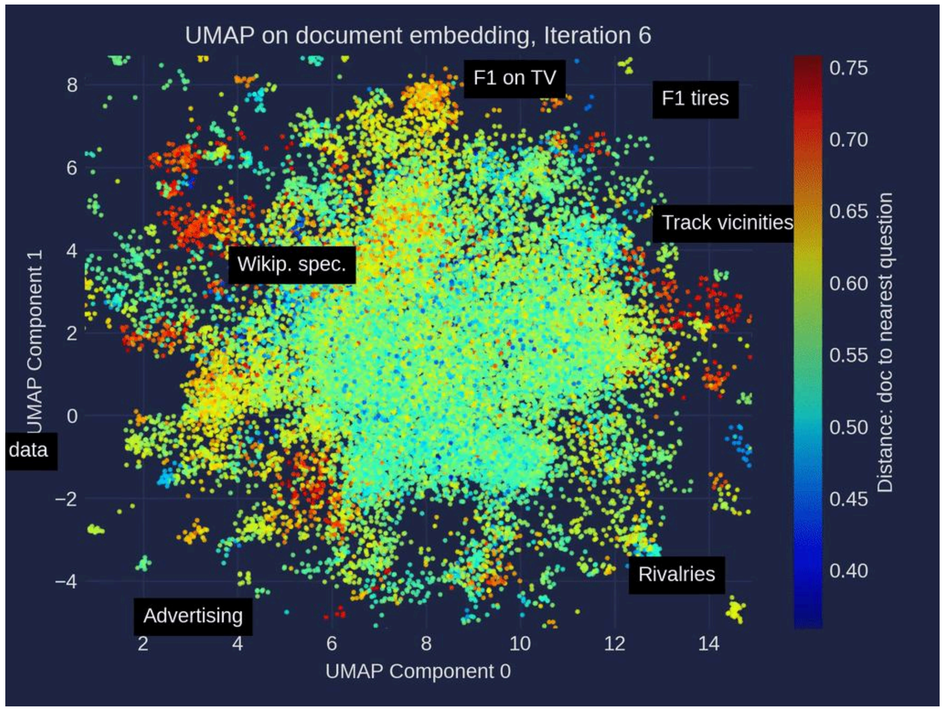
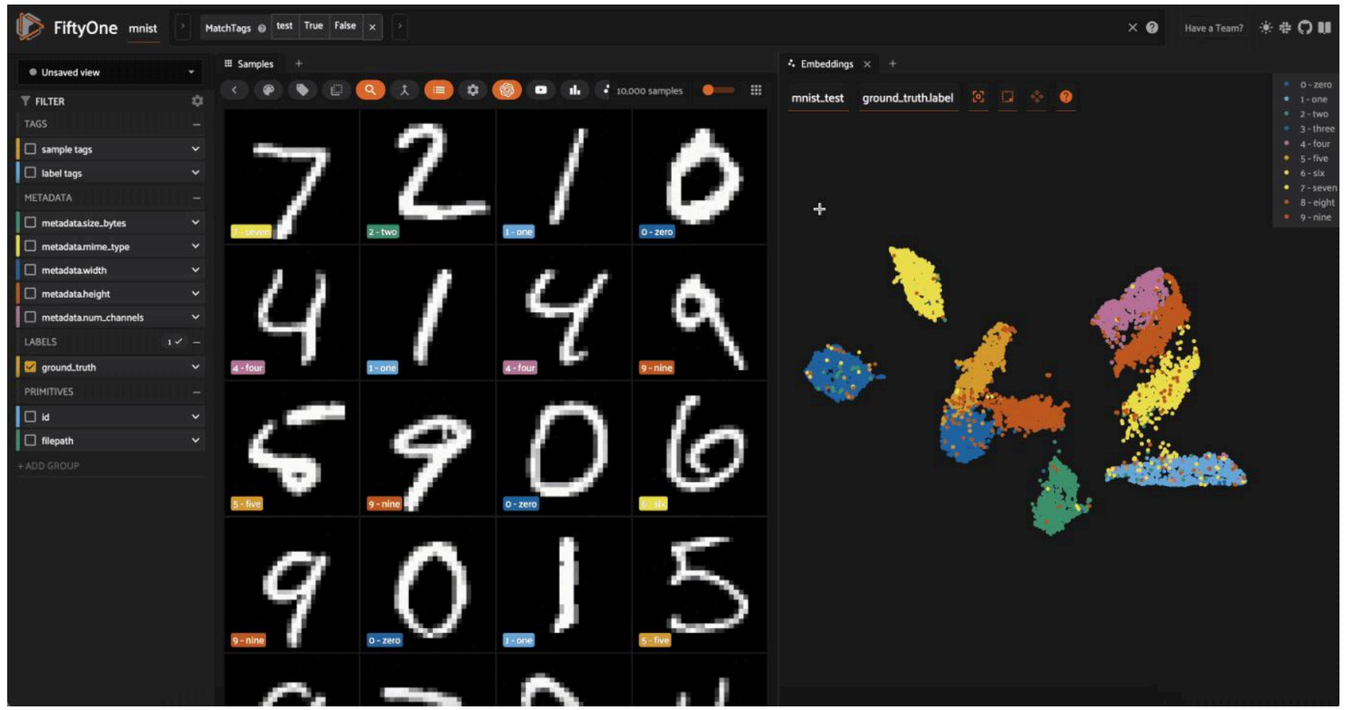
understanding embeddings



