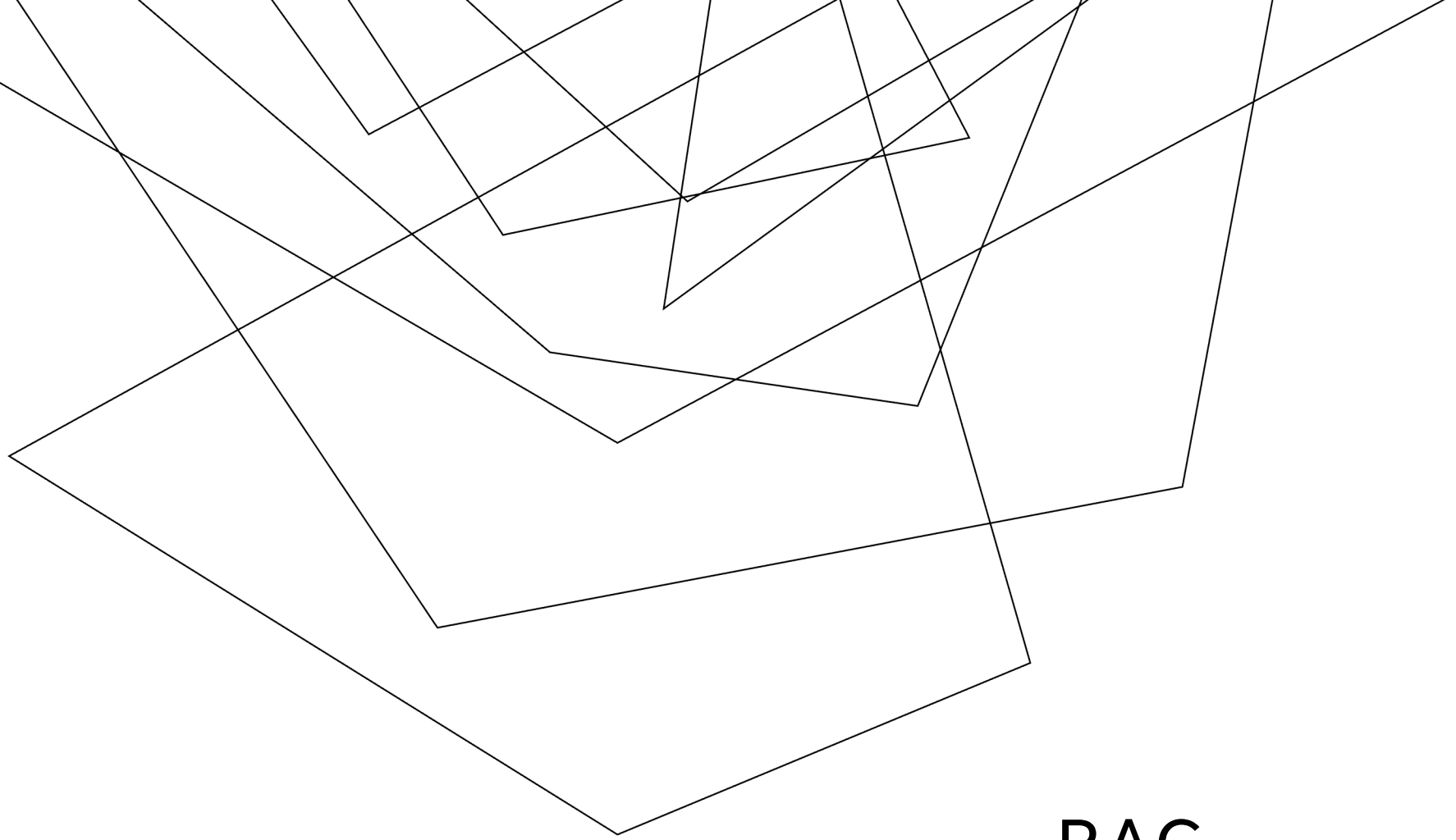
