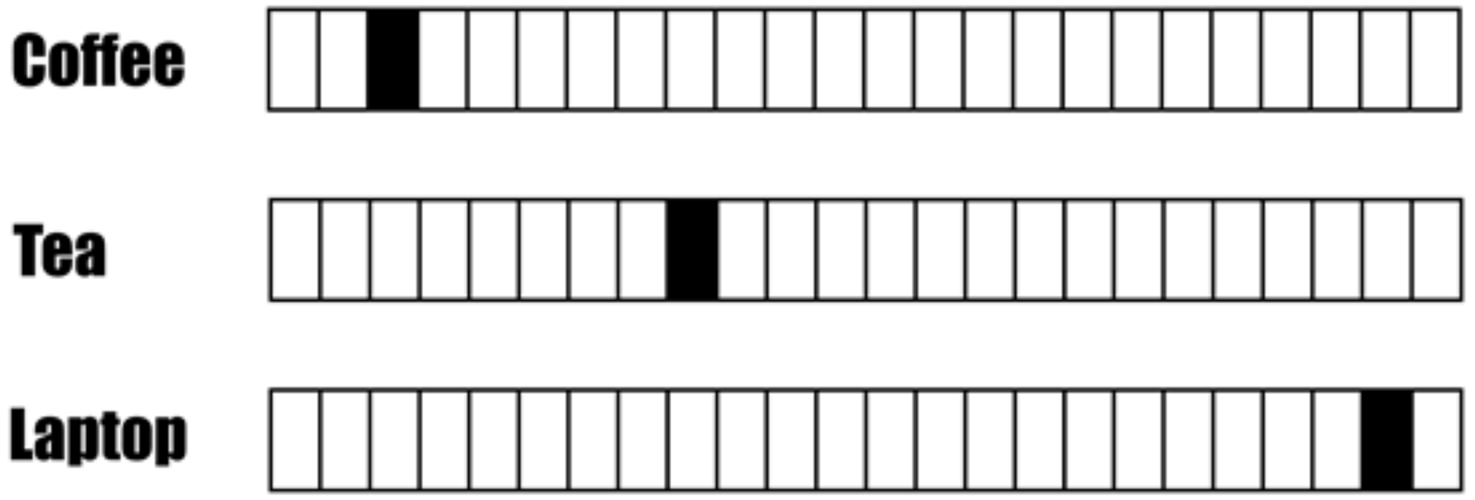
AI-powered retrieval via embeddings

- **All about embeddings**
- Embedding relevance and indexes
- Embedding databases
- Beyond naive nearest neighbor

Embeddings are an *abstract*, *dense*, *compact*, *fixed-size*, (usually) *learned* representation of data

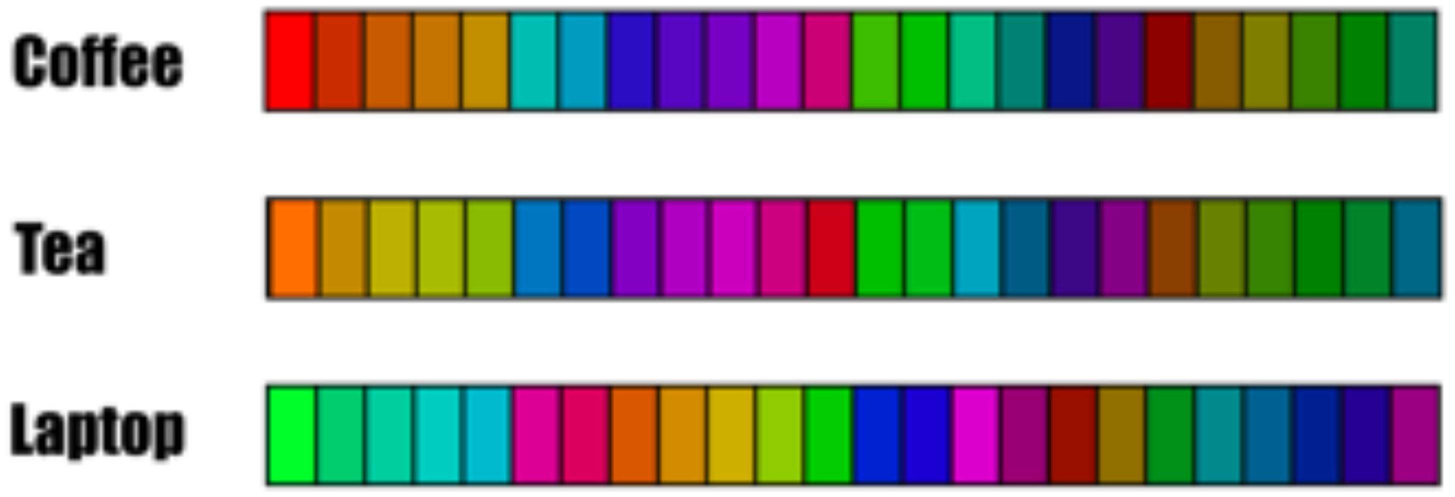
“Sparse” representation

Contains “coffee”
Contains “tea”
…
Contains “laptop”



“Dense” representation

0.1231
0.7412
…
0.6221

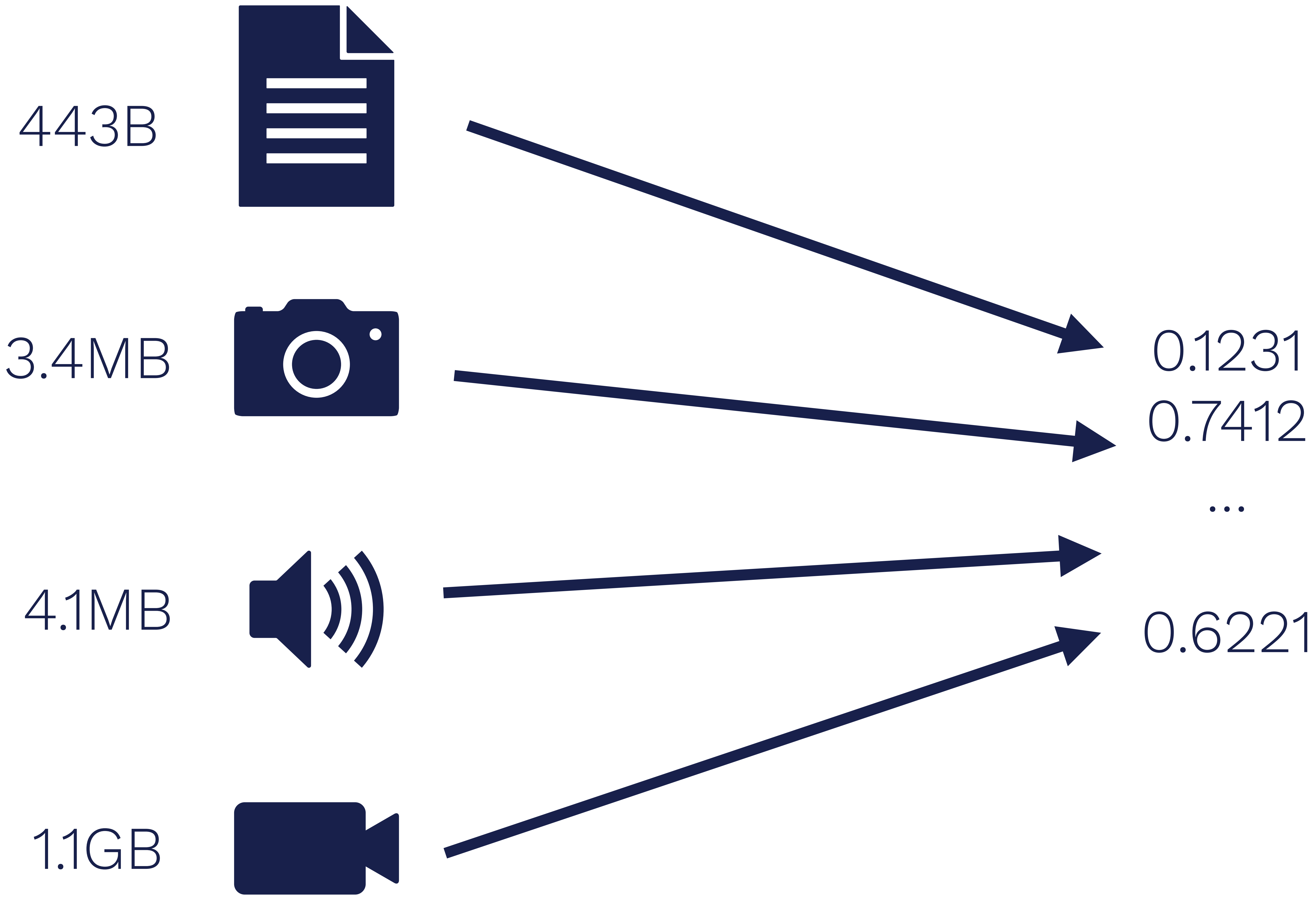


What are embeddings *not*?

- Embeddings are not necessarily one of the last learned layers of a network
- A single embedding doesn't have to refer to one single type of input
- Embeddings don't have to be from a neural network at all
- Embeddings don't have to be directly comparable in vector space
- …

What is an embedding, anyways? (https://simplicityissota.substack.com/p/what-is-an-embedding-anyways)

Why embeddings? Vectors are a compact, universal representation of data



443B

3.4MB

4.1MB

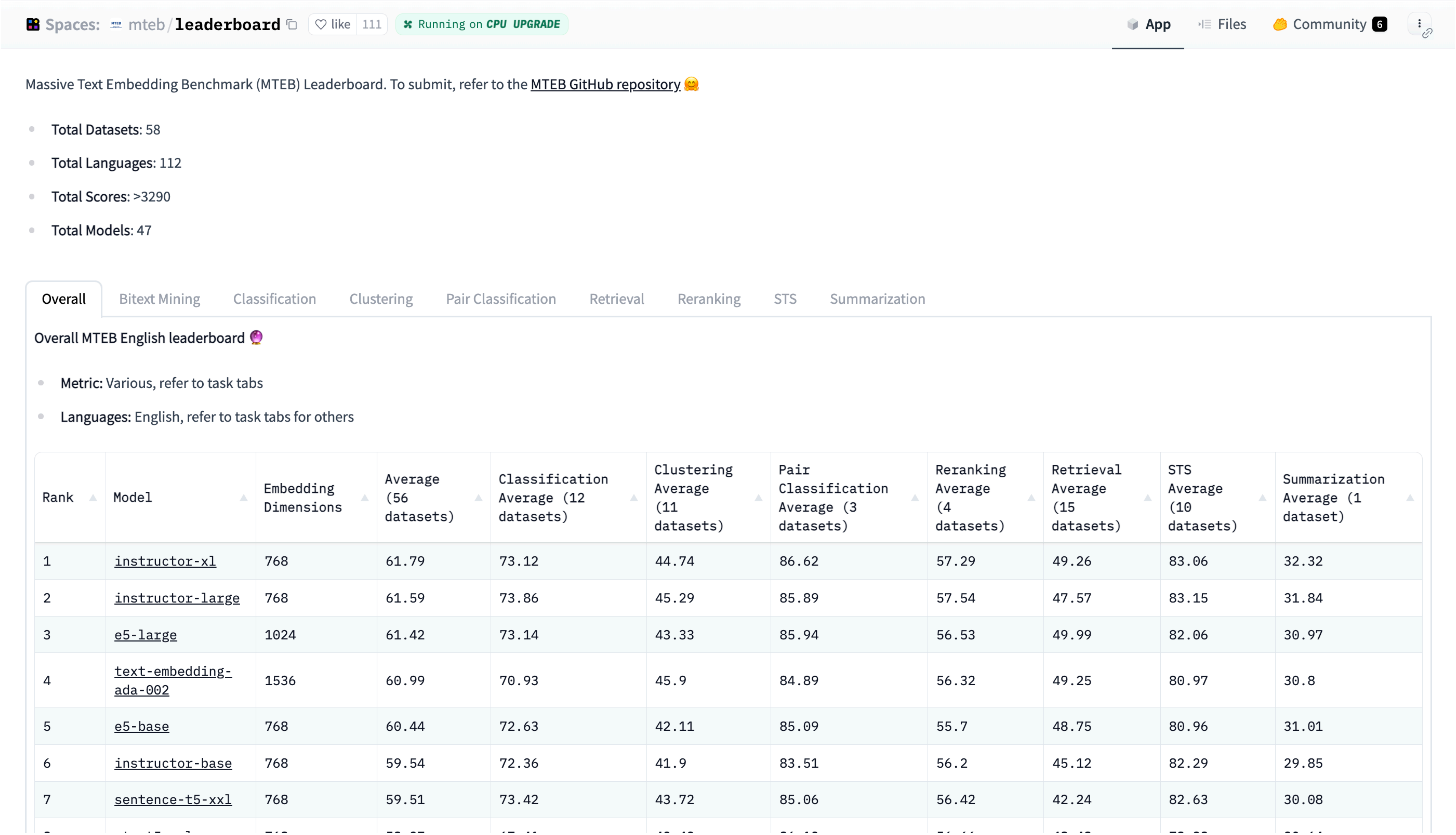
1.1GB

0.1231
0.7412
…
0.6221

What makes a good embedding?

- Utility for the downstream task
- Similar things should be close together

Utility for the downstream task



Massive Text Embedding Benchmark (MTEB) Leaderboard. To submit, refer to the MTEB GitHub repository 🤗

- **Total Datasets:** 58
- **Total Languages:** 112
- **Total Scores:** >3290
- **Total Models:** 47

Overall MTEB English leaderboard 🔮

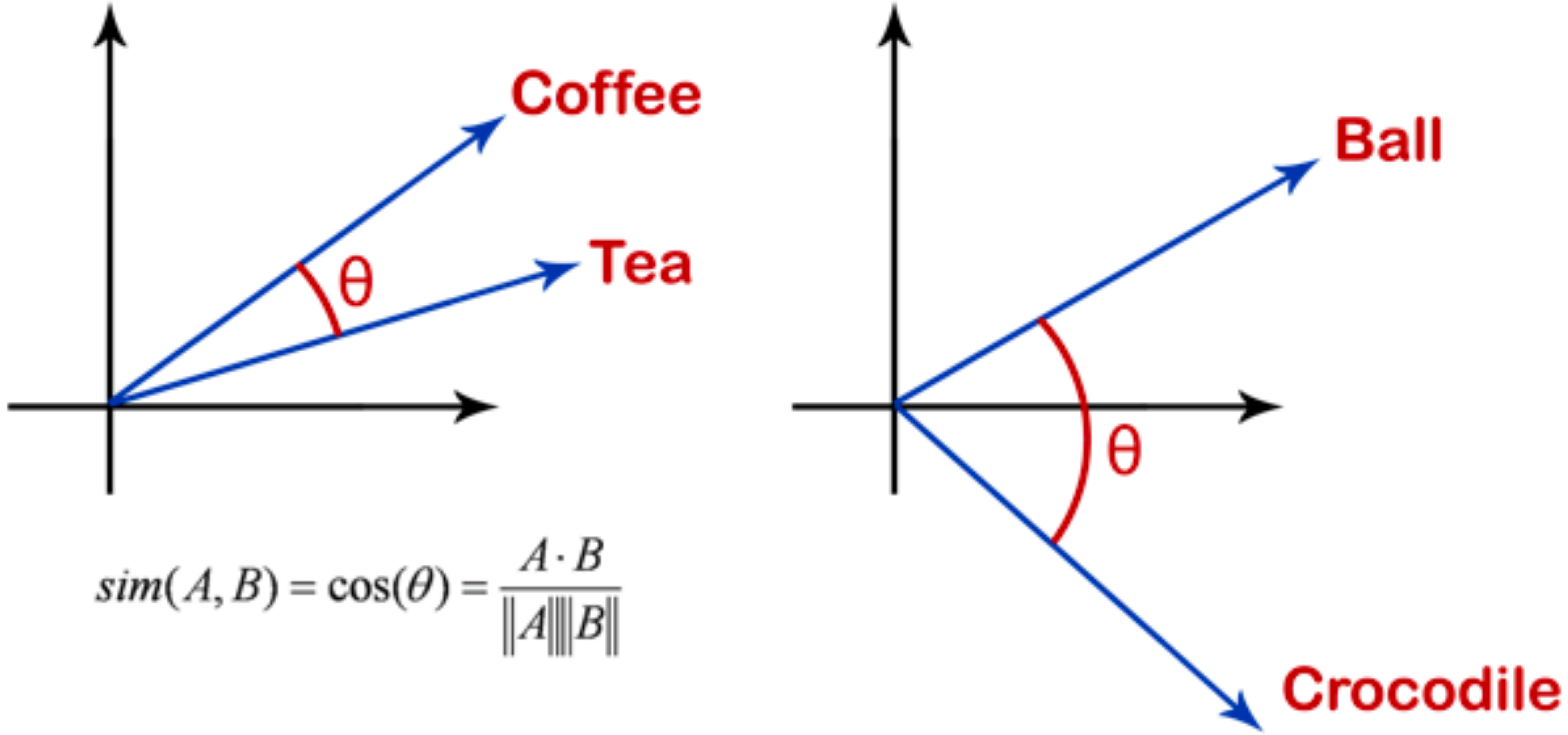
- **Metric:** Various, refer to task tabs
- **Languages:** English, refer to task tabs for others