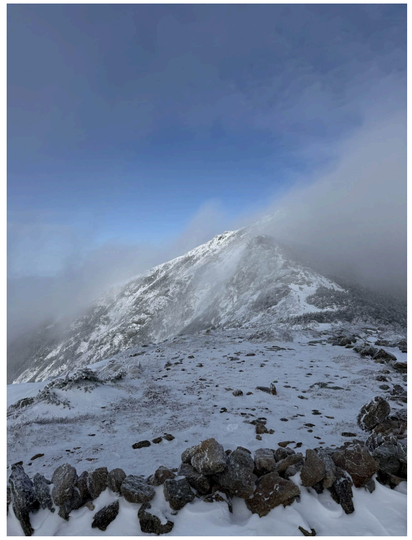


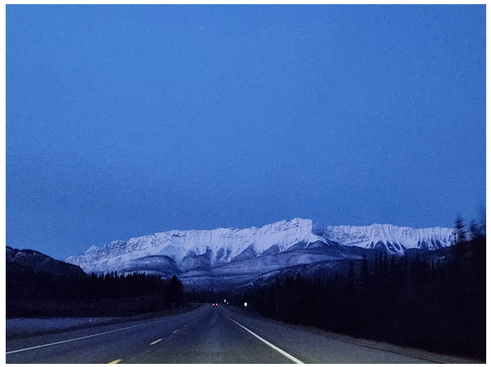
intro week 1: february 11, 2026 | rutgers ieee ml/ai

natural language processing track by mehek

hullo i am mehek

- i am from florida
- i lost sleep at too many hackathons last year
- i worked at a startup for a few months this fall
- i like explainable ai research
- sometimes i run
- sometimes i hike / scuba dive
- sometimes i crash out

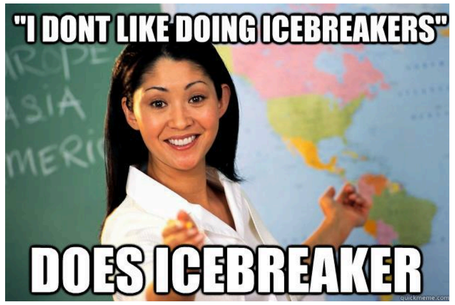




omg yay icebreaker

pls say your

- name
- major
- what you wanna do after undergrad (phd/industry/startup)
- your favorite item that you bought this year



me coming up with this slide

intro to nlp

what is nlp? (natural language processing)

enables computers to interpret, analyze, generate, and understand human language (both text and speech)

what is nlp used for?

- virtual assistants / chatbots
- sentiment analysis
- spam detection & content moderation
- topic modeling (categorization)