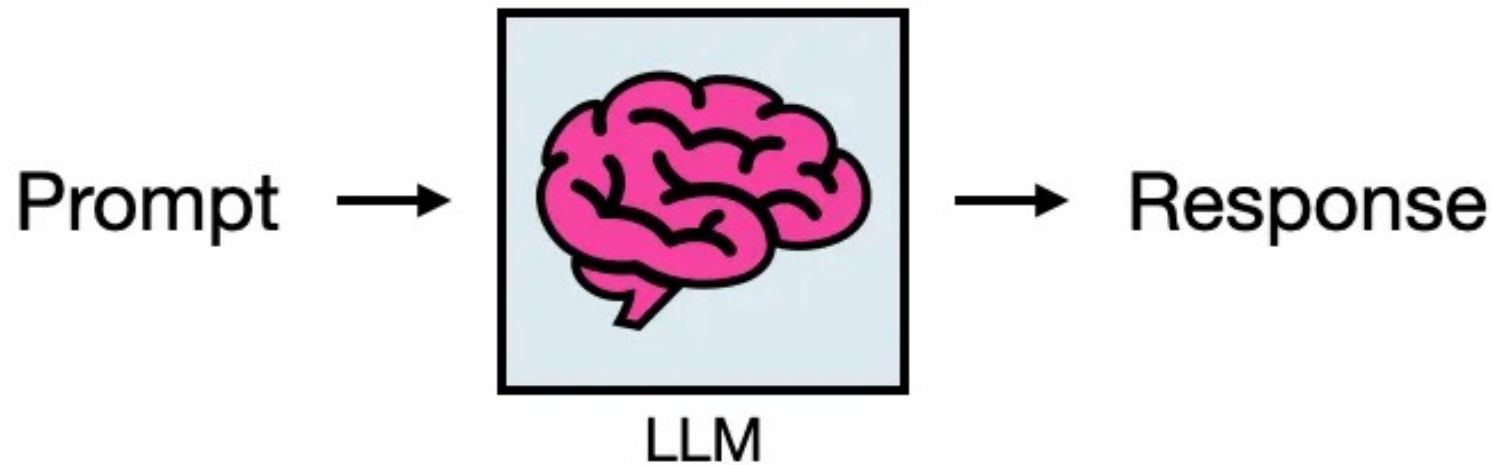