Rank	Model	Embedding Dimensions	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	Pair Classification Average (3 datasets)	Reranking Average (4 datasets)	Retrieval Average (15 datasets)	STS Average (10 datasets)	Summarization Average (1 dataset)
1	instructor-xl	768	61.79	73.12	44.74	86.62	57.29	49.26	83.06	32.32
2	instructor-large	768	61.59	73.86	45.29	85.89	57.54	47.57	83.15	31.84
3	e5-large	1024	61.42	73.14	43.33	85.94	56.53	49.99	82.06	30.97
4	text-embedding-ada-002	1536	60.99	70.93	45.9	84.89	56.32	49.25	80.97	30.8
5	e5-base	768	60.44	72.63	42.11	85.09	55.7	48.75	80.96	31.01
6	instructor-base	768	59.54	72.36	41.9	83.51	56.2	45.12	82.29	29.85
7	sentence-t5-xxl	768	59.51	73.42	43.72	85.06	56.42	42.24	82.63	30.08

- Use **your task!**
- If you can’t, pick a **broad benchmark**

https://huggingface.co/spaces/mteb/leaderboard

Similar things should be close, different things far



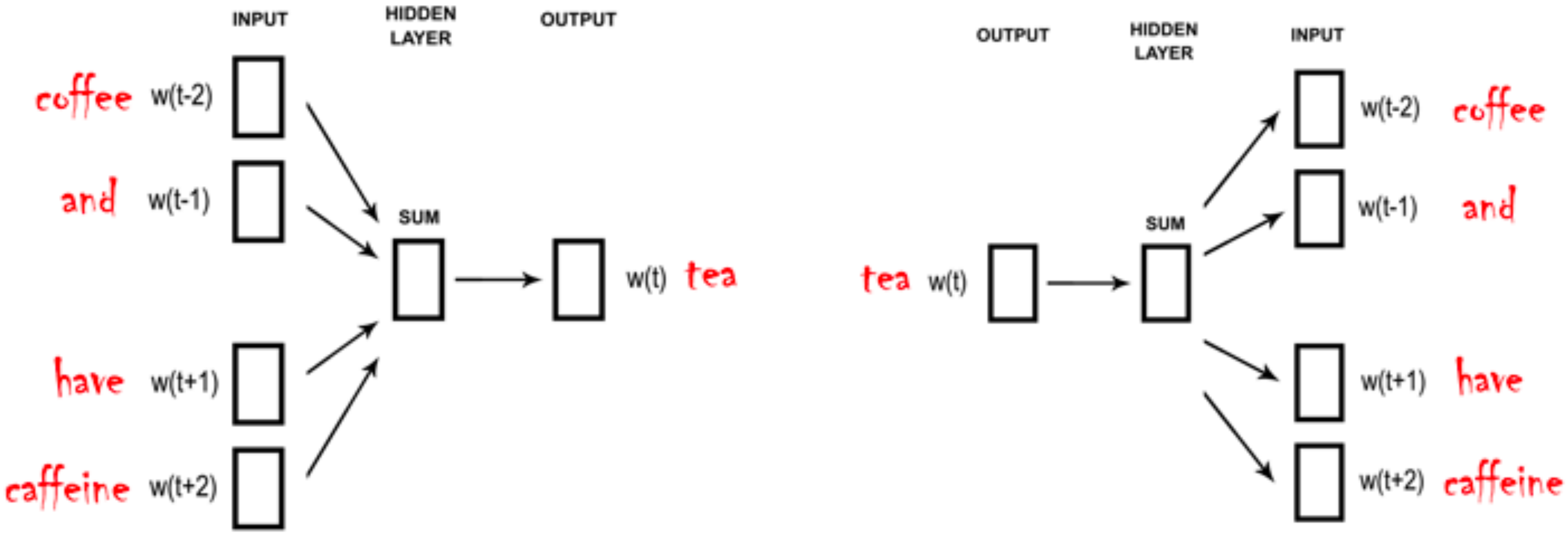
$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Embeddings to know

- The OG: **Word2Vec**
- The baseline: **Sentence transformers**
- A multimodal option: **CLIP**
- The one to use: **OpenAI**
- Where things are going: **Instructor**

An OG embedding: Word2Vec

Try to predict the center word from the surrounding context



https://predictivehacks.com/a-high-level-introduction-to-word-embeddings/

A solid baseline: sentence transformers

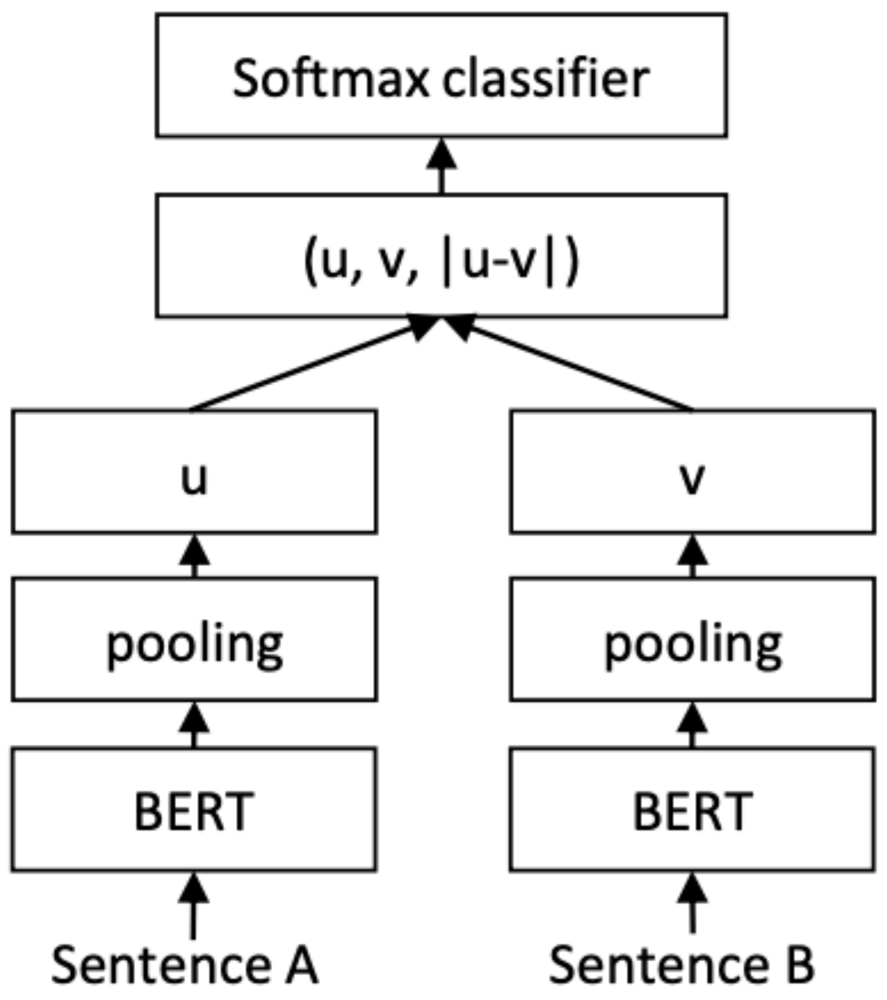


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

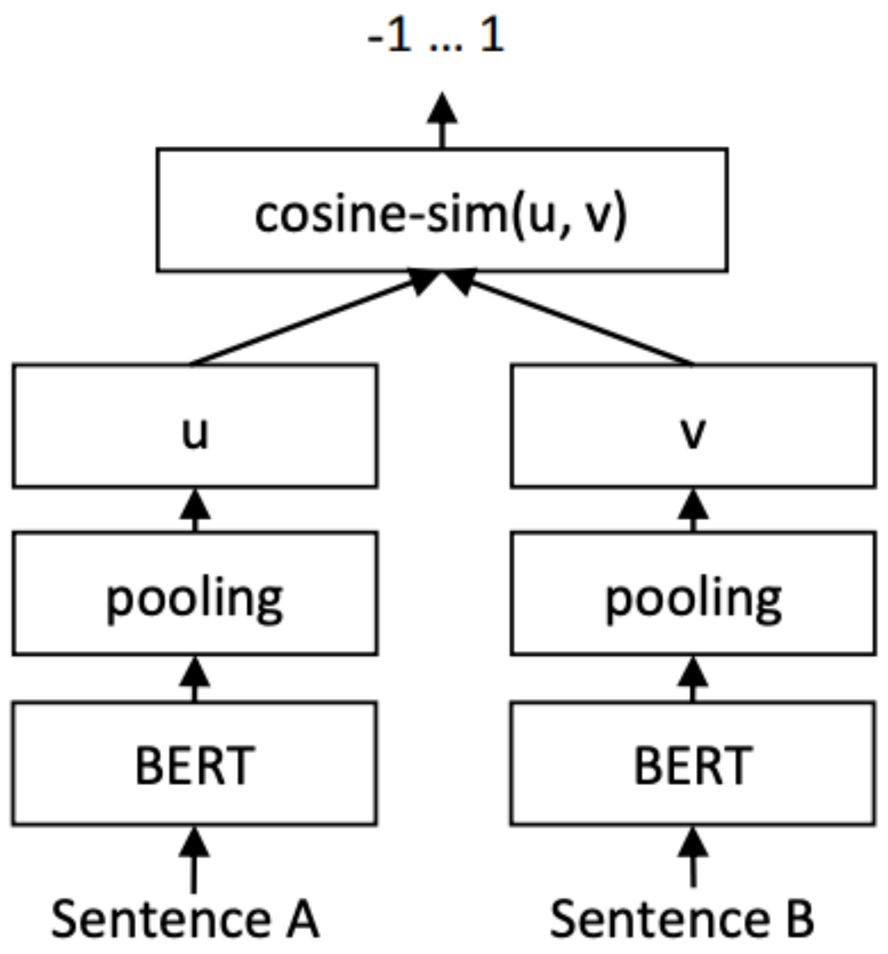


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

- Cheap / fast to run
- Widely available
- Works decently well

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

A multimodal option: CLIP

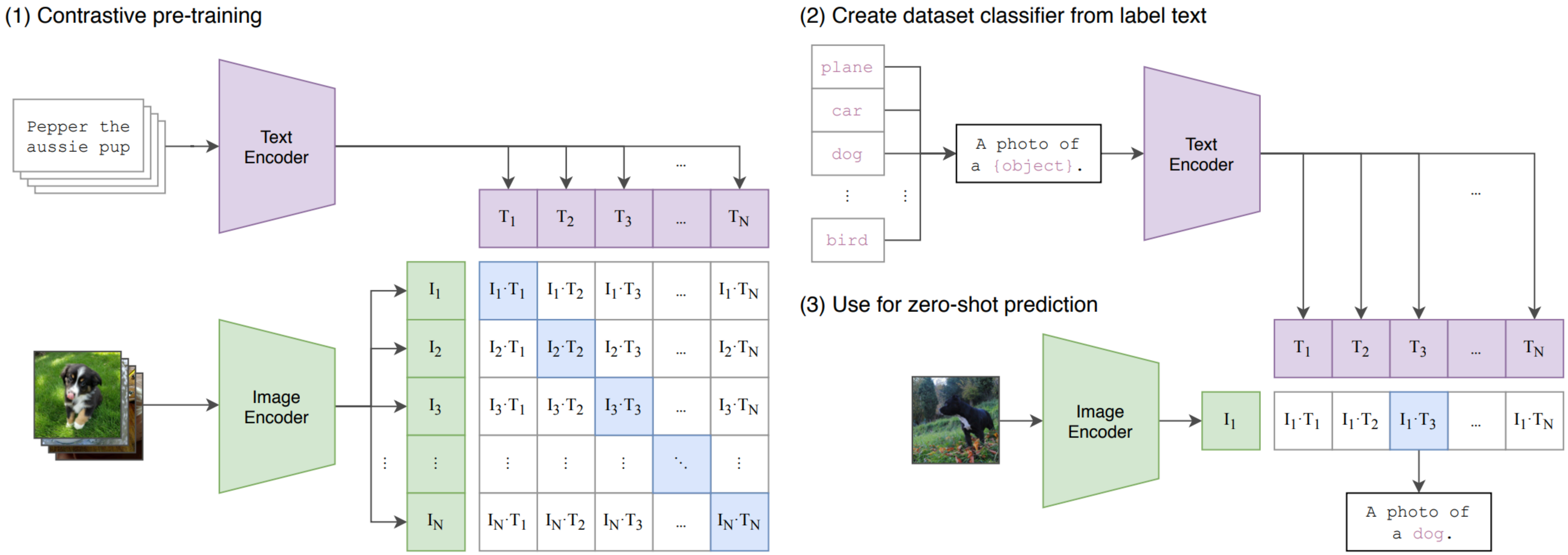

(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
T_1 T_2 T_3 ... T_N
(3) Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

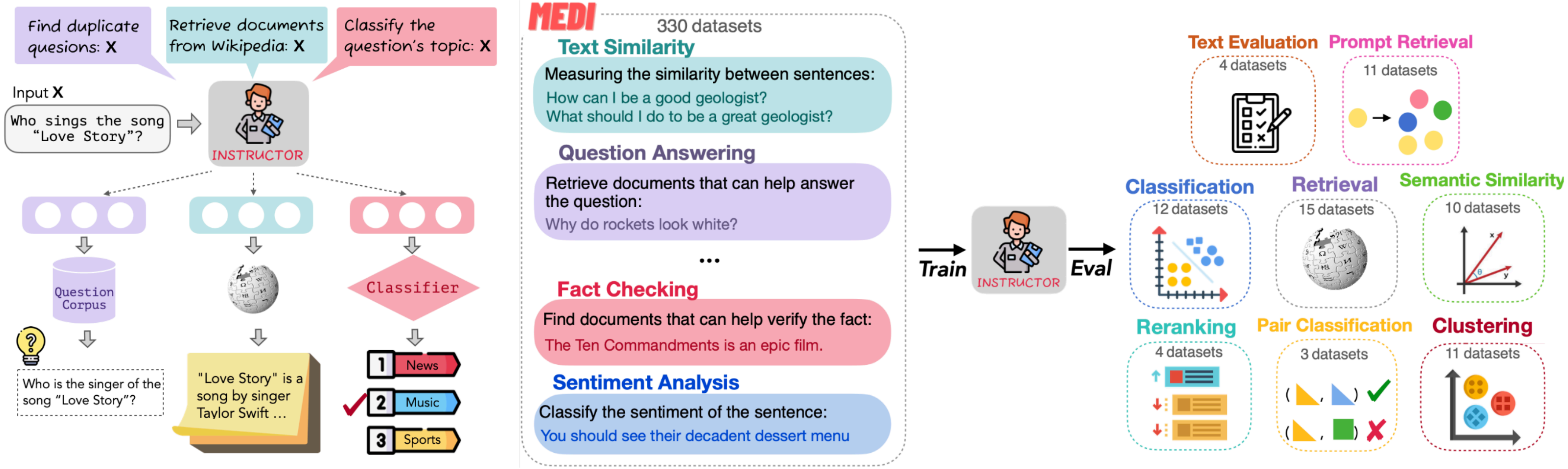
Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Good, fast, and cheap: OpenAI embeddings

- Use **text-embedding-ada-002**
- Near-SoTA
- Easy to use, good results in practice

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

State-of-the-art: Instructor


Find duplicate quesions: X
Retrieve documents from Wikipedia: X
Classify the question's topic: X
Input X
Who sings the song "Love Story"?
INSTRUCTOR
Question Corpus
Classifier
Who is the singer of the song "Love Story"?
"Love Story" is a song by singer Taylor Swift …
1 News
2 Music
3 Sports
MEDI
330 datasets
Text Similarity
Measuring the similarity between sentences:
How can I be a good geologist?
What should I do to be a great geologist?
Question Answering
Retrieve documents that can help answer the question:
Why do rockets look white?
…
Fact Checking
Find documents that can help verify the fact:
The Ten Commandments is an epic film.
Sentiment Analysis
Classify the sentiment of the sentence:
You should see their decadent dessert menu
Train
INSTRUCTOR
Eval
Text Evaluation
4 datasets
Prompt Retrieval
11 datasets
Classification
12 datasets
Retrieval
15 datasets
Semantic Similarity
10 datasets
Reranking
4 datasets
Pair Classification
3 datasets
Clustering
11 datasets