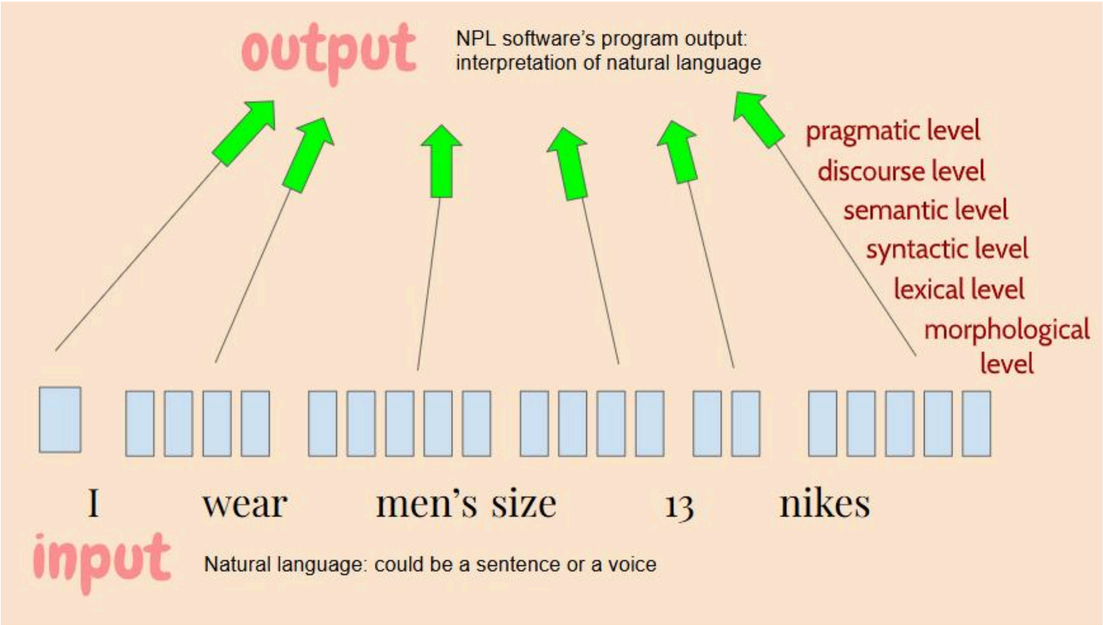
nlp algorithms / models / etc

rule based models

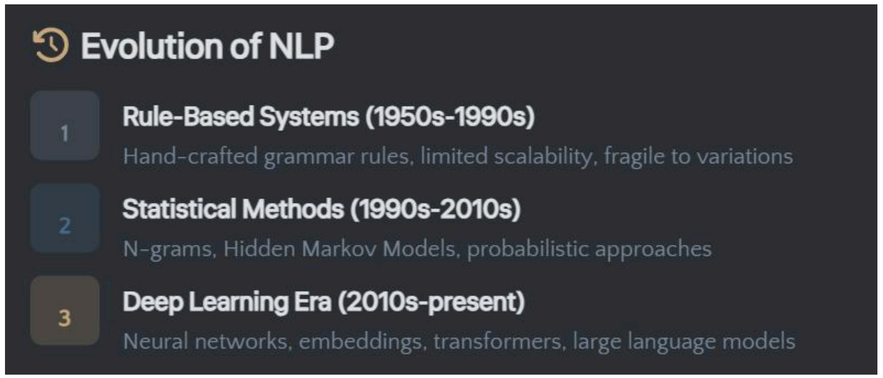
- naive bayes, hidden markov models, support vector machines

deep learning models

- rnns, lstms, transformers (bert/gpt)



these are more classical approaches, but are still used because of computational efficiency / interpretability



data preprocessing

with more classical approaches:

- lowercasing
- removing punctuation
- removing stop words (ex: the, is, at, and)
- removing numbers, special characters
- text normalization (decoding contractions, handling slang)

with transformers (bert/gpt):

- tokenization

Bag of Words (BoW)