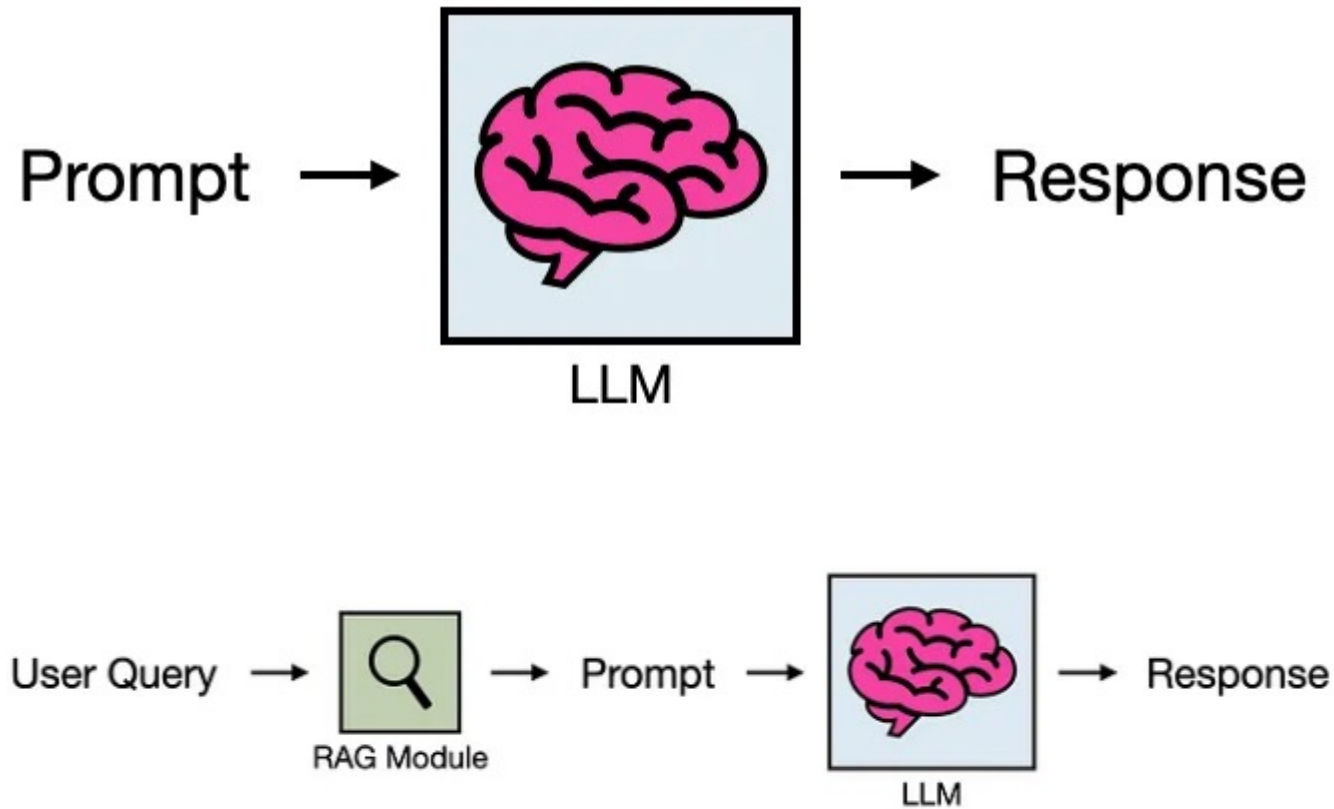