- Prepend the task description to the text, then embed it
- At embedding time, describe your task, get task-specific embeddings

One Embedder, Any Task: Instruction-Finetuned Text Embeddings

Off-the-shelf embeddings: good start, but limited

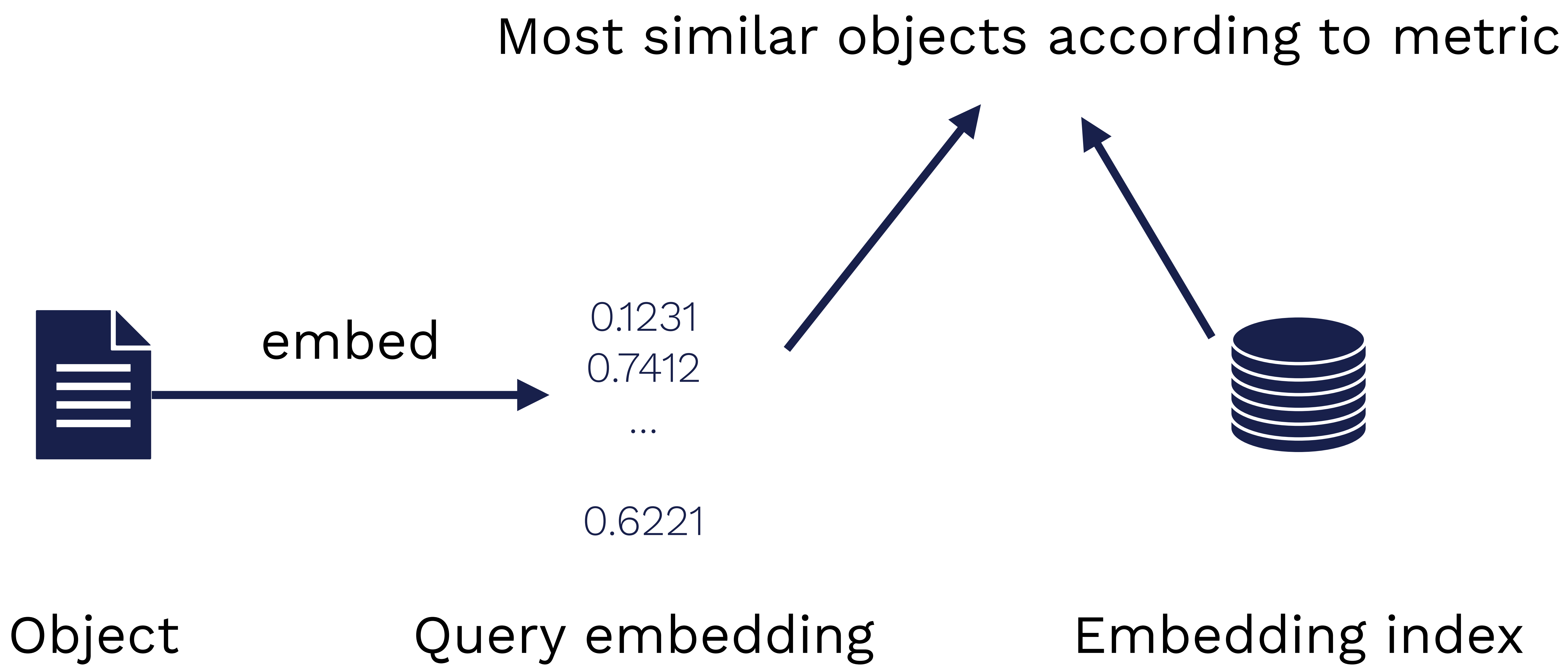
- Off-the-shelf embeddings are not as reliable as off-the-shelf LLMs
- Soon, approaches like Instructor might help
- For now, if retrieval quality is important, you can’t escape training your own

Questions?

AI-powered retrieval via embeddings

- All about embeddings
- **Embedding relevance and indexes**
- Embedding databases
- Beyond naive nearest neighbor

Finding relevant objects with embeddings



Most similar objects according to metric

embed

0.1231
0.7412
...

0.6221

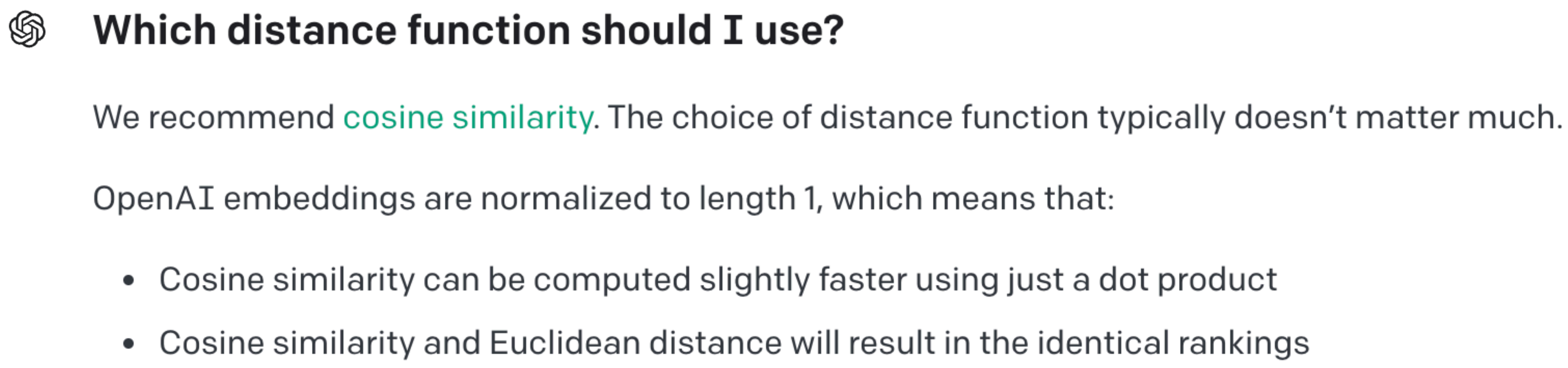
Object

Query embedding

Embedding index

Similarity metrics

- Cosine similarity $\mathbf{A} \cdot \mathbf{B} = \left\|\mathbf{A}\right\| \left\|\mathbf{B}\right\| \cos\theta$
- Dot product
- Euclidean distance
- Hamming distance



Which distance function should I use?

We recommend cosine similarity. The choice of distance function typically doesn’t matter much.

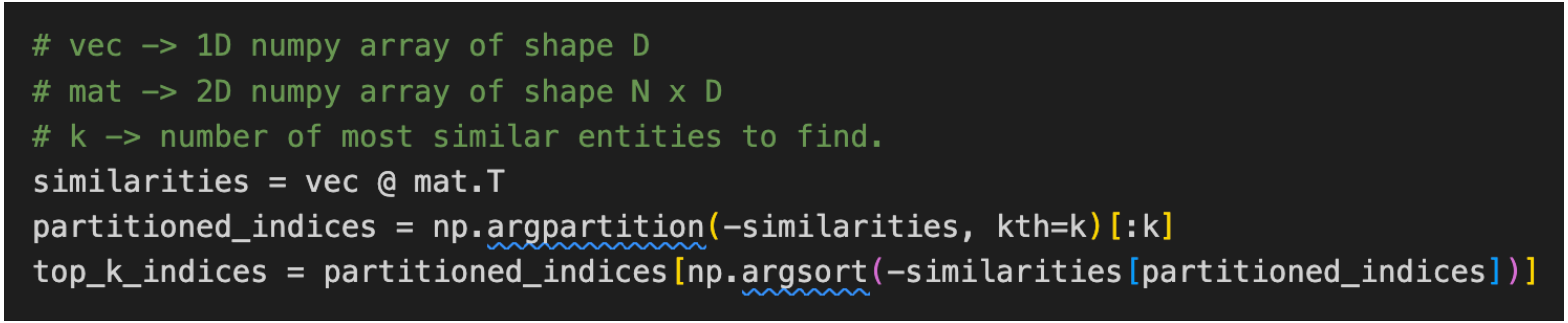
OpenAI embeddings are normalized to length 1, which means that:

- Cosine similarity can be computed slightly faster using just a dot product
- Cosine similarity and Euclidean distance will result in the identical rankings

https://platform.openai.com/docs/guides/embeddings/limitations-risks

A minimal recipe for nearest neighbor similarity

- Embed your corpus
- Store embeddings as an array
- Embed the query, compute dot product with the array

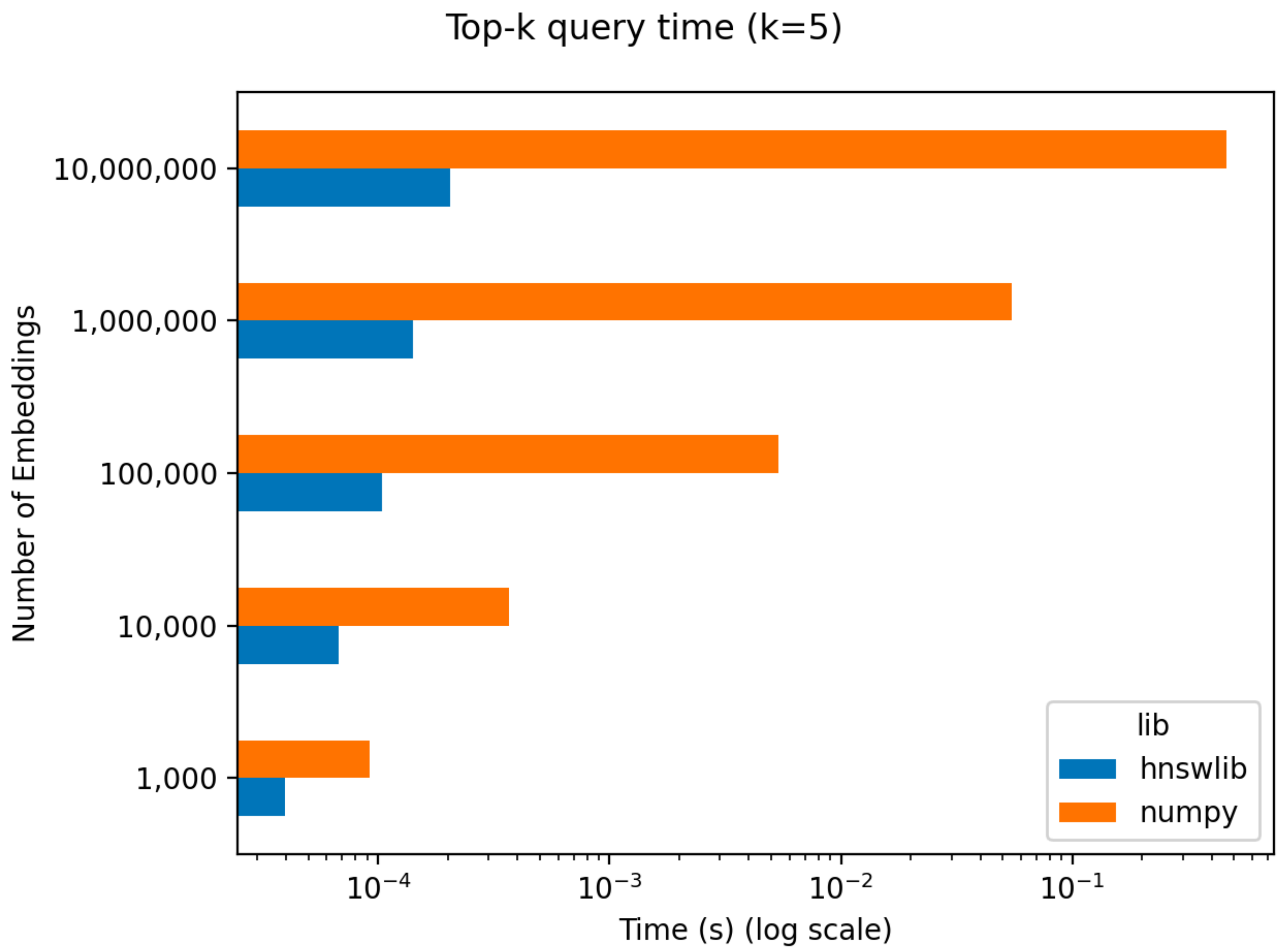


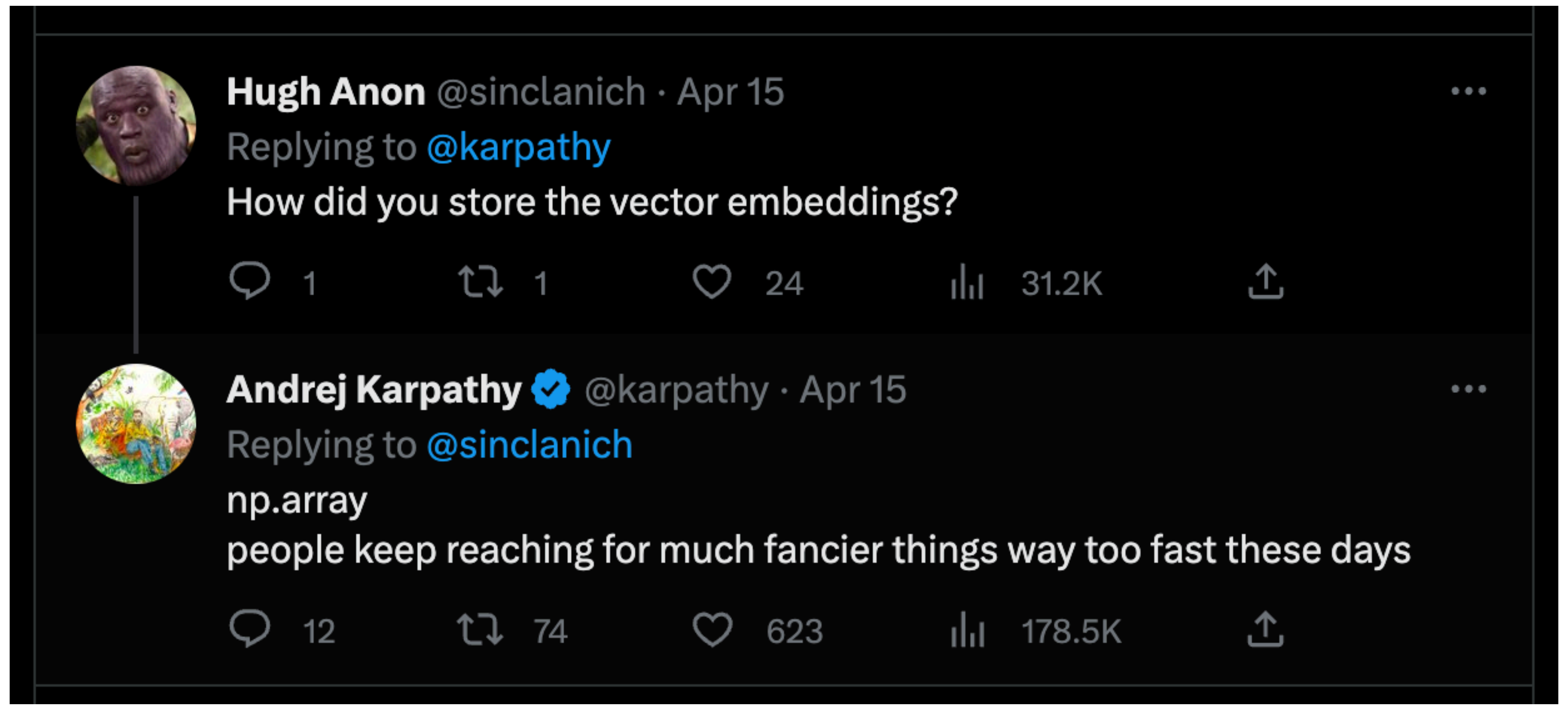
```
# vec -> 1D numpy array of shape D
# mat -> 2D numpy array of shape N x D
# k -> number of most similar entities to find.
similarities = vec @ mat.T
partitioned_indices = np.argpartition(-similarities, kth=k)[:k]
top_k_indices = partitioned_indices[np.argsort(-similarities[partitioned_indices])]
```

https://www.ethanrosenthal.com/2023/04/10/nn-vs-ann/

When do you need more than that?