simple word counts

TF-IDF

weights words by importance across documents

Embeddings

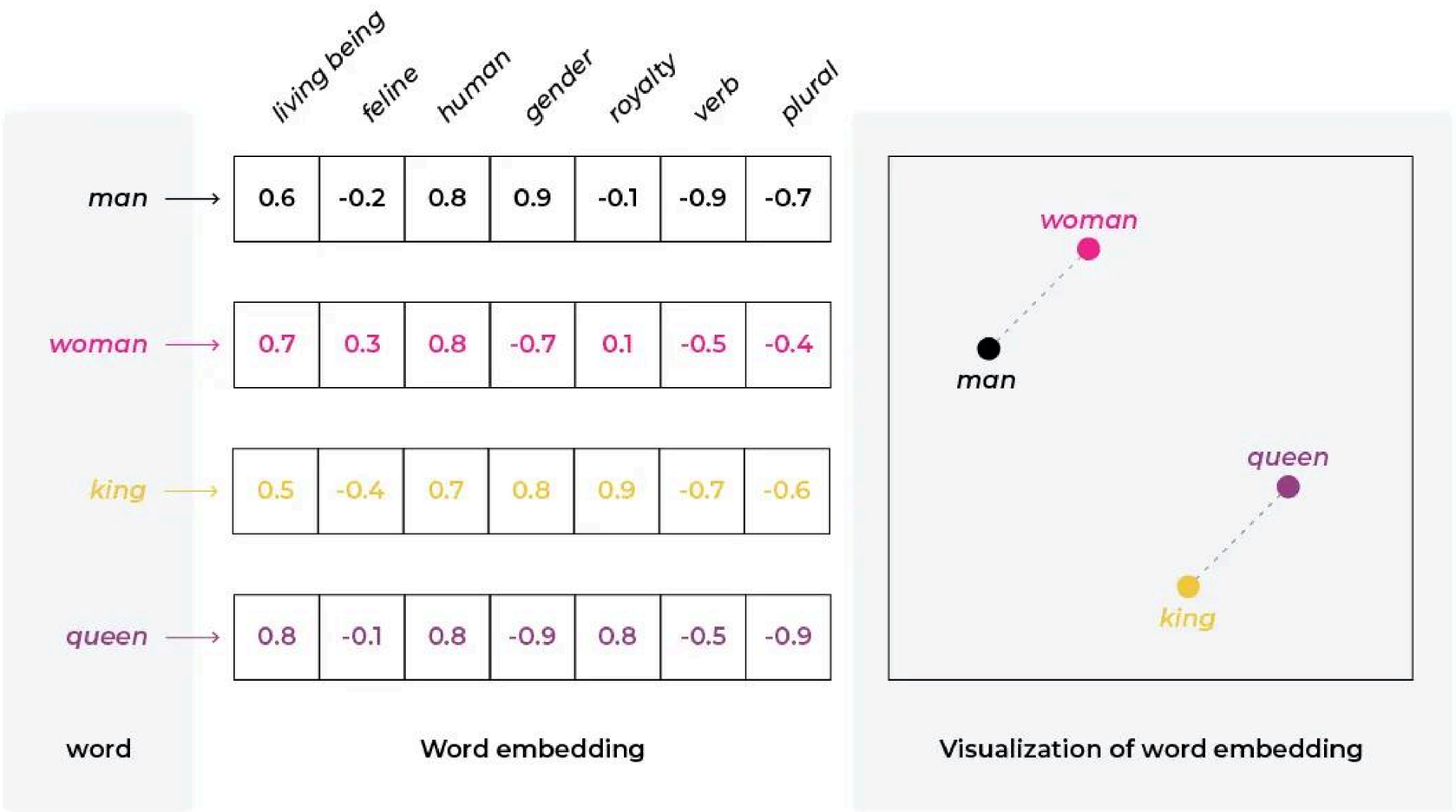
convert words into dense vectors

turning text into numbers: embeddings

what is an embedding?