RAG

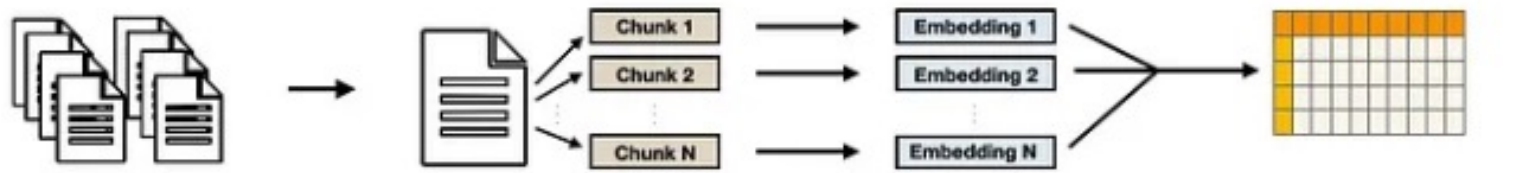
What is RAG?

The basic usage of an LLM consists of giving it a prompt and getting back a response.





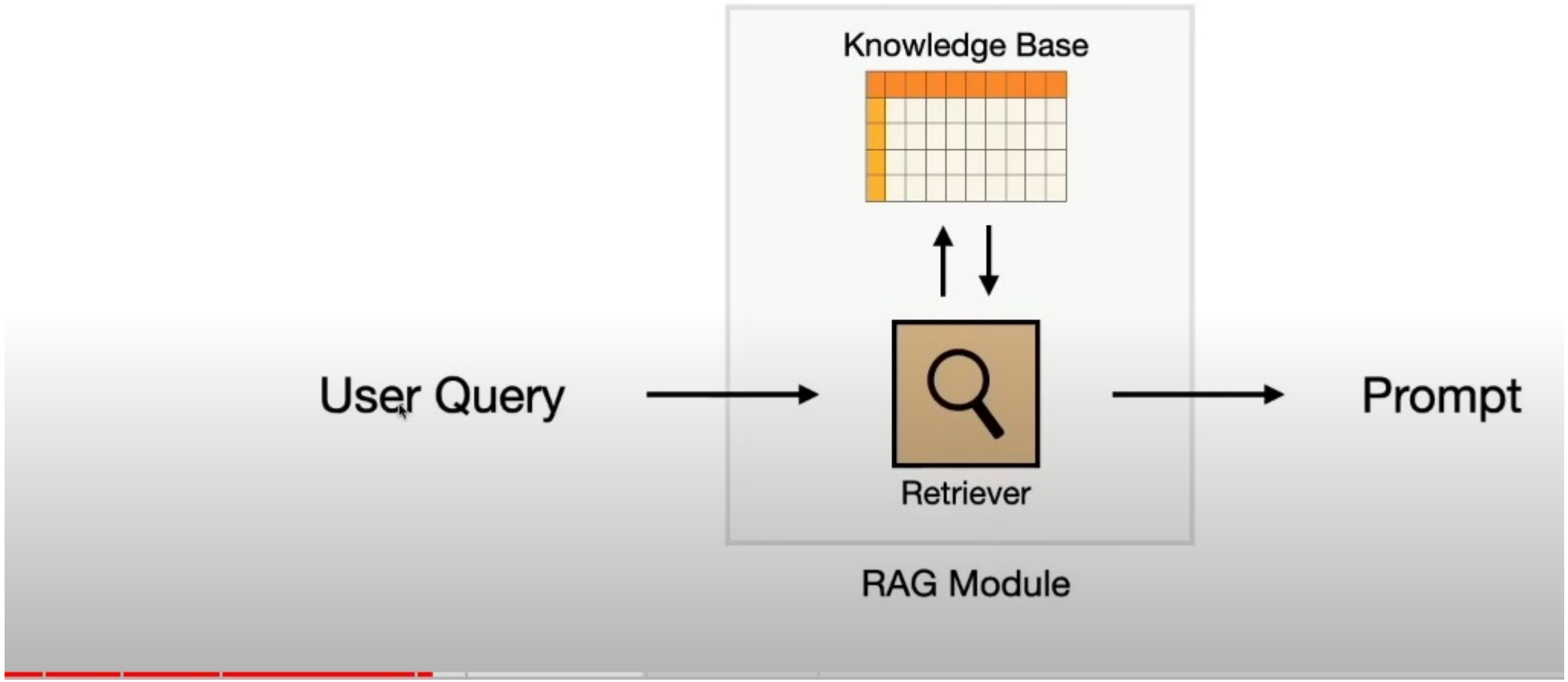
Overview of RAG system. Image by author.