- If you have <100K vectors or so, you probably won’t notice the difference in speed
- Above a certain scale, it does matter





Hugh Anon @sinclanich · Apr 15
Replying to @karpathy
How did you store the vector embeddings?
1 1 24 31.2K

Andrej Karpathy @karpathy · Apr 15
Replying to @sinclanich
np.array
people keep reaching for much fancier things way too fast these days
12 74 623 178.5K

https://github.com/EthanRosenthal/nn-vs-ann

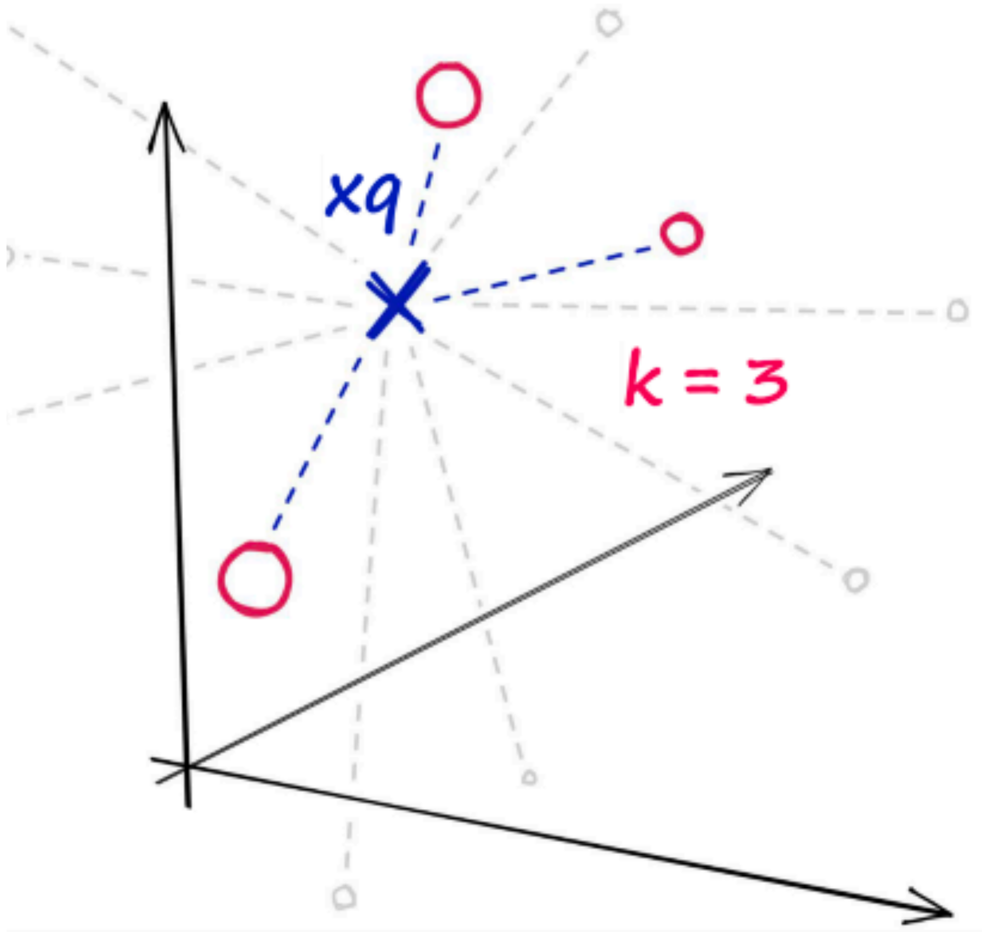
https://www.ethanrosenthal.com/2023/04/10/nn-vs-ann/

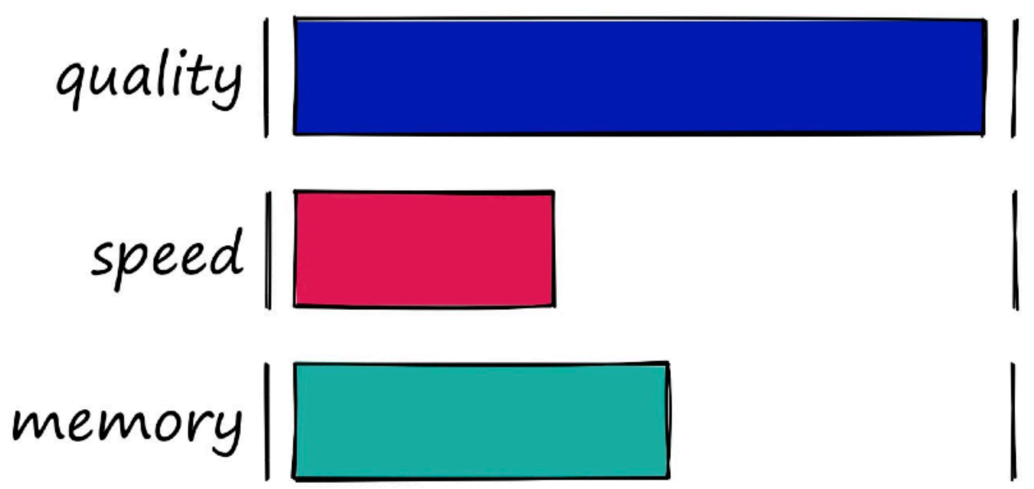
When you need more: approximate nearest neighbors (ANN)

- **Embedding indexes** are data structures that let us perform *approximate* nearest neighbor search
- **Different index types are available** that make different tradeoffs between speed, scalability, and accuracy

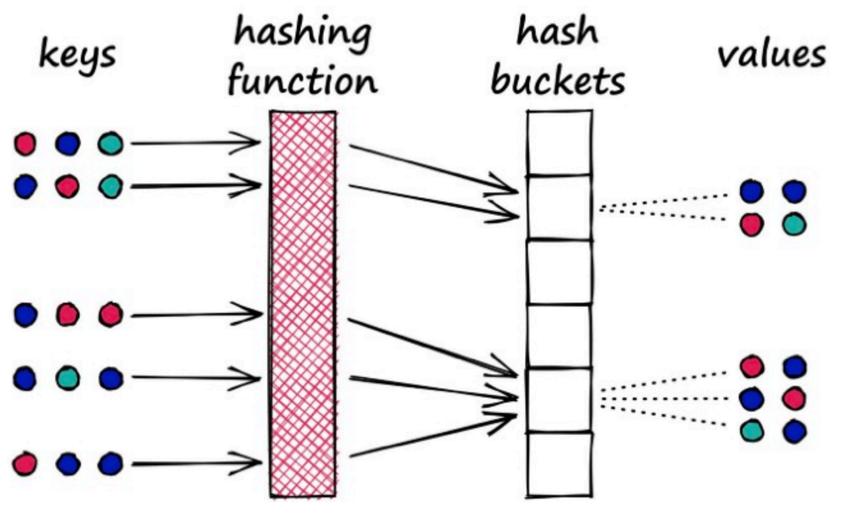
A tour of ANN algorithms

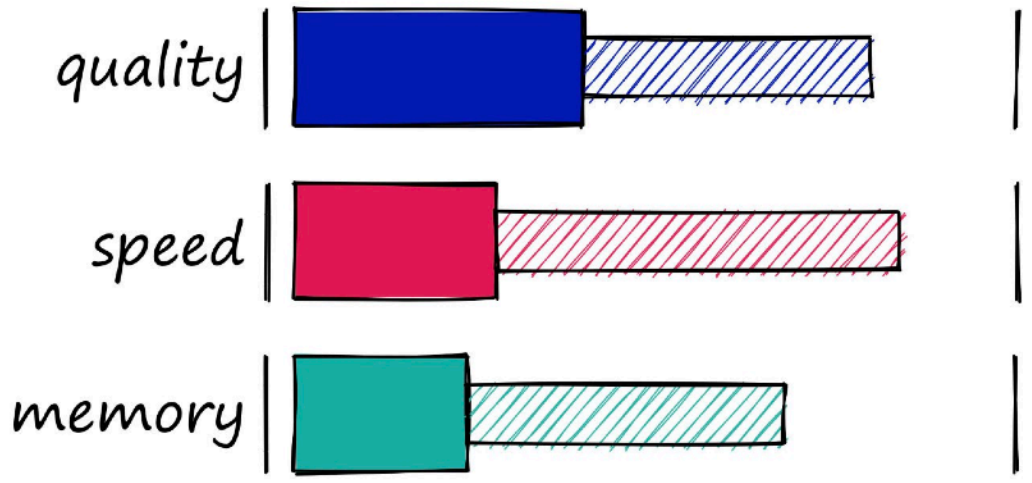
Flat



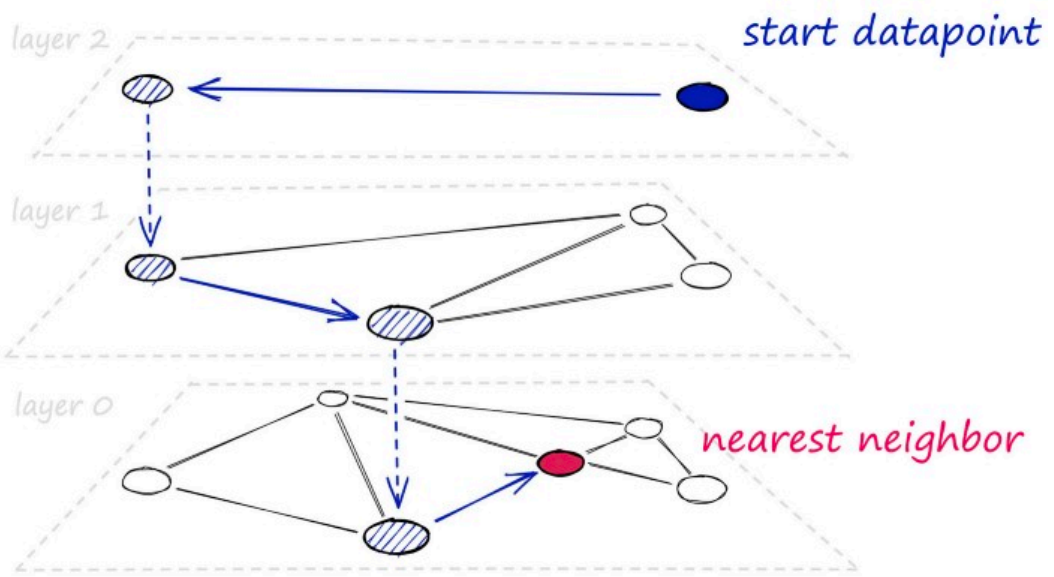


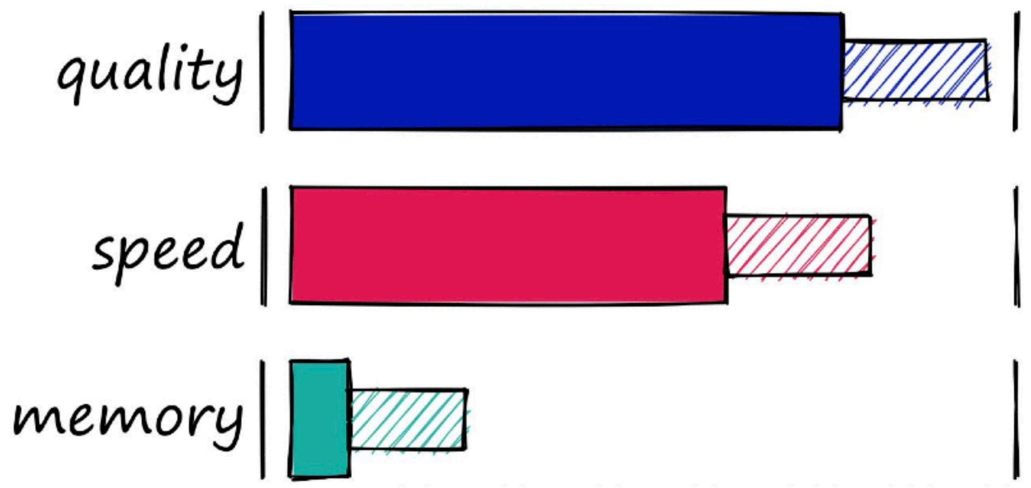
LSH



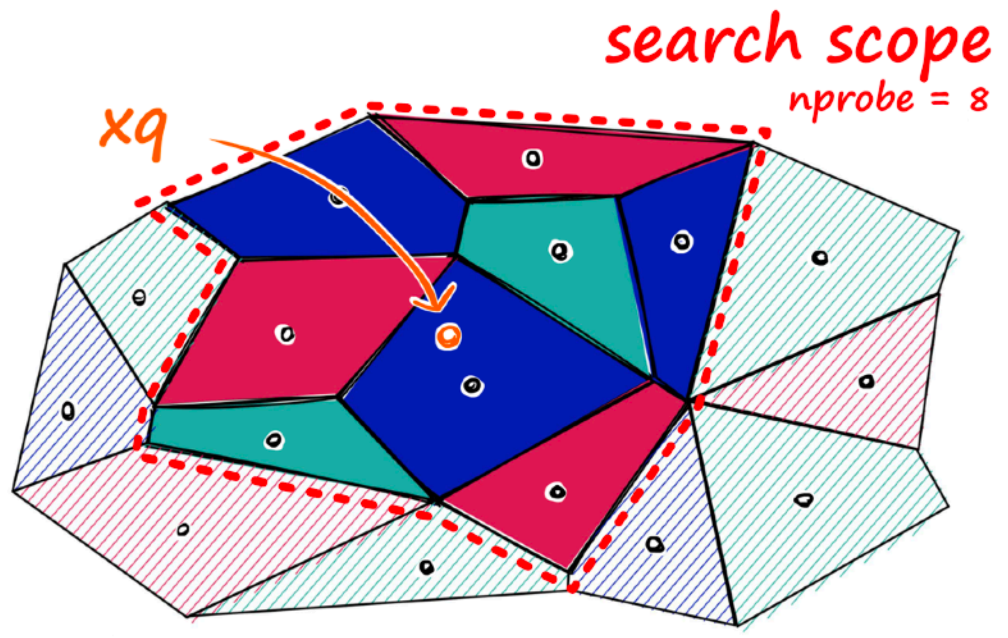


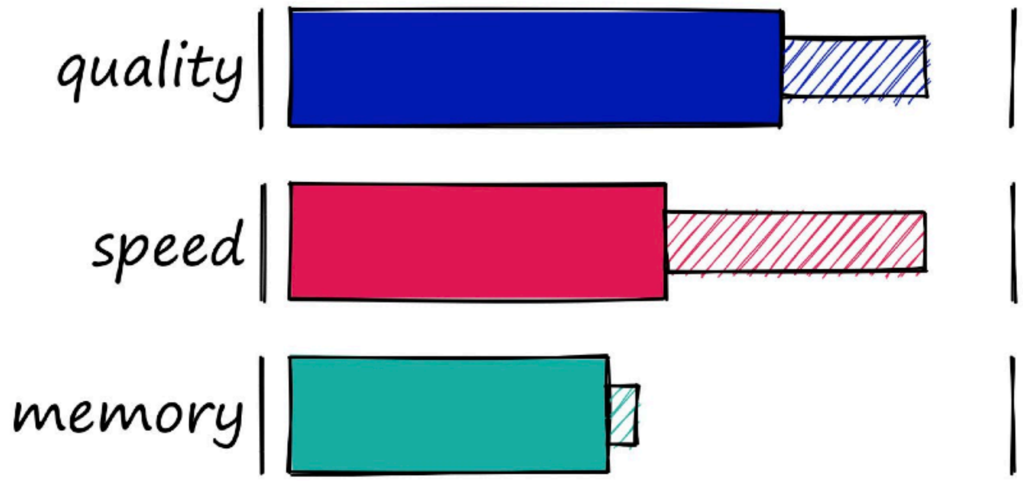
HNSW





IVF



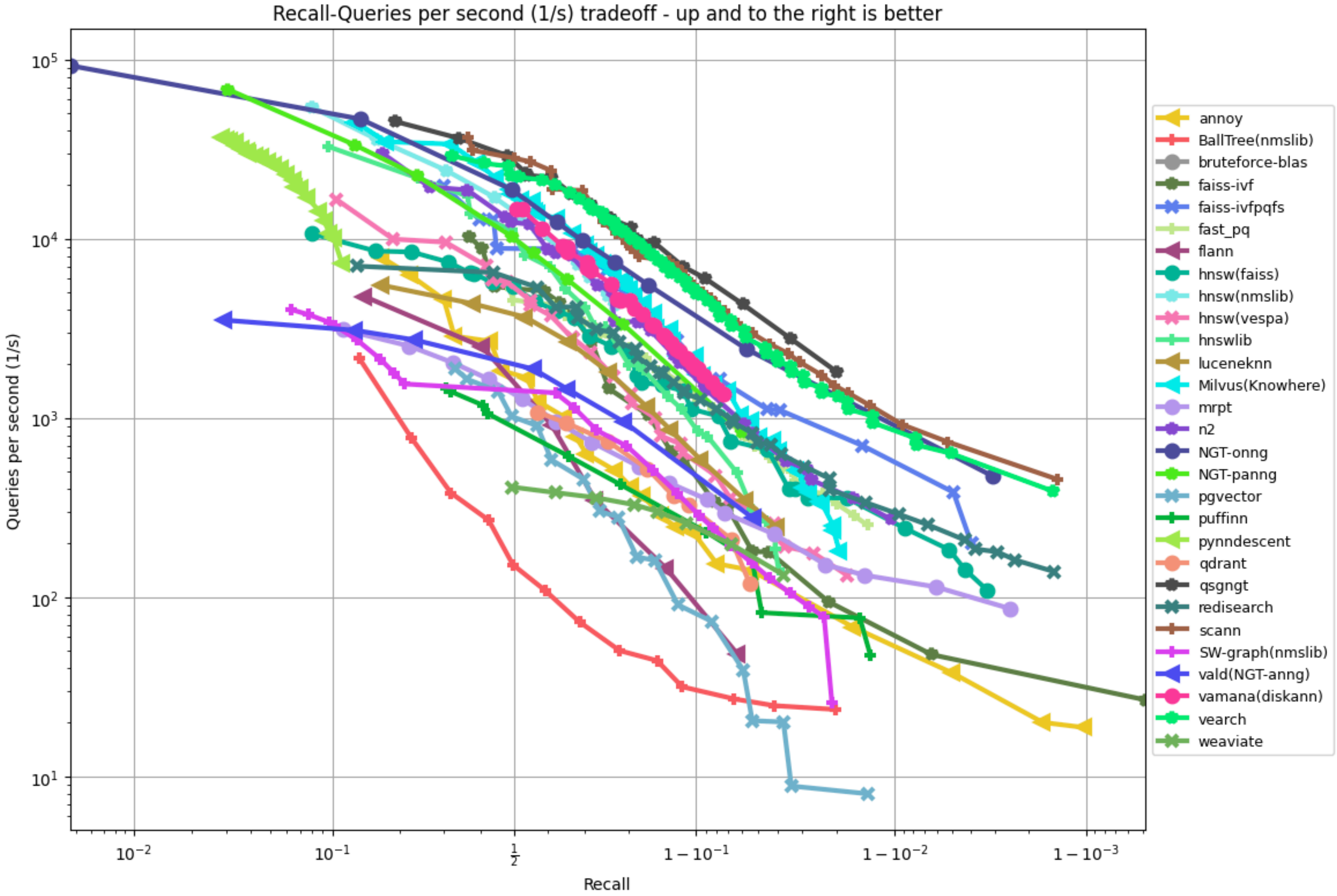


https://www.pinecone.io/learn/vector-indexes/

ANN index tools

- Facebook AI Similarity Search (FAISS)
  - CPU + GPU
  - Supports lots of algos
- Hnswlib
  - Easy to use HNSW implementation
- nmbslib
  - More performant HNSW implementation
- Annoy
  - Very little code + easy but lower performance

ANN index performance benchmarks



https://github.com/erikbern/ann-benchmarks

ANN indexes: recommendations

- Do you need to understand all of this?
  - No, at least not at first
- Which index should I choose?
  - When prototyping, it doesn’t matter. Use numpy if you want
  - When productionizing, the much more important choice is not the index, but the IR system it’s part of
  - If you must choose, FAISS + HNSW is a reasonable start

ANN indexes: recommendations

- Do you need to understand all of this?
  - No, at least not at first
- Which index should I choose?
  - When prototyping, it doesn't matter. Use numpy if you want
  - When productionizing, the much more important choice is not the index, but the IR system it's part of
  - If you must choose, FAISS + HNSW is a reasonable start

Limitations of ANN indexes

- **They're just a data structure**. They do *not* offer:
  - Hosting
  - Storing data / metadata alongside vectors
  - Combining sparse + dense retrieval
  - Managing the embedding functions themselves
  - Vertical / horizontal scaling
- Beyond prototyping, **you'll want an IR system / database that supports more of these**

Database

Index

Questions?

AI-powered retrieval via embeddings

- All about embeddings
- Embedding relevance and indexes
- **Embedding databases**
- Beyond naive nearest neighbor

Do you need an “embedding database” or just a database?

- Elasticsearch, postgres, redis run NN / ANN
- You’re probably using one of those already
- Won’t work for the most complicated queries or highest scale, but **will probably work for you**

pgvector / pgvector Public

elasticsearch

redis

The database you want is probably the one you’re using already

If you really need something more advanced...

The dream of how this would work

Dump in a bunch of data

📄 📷

🔊 📹

Run a query

Descriptions of recent concerts in the bay area

Get the most relevant data back

[{“document_id”:
“12345”,
“snippet”: “…”},
…

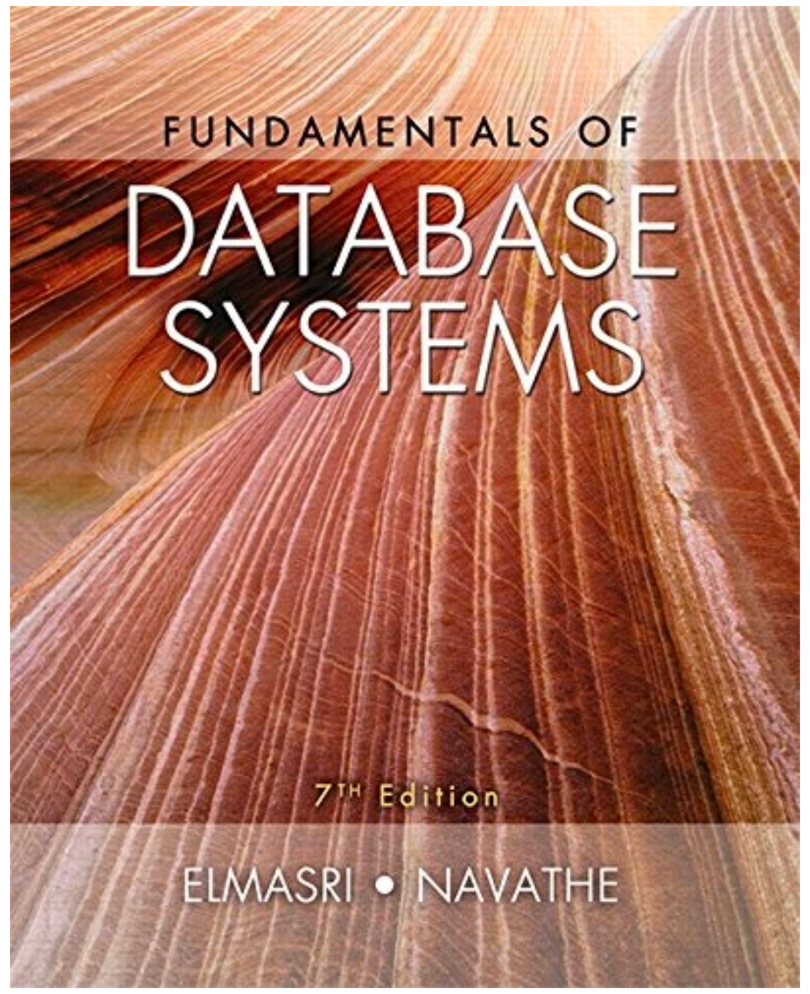
]

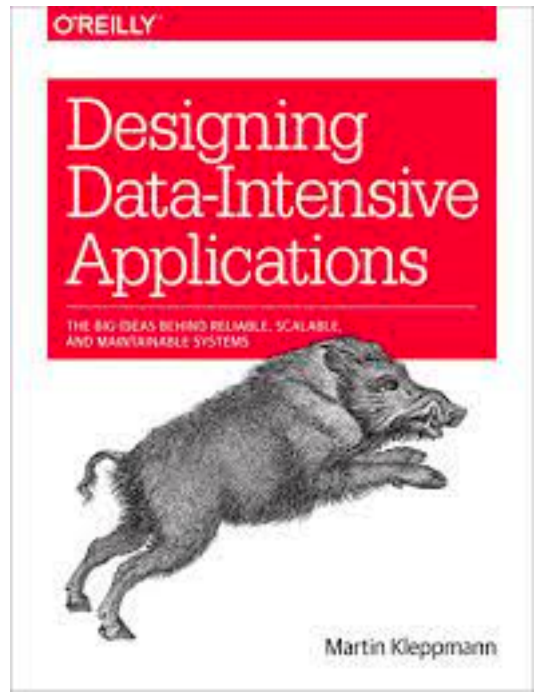
Challenges

Dump in a bunch of data	• Database stuff: scale, reliability • Document splitting & embedding mgmt
Run a query	• Query language
Get the most relevant data back	• Search algorithm

“Database stuff”

- **Scale**
  - Storing data in memory only goes so far, but splitting across machines introduces lots of complexity (strong vs eventual consistency, etc)
- **Reliability**
  - What happens if the machine storing your data goes down?

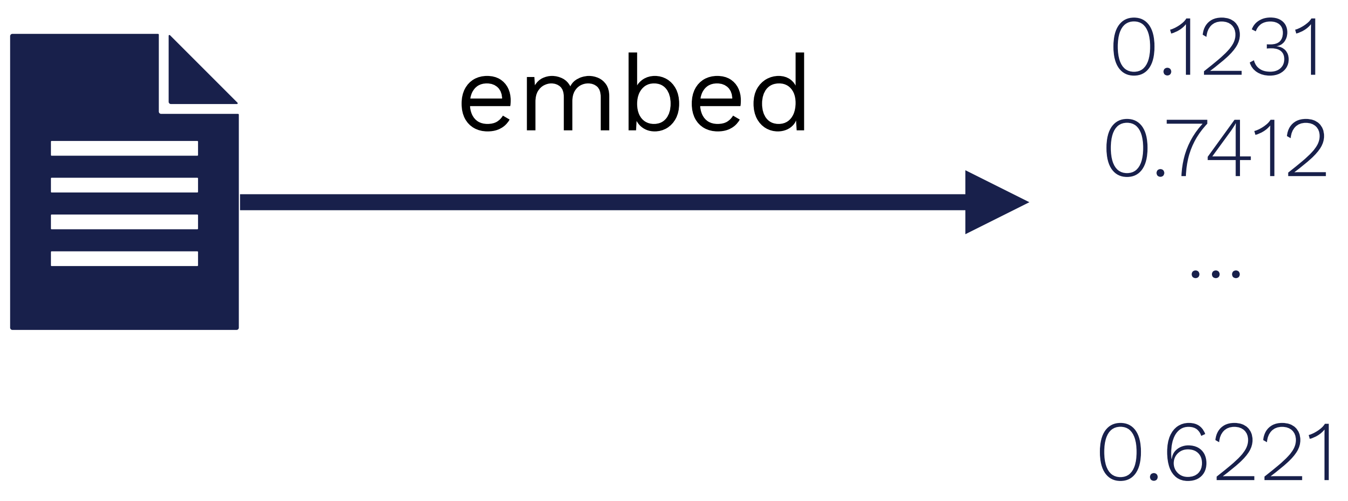



O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann

Managing the embedding

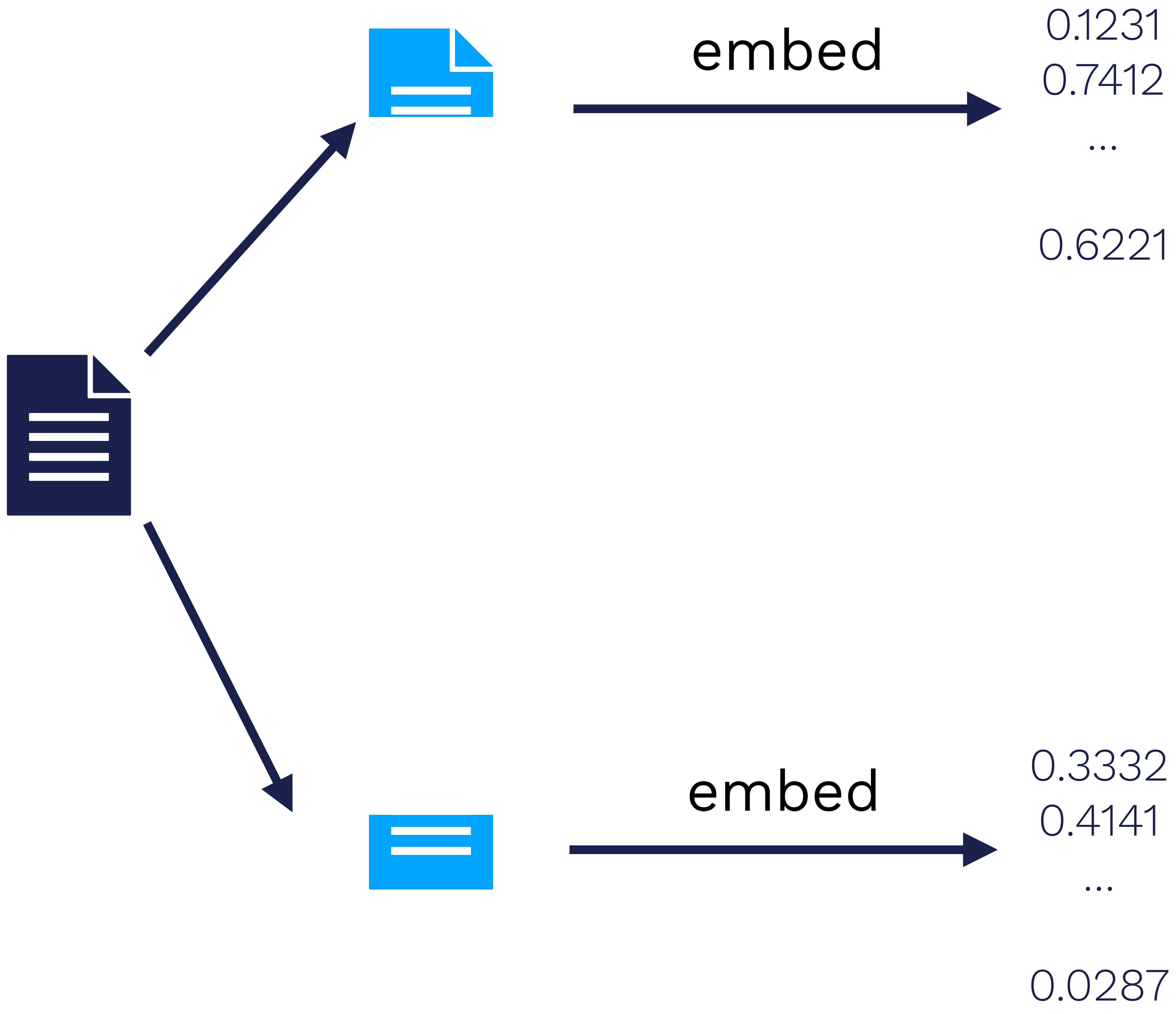
- What embedding should we use for different data types?
- What happens if we want to change the embedding function?

Document splitting



What happens if the document is **too long** to fit in the context for the embedding model?

Document splitting



- Pick a separator (“\n”)
- Split text by separator up to max size chunks
- Advanced: try to make the chunks more semantically consistent

https://python.langchain.com/en/latest/modules/indexes/text_splitters.html

Query language

- “most similar documents” is easy
- What if you want to filter on other metadata (recency, etc?)
- What if you don’t have a query document in mind, you just have:
  - A search string
  - A request for a summary
  - Etc

Search algorithm

- How to deal with hierarchical structure in the index?
  - What if all of the K-NNs are chunks from the same doc?
  - What if all of the docs are from the same corpus?
  - What if all of the docs are old?

Managed embedding databases

Tool	Prominent users	DB features	Embedding mgmt	Sql-like Filtering	Full text search	It's for…
Chroma	N/A	✓	✓	✓	✗	Betting on the most "AI-native" tool in the category
Milvus	ebay Walmart	✓	✗	✓	✗	Scale & enterprise
Pinecone	shopify GONG	✓	✗	✓	✗	Fastest to get started
vespa	yahoo! Spotify R&D	✓	✓	✓	✓	Battle-tested; most powerful
weaviate	N/A	✓	✓	✓	✓	Embedding mgmt and flexible GraphQL-like query interface

Embedding databases: recommendations

- When you are ready to move on from prototyping, move to a database you are using already (Postgres / elastic / redid)
- If you don’t have one of those databases, then move to Pinecone for speed of setup
- When you’re ready to “upgrade”, consider why:
  - Need more flexible queries? Try Vespa or Weaviate
  - Need more scale / reliability? Try Vespa or Milvus

Questions?