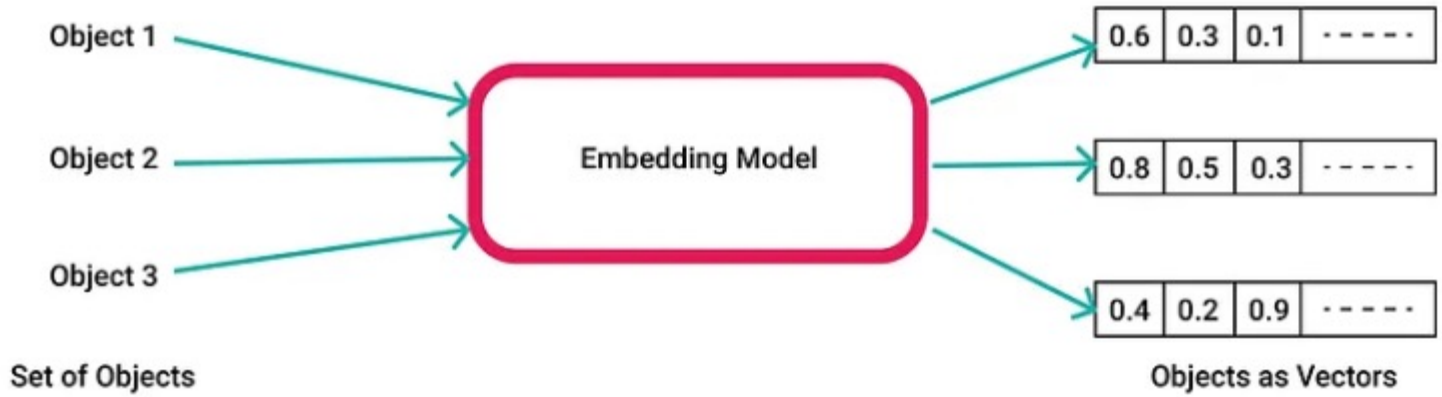


1) Load Docs 2) Chunk Docs 3) Embed Chunks 4) Load into VDB

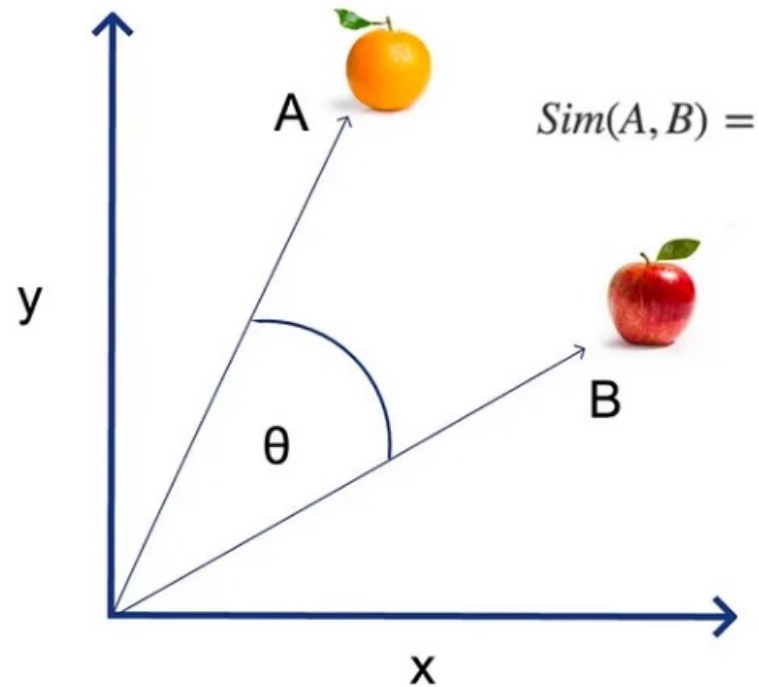
How it works?