AI-powered retrieval via embeddings

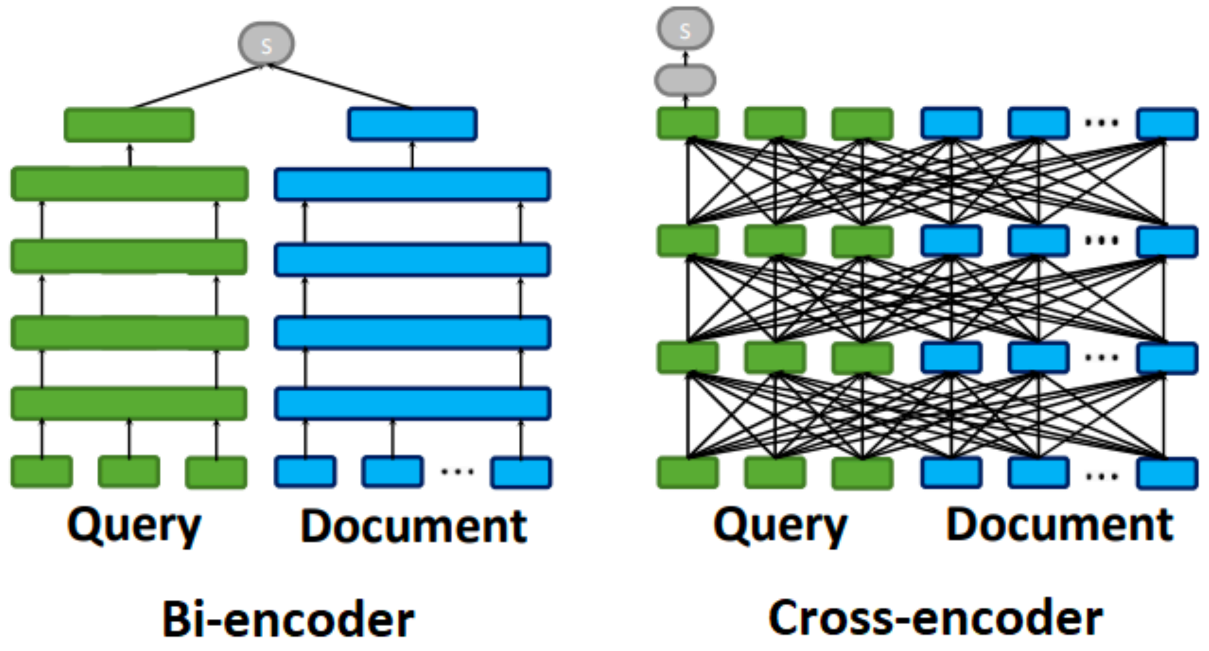
- All about embeddings
- Embedding relevance and indexes
- Embedding databases
- **Beyond naive nearest neighbor**

Problem: your queries and docs are not the same

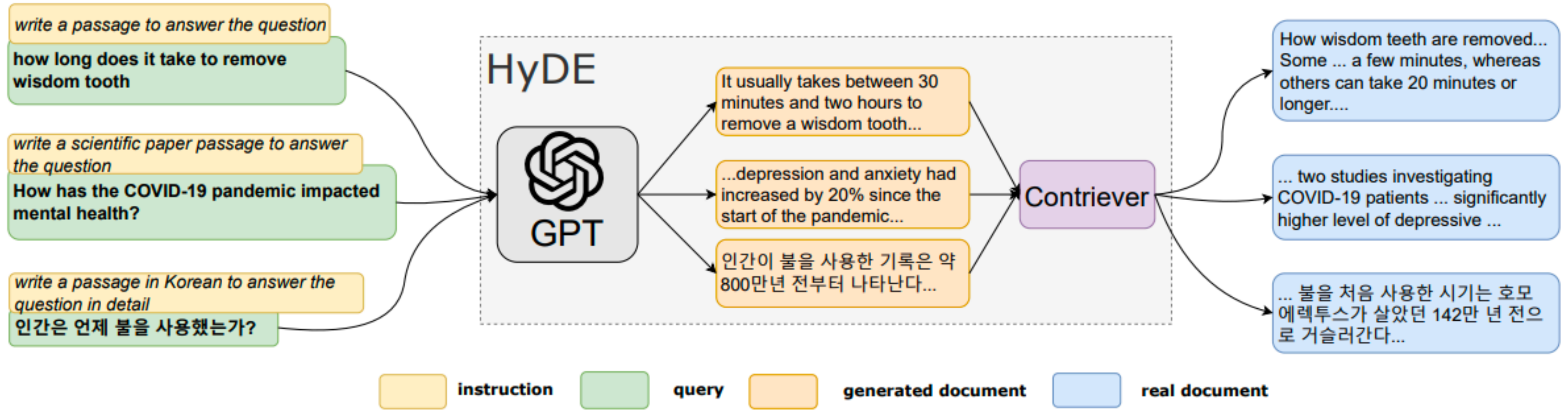
- Your queries are short questions
- Your docs are long form
- Embeddings are not that comparable
  - Your data might be totally different than what the pre-trained embedding expects!

Some approaches to address this

Joint training



Hypothetical document embeddings (HyDE)


write a passage to answer the question

how long does it take to remove wisdom tooth

write a scientific paper passage to answer the question

How has the COVID-19 pandemic impacted mental health?

write a passage in Korean to answer the question in detail

인간은 언제 불을 사용했는가?

HyDE

GPT

It usually takes between 30 minutes and two hours to remove a wisdom tooth...

...depression and anxiety had increased by 20% since the start of the pandemic...

인간이 불을 사용한 기록은 약 800만년 전부터 나타난다...

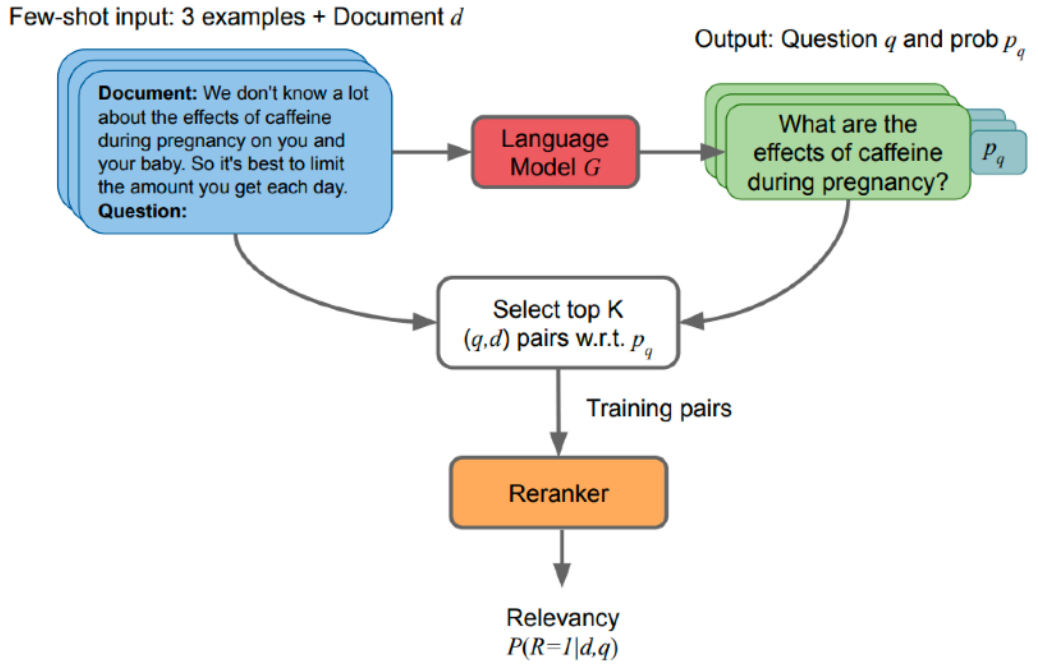
Contriever

How wisdom teeth are removed... Some ... a few minutes, whereas others can take 20 minutes or longer....

... two studies investigating COVID-19 patients ... significantly higher level of depressive ...

... 불을 처음 사용한 시기는 호모 에렉투스가 살았던 142만 년 전으로 거슬러간다...

Re-ranking


Few-shot input: 3 examples + Document d

Document: We don't know a lot about the effects of caffeine during pregnancy on you and your baby. So it's best to limit the amount you get each day.
Question:

Language Model G

Output: Question q and prob p_q

What are the effects of caffeine during pregnancy?

p_q

Select top K (q,d) pairs w.r.t. p_q

Training pairs

Reranker

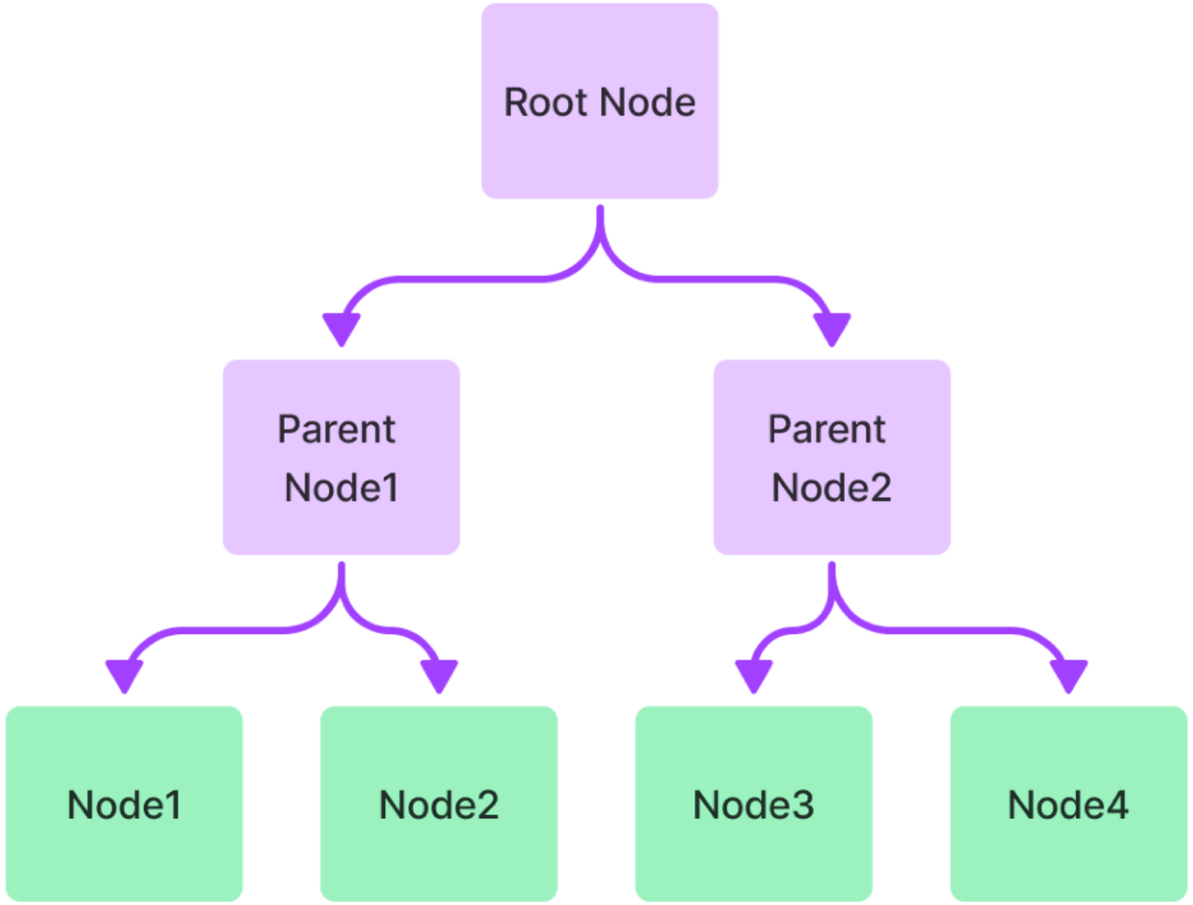
Relevancy
$P(R=1|d,q)$

https://blog.reachsumit.com/posts/2023/03/llm-for-text-ranking/

Problem: you might have some structure to your data

LlamaIndex

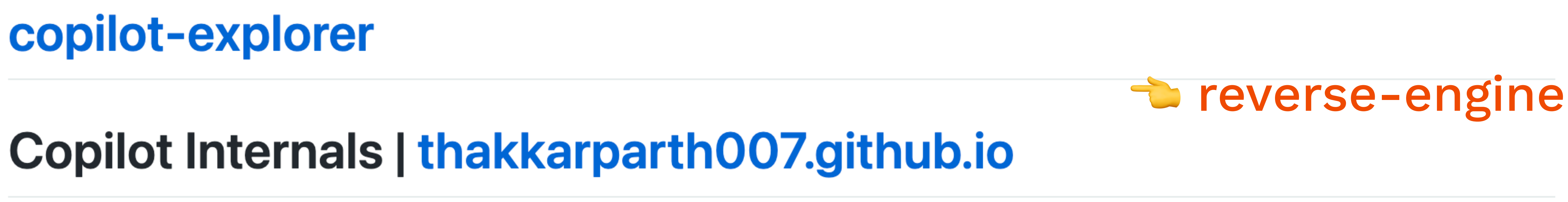
- Searching over the whole index at the same time might not be efficient
- Instead, search in a way that respects the structure of your data



Questions?

D. Patterns and case studies

Copilot case study



copilot-explorer

Copilot Internals | thakkarparth007.github.io

👈 reverse-engineer copilot, explore how they built it

How to get the relevant context into copilot?

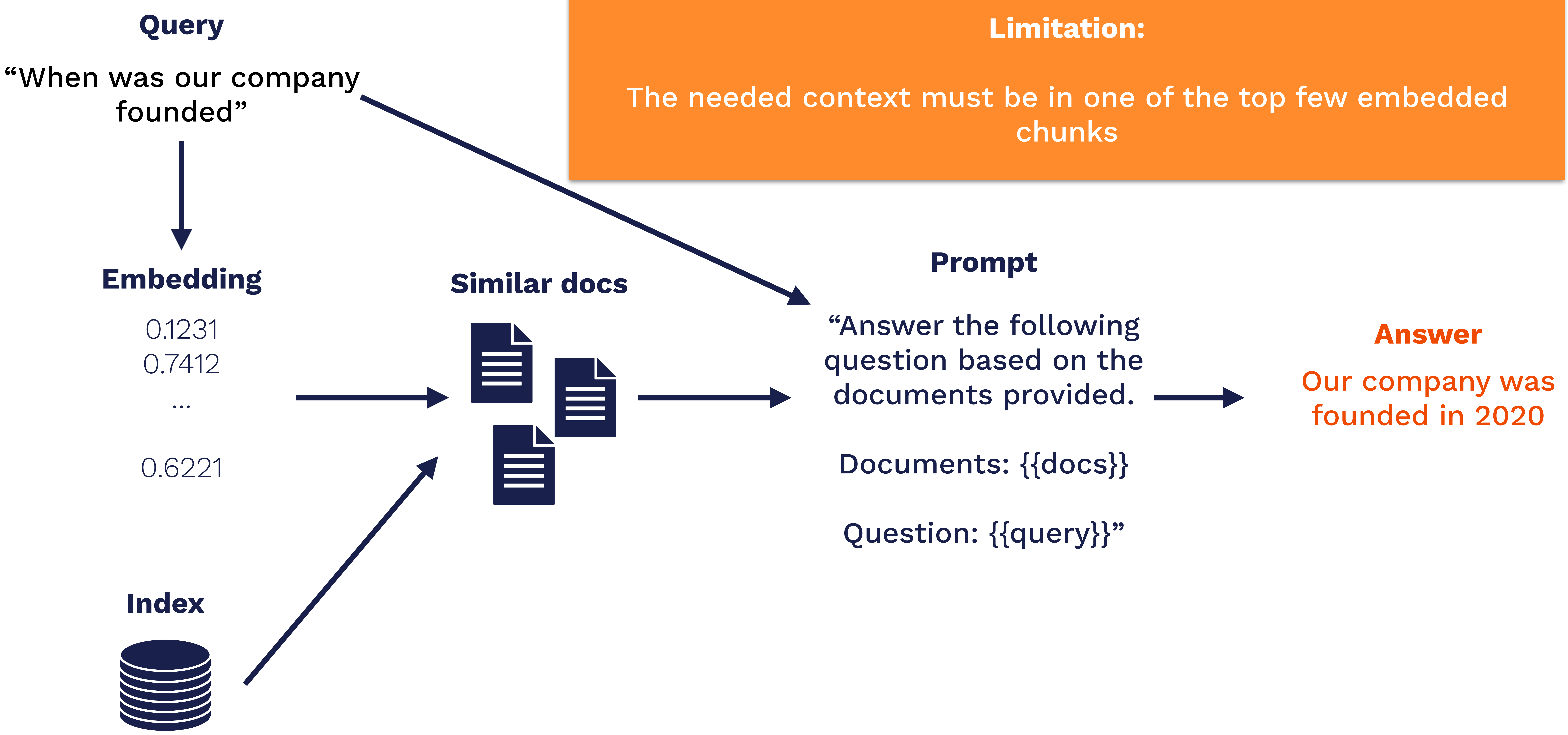
1. **Retrieval.** Most recently accessed 20 docs w/ same language

2. **Postprocessing.** Local context like what was before and after the cursor, “most relevant” snippet from the candidate docs (NN similarity)

3. **Ranking.** Sort the prompt wishlist using heuristics to give you the max information

https://thakkarparth007.github.io/copilot-explorer/posts/copilot-internals

Retrieval augmentation question answering pattern



Limitation:

The needed context must be in one of the top few embedded chunks

Query

“When was our company founded”

Embedding

0.1231
0.7412
…

0.6221

Index

Similar docs

Prompt

“Answer the following question based on the documents provided.

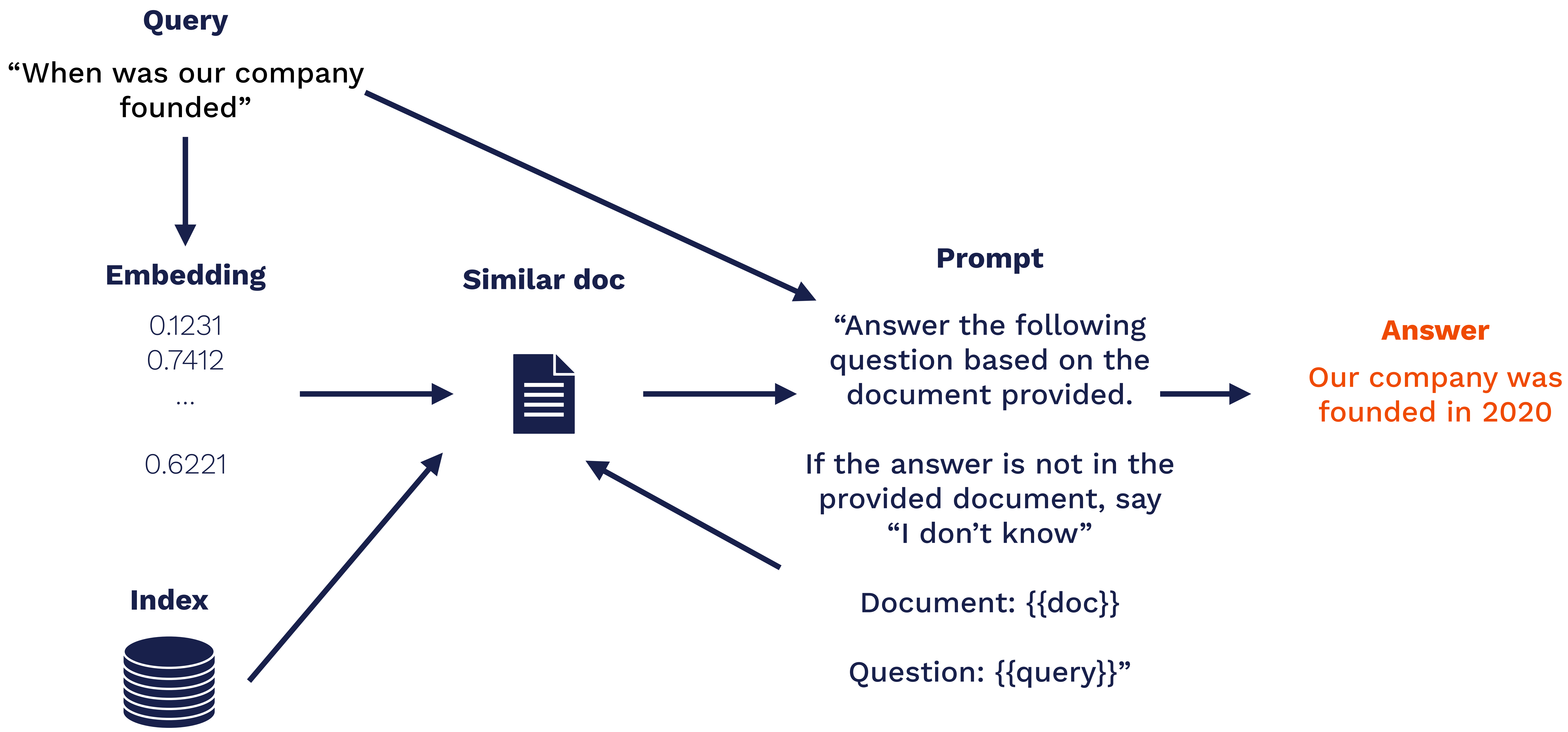
Documents: {{docs}}

Question: {{query}}”

Answer

Our company was founded in 2020

Getting around the limitation



Query

“When was our company founded”

Embedding

0.1231
0.7412
…

0.6221

Index

Similar doc

Prompt

“Answer the following question based on the document provided.

If the answer is not in the provided document, say “I don’t know”

Document: {{doc}}

Question: {{query}}”

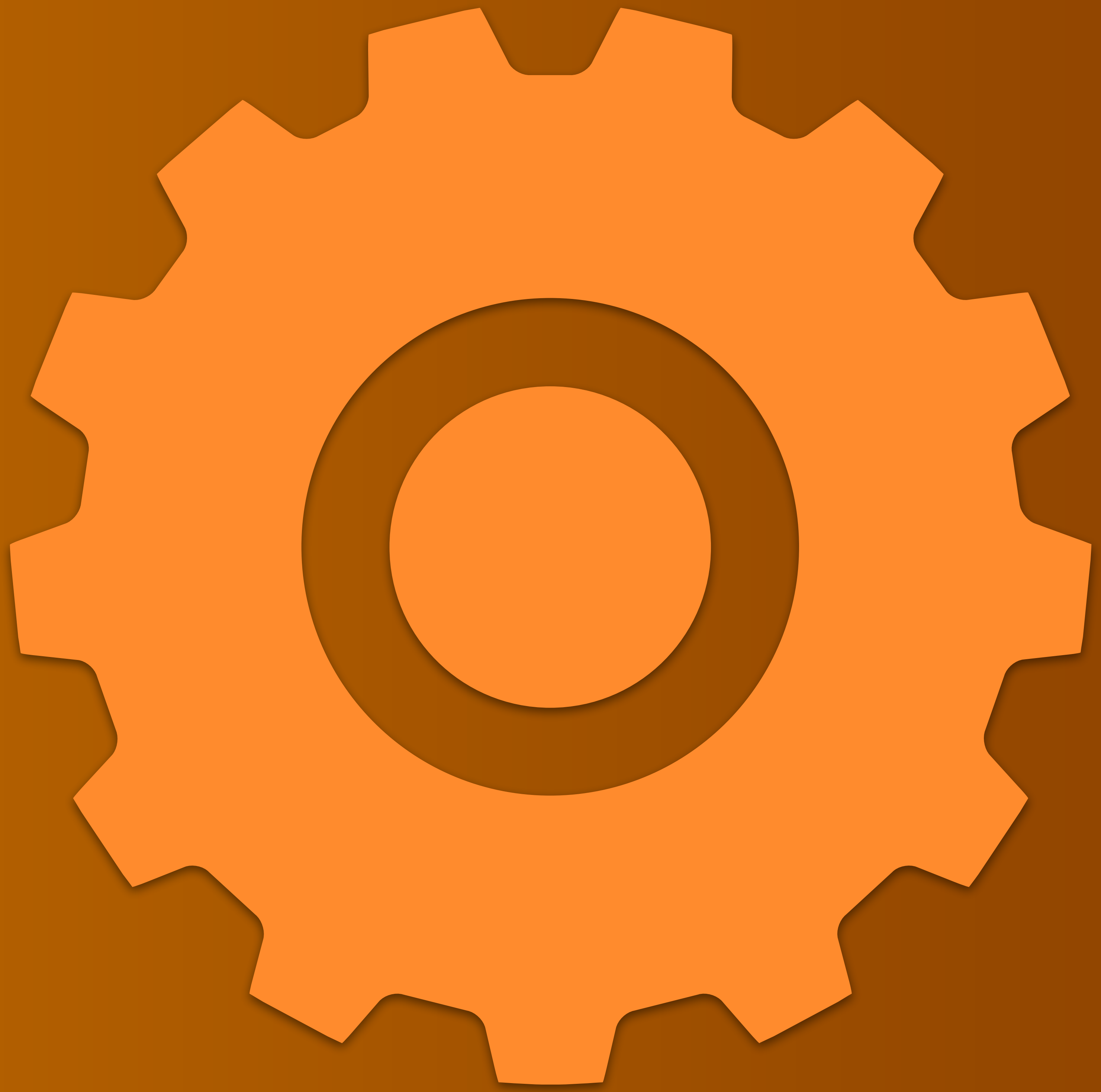
Answer

Our company was founded in 2020

Questions?

02

Chains



Building chains of LLM calls

- Sometimes the best context for your LLM doesn’t exist directly in your corpus
- Instead, the best context for your LLM might be the output of another LLM!

Example patterns for building chains

- The QA pattern
  - Question ➡️ embedding ➡️ similar docs ➡️ QA prompt
- Hypothetical document embeddings (HyDE)
  - Question ➡️ document generating prompt ➡️ rest of QA chain
- Summarization
  - Document corpus ➡️ apply a summarization prompt to each ➡️ pass all document summaries to another prompt ➡️ get global summary back

Tools for building chains of models

LangChain

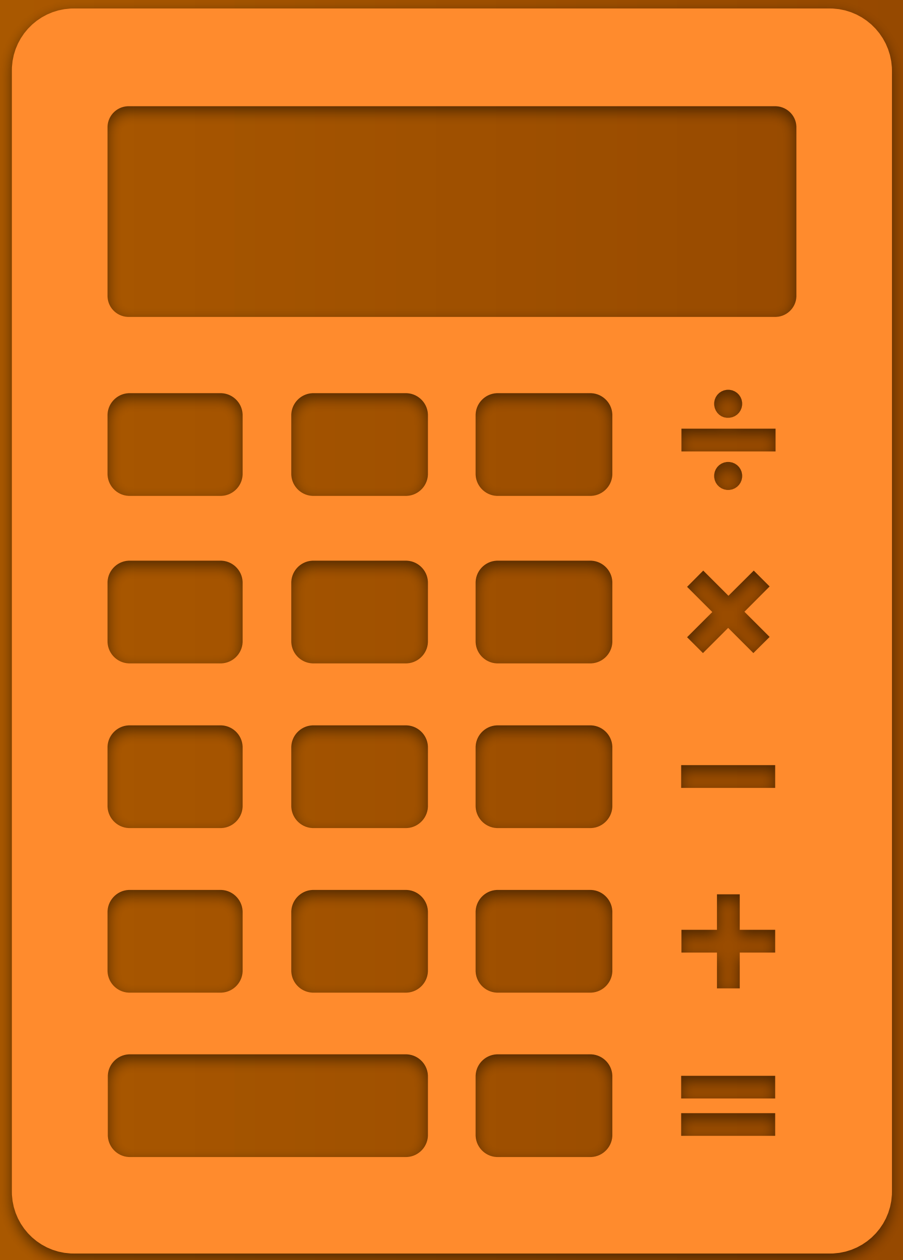
🦜🔗

- One of the fastest growing OSS projects of all time
- Python + JS
- Alternatively, many people just roll their own

Questions?

03

Tools



“Wait”, you might ask,

“Why do we need to make our own search engine?
Can’t we just use Google?”

The “I’m feeling lucky” chain

- Question ➡️ ***search google for it*** ➡️ get top result ➡️ pass top result to a summarization prompt ➡️ return summary

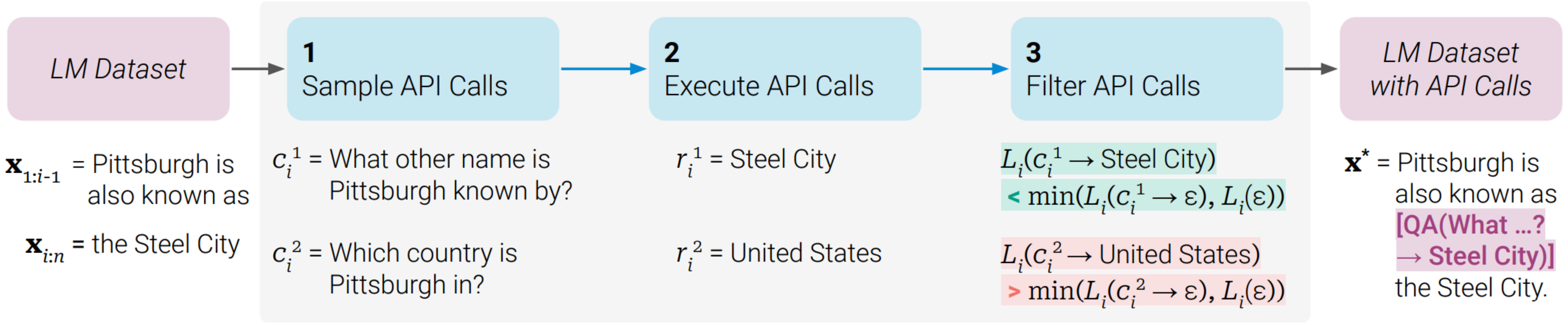
Insight:

Another way to give LLMs access to the outside world is to let them use *tools*

How far can we take this?

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick **Jane Dwivedi-Yu** **Roberto Dessì**[†] **Roberta Raileanu**
Maria Lomeli **Luke Zettlemoyer** **Nicola Cancedda** **Thomas Scialom**
Meta AI Research [†]Universitat Pompeu Fabra



$\mathbf{x}_{1:i-1}$ = Pittsburgh is also known as

$\mathbf{x}_{i:n}$ = the Steel City

c_i^1 = What other name is Pittsburgh known by?

c_i^2 = Which country is Pittsburgh in?