2 key elements: retriever and knowledge base





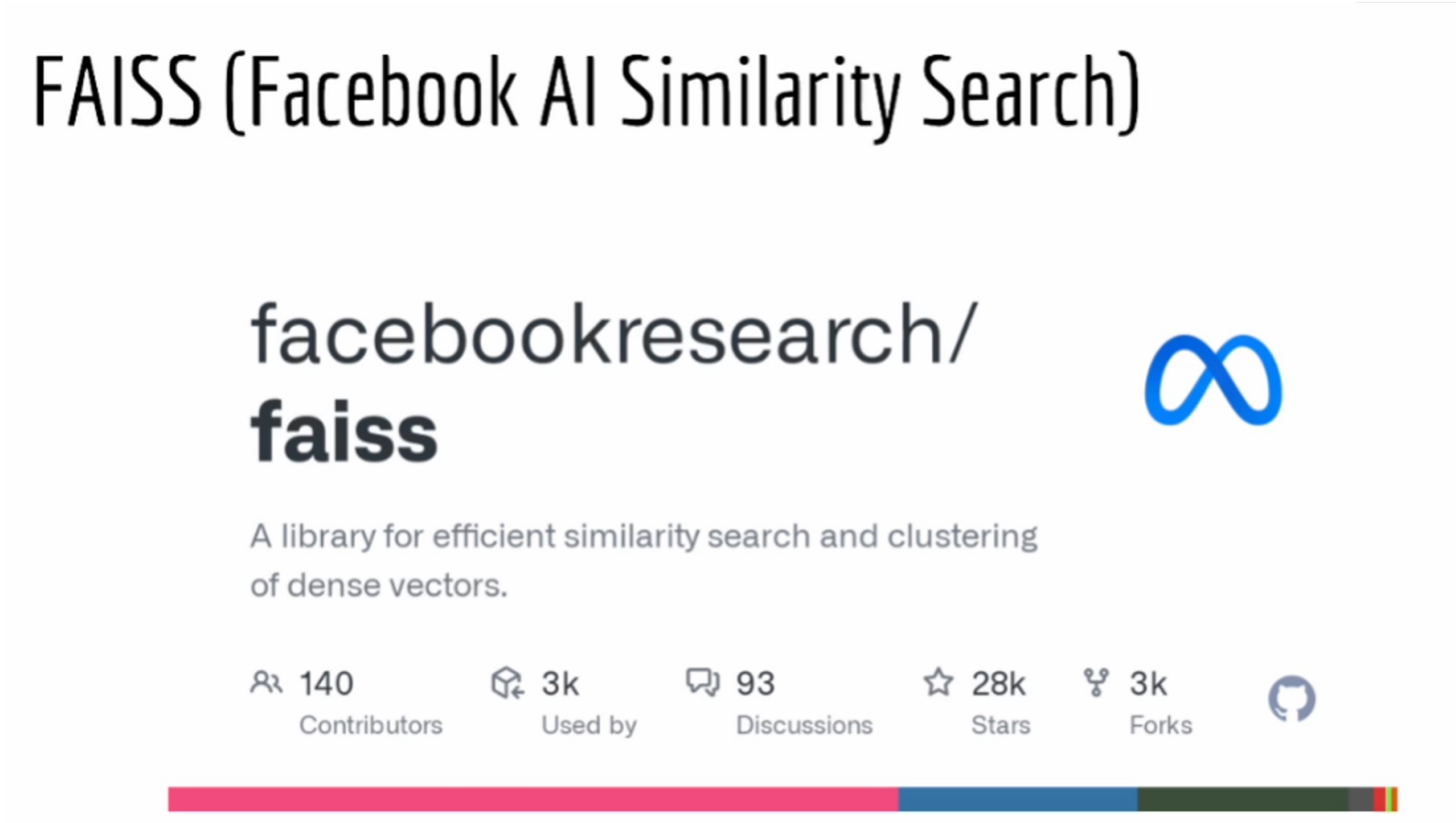
vector embedding



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

Cosine similarity

FAISS (Facebook AI Similarity Search)



Installation

- `pip install faiss-cpu` (I'll use this)
- `pip install faiss-gpu` (requires NVIDIA GPU)

Create Index

- Also possible to use "inner product" distance (**`IndexFlatIP`**)
- Inner product is "similar" to cosine distance, since $\cos\theta = \langle a, b \rangle / \|a\|\|b\|$
- See FAISS Github repo for discussion about why it wasn't included
- When vectors are normalized, using L2 is equivalent to IP
- Exercise: prove it mathematically (we did it in NLP course)

```
import faiss
index = faiss.IndexFlatL2(D)
```

Add Vectors to Index

- Must be 2-D (N x D, where N=number of vectors, D=vector dimensionality)
- Must have a "shape" parameter (e.g. Numpy array)
- E.g. list of lists won't work, even though for SKLearn / TF it's fine

```
index.add(vectors) # vectors is NxD
```

Query Your Index

- Query vector must also be N x D, even if N=1
- Indices (return value) tells you where in your *original* vectors the closest match is found (so keep track of this!)
- Distances and indices are sorted in corresponding order (closest-furthest)

```
distances, indices = index.search(query_vec, k=5)
distances.shape # N x k
indices.shape # N x k
```