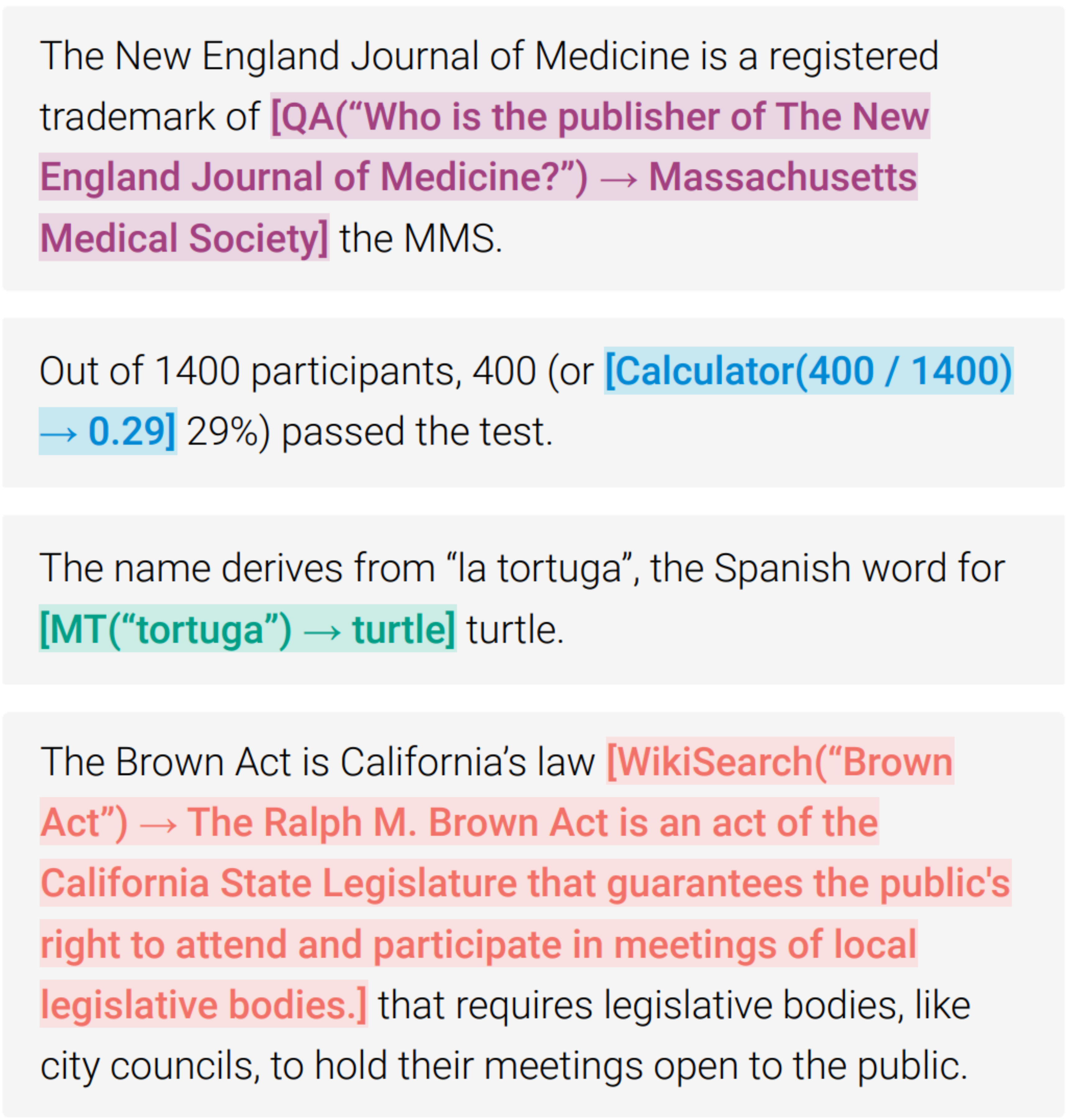
r_i^1 = Steel City

r_i^2 = United States

$L_i(c_i^1 \rightarrow \text{Steel City}) < \min(L_i(c_i^1 \rightarrow \varepsilon), L_i(\varepsilon))$

$L_i(c_i^2 \rightarrow \text{United States}) > \min(L_i(c_i^2 \rightarrow \varepsilon), L_i(\varepsilon))$

$\mathbf{x}^*$ = Pittsburgh is also known as [QA(What …? → Steel City)] the Steel City.



The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

- Only a few tools
- Dataset is self-generated

Tool use in langchain

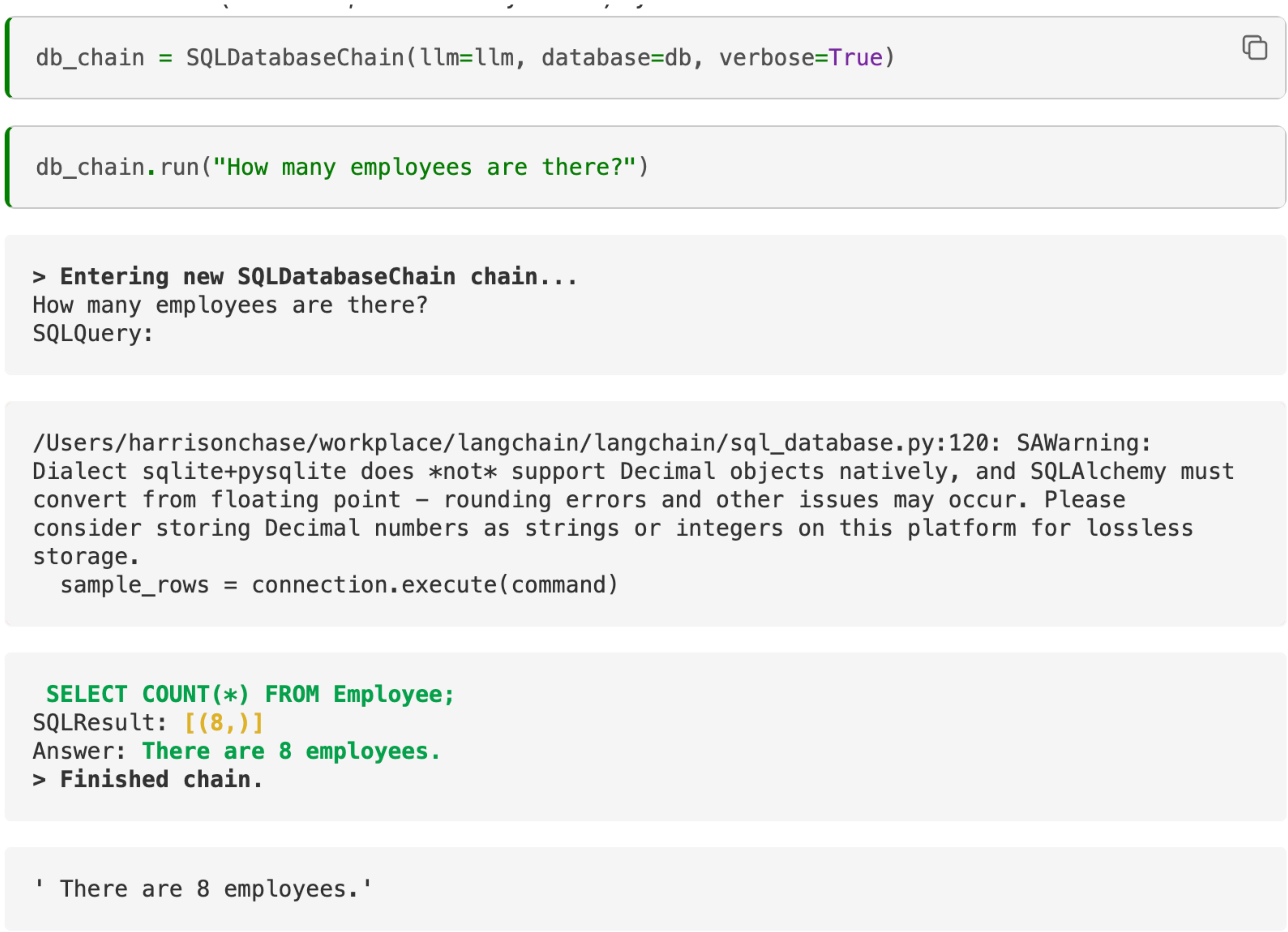
- Tools can be used like any other element in a chain
- They can also be provided to “agents” a-la toolformer or ChatGPT plugins

Example tools

- Arxiv
- Bash
- Bing search
- Google
- IFTTT
- Python
- Wikipedia
- Wolfram alpha
- Zapier

Example tool chain: querying sql

- User asks a question
- Put the question, with some some information about your db, into a prompt
- Run the prompt to get a SQL query
- Run the SQL query on the db
- Pass the result and the question to another LLM
- Get an answer back!



```
db_chain = SQLDatabaseChain(llm=llm, database=db, verbose=True)
```

```
db_chain.run("How many employees are there?")
```

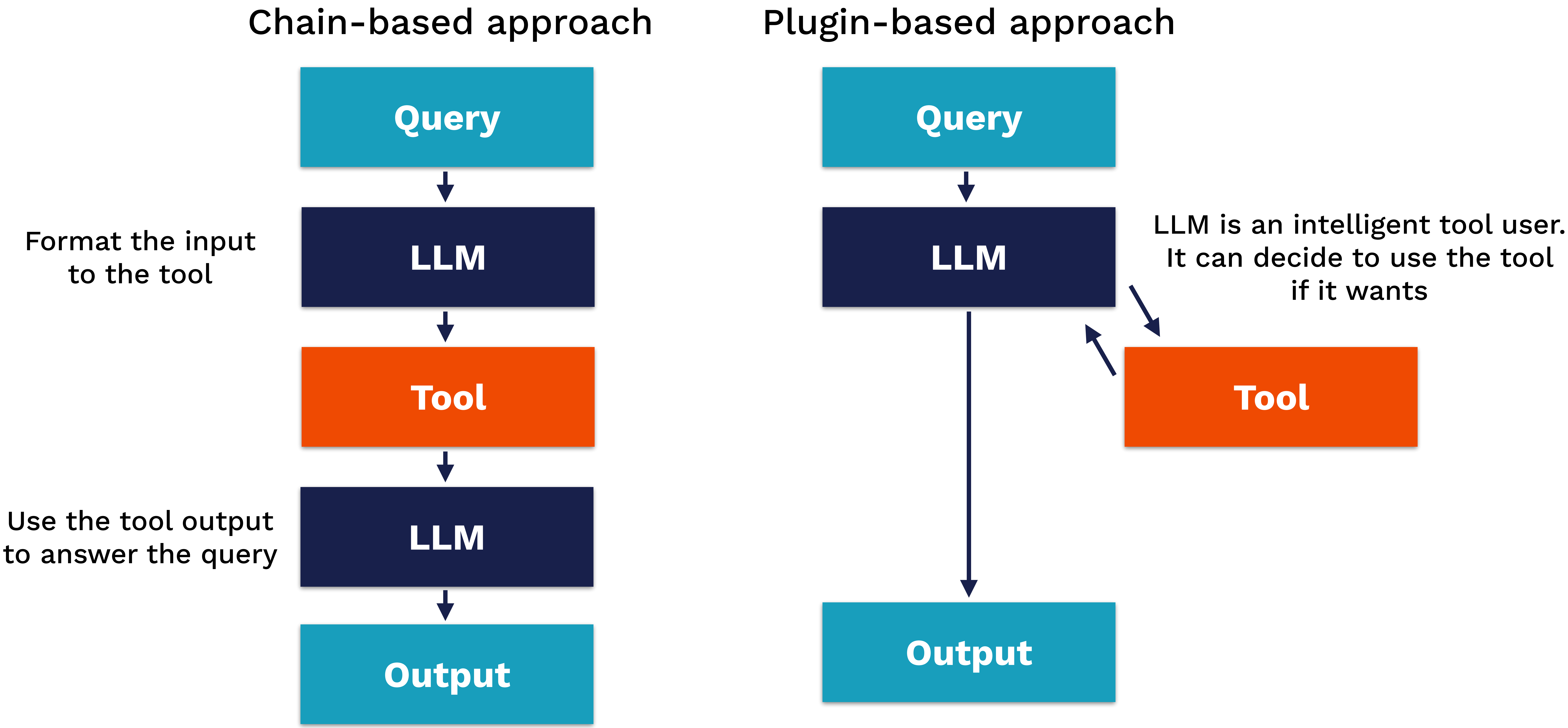
```
> Entering new SQLDatabaseChain chain...
How many employees are there?
SQLQuery:
```

```
/Users/harrisonchase/workplace/langchain/langchain/sql_database.py:120: SAWarning:
Dialect sqlite+pysqlite does *not* support Decimal objects natively, and SQLAlchemy must
convert from floating point - rounding errors and other issues may occur. Please
consider storing Decimal numbers as strings or integers on this platform for lossless
storage.
  sample_rows = connection.execute(command)
```

```
 SELECT COUNT(*) FROM Employee;
SQLResult: [(8,)]
Answer: There are 8 employees.
> Finished chain.
```

```
' There are 8 employees.'
```

A more automated approach: plugins



Chain-based approach

Query

LLM

Format the input to the tool

Tool

LLM

Use the tool output to answer the query

Output

Plugin-based approach

Query

LLM

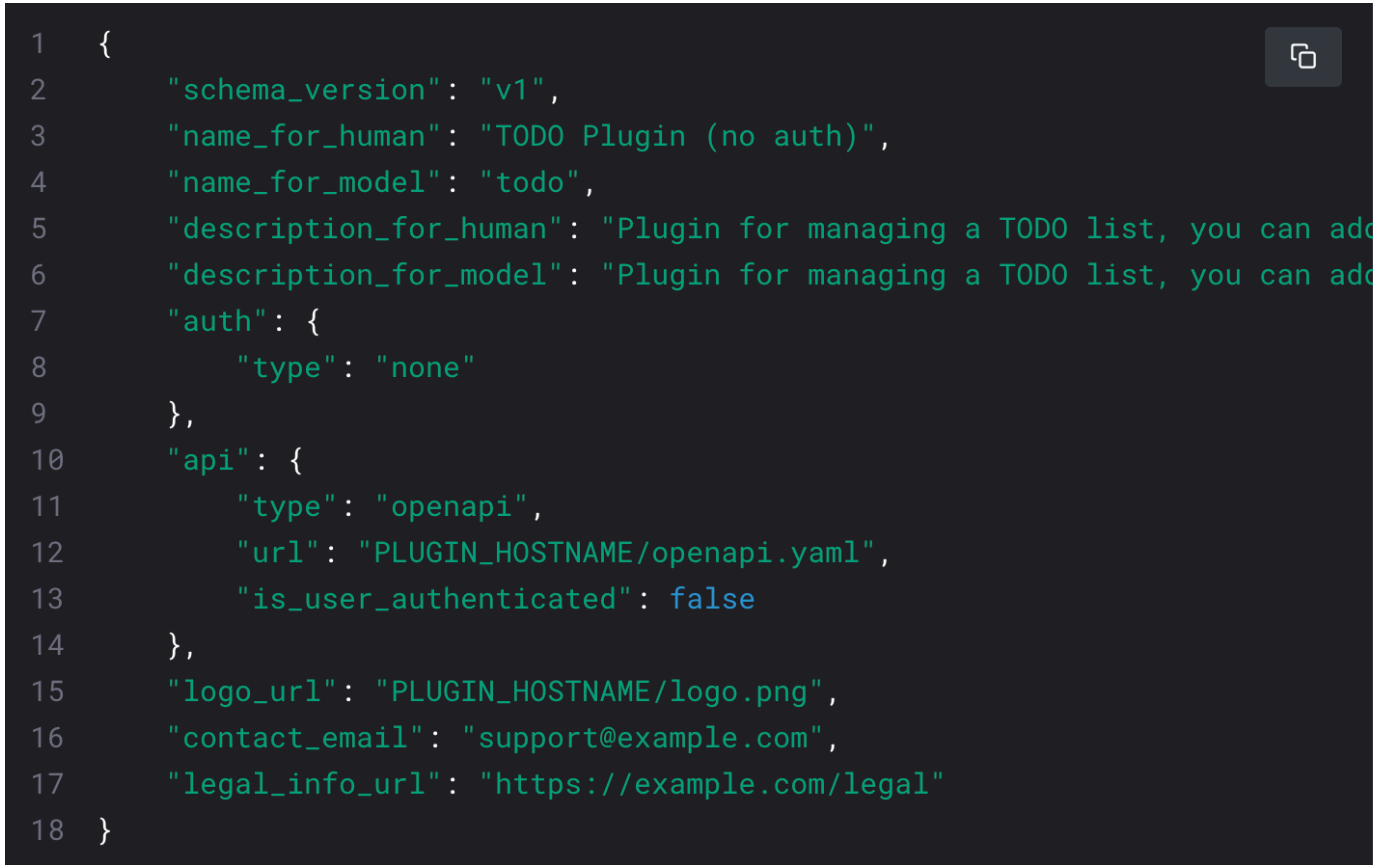
LLM is an intelligent tool user. It can decide to use the tool if it wants

Tool

Output

Building ChatGPT plugins

- Provide API spec and description so the model knows how to use it
- Description is passed in a system message to ChatGPT
- Model can choose to invoke the API and include the results in the response



```
{
    "schema_version": "v1",
    "name_for_human": "TODO Plugin (no auth)",
    "name_for_model": "todo",
    "description_for_human": "Plugin for managing a TODO list, you can ad
    "description_for_model": "Plugin for managing a TODO list, you can ad
    "auth": {
        "type": "none"
    },
    "api": {
        "type": "openapi",
        "url": "PLUGIN_HOSTNAME/openapi.yaml",
        "is_user_authenticated": false
    },
    "logo_url": "PLUGIN_HOSTNAME/logo.png",
    "contact_email": "support@example.com",
    "legal_info_url": "https://example.com/legal"
}
```

Tool use: recommendations

- Tools are a **more flexible way** to give your model access to external data: retrieval is just one tool
- If you want interactivity / flexibility, go with **plugins**
- If you want reliability, start with **chains**

Questions?

Conclusion

- LLMs are **more powerful when connected to external data**
- You can do a lot with **rules and heuristics**
- As your knowledge base scales you should start thinking about it as an **information retrieval system**
- Chains can help you **encode more complex reasoning** and get around token limits
- Tools give your model access to a much **broader body of knowledge**

Where to go to learn more?

- Augmented Language Models: a Survey (Mialon et al, 2023): https://arxiv.org/abs/2302.07842
- A great course about information retrieval: https://github.com/sebastian-hofstaetter/teaching
- Langchain documentation: https://python.langchain.com/en/latest/index.html

Thank you!