a dense vector representation of a word, sentence, or document

- similar words ⟶ similar vectors
- capture semantic meaning



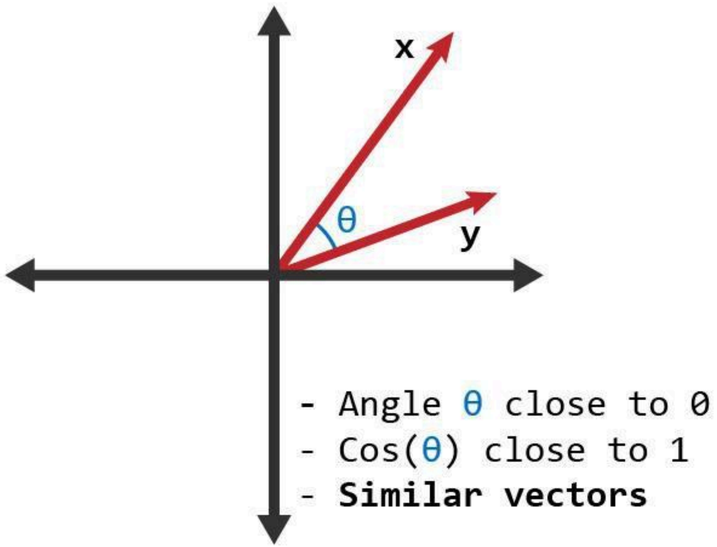
vectors, colors, and cosine similarity

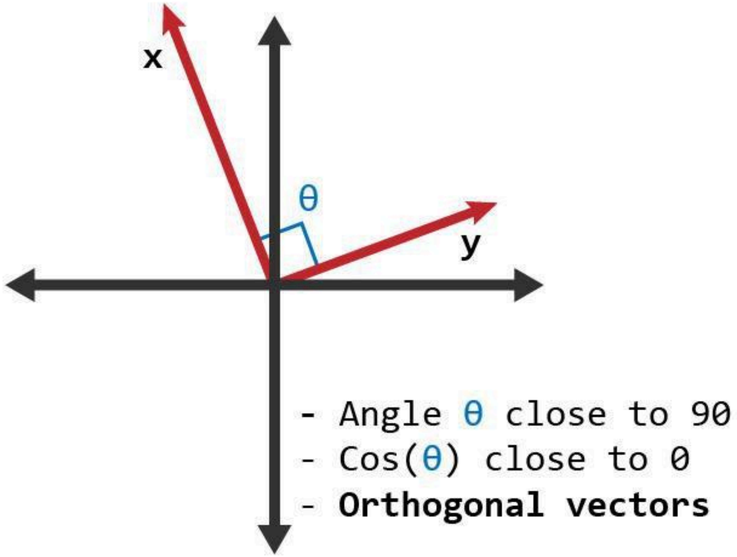
RGB as an [embedding analogy]

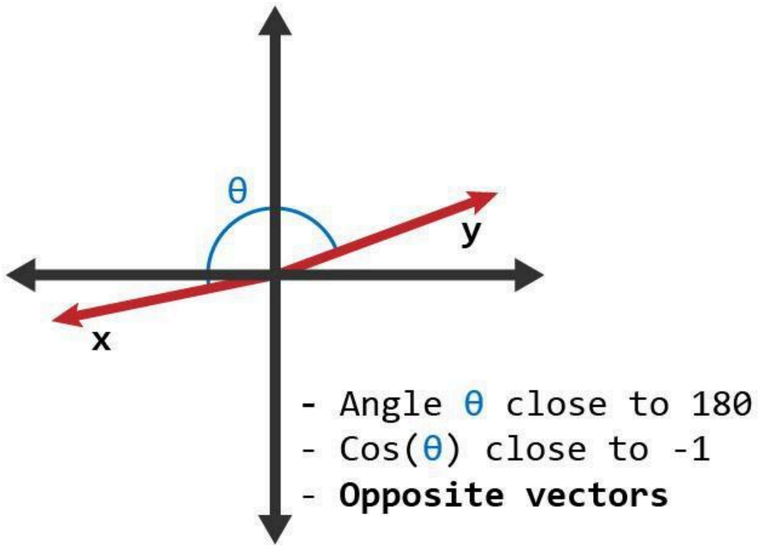
color as RGB is a 3d vector of (red, green, blue)
red = (255, 0, 0)
dark = (200, 0, 0)

how do we measure similarity?

euclidean distance (considers magnitude)
dot product (considers magnitude)
cosine similarity (considers only direction)





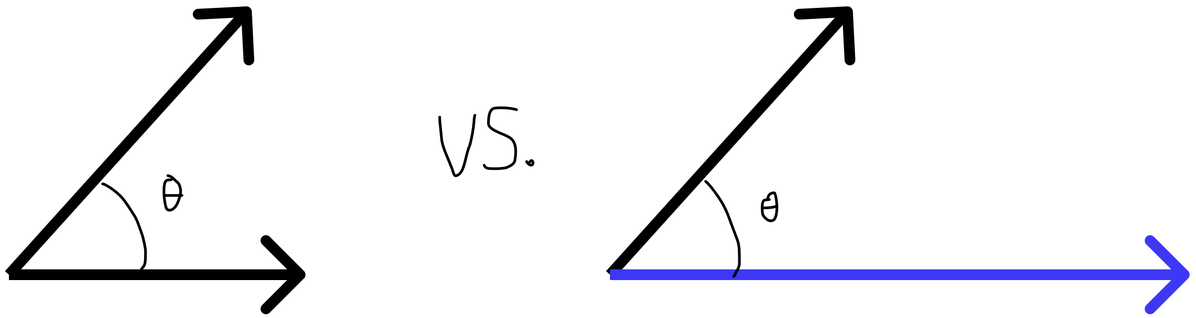


cosine similarity (cont.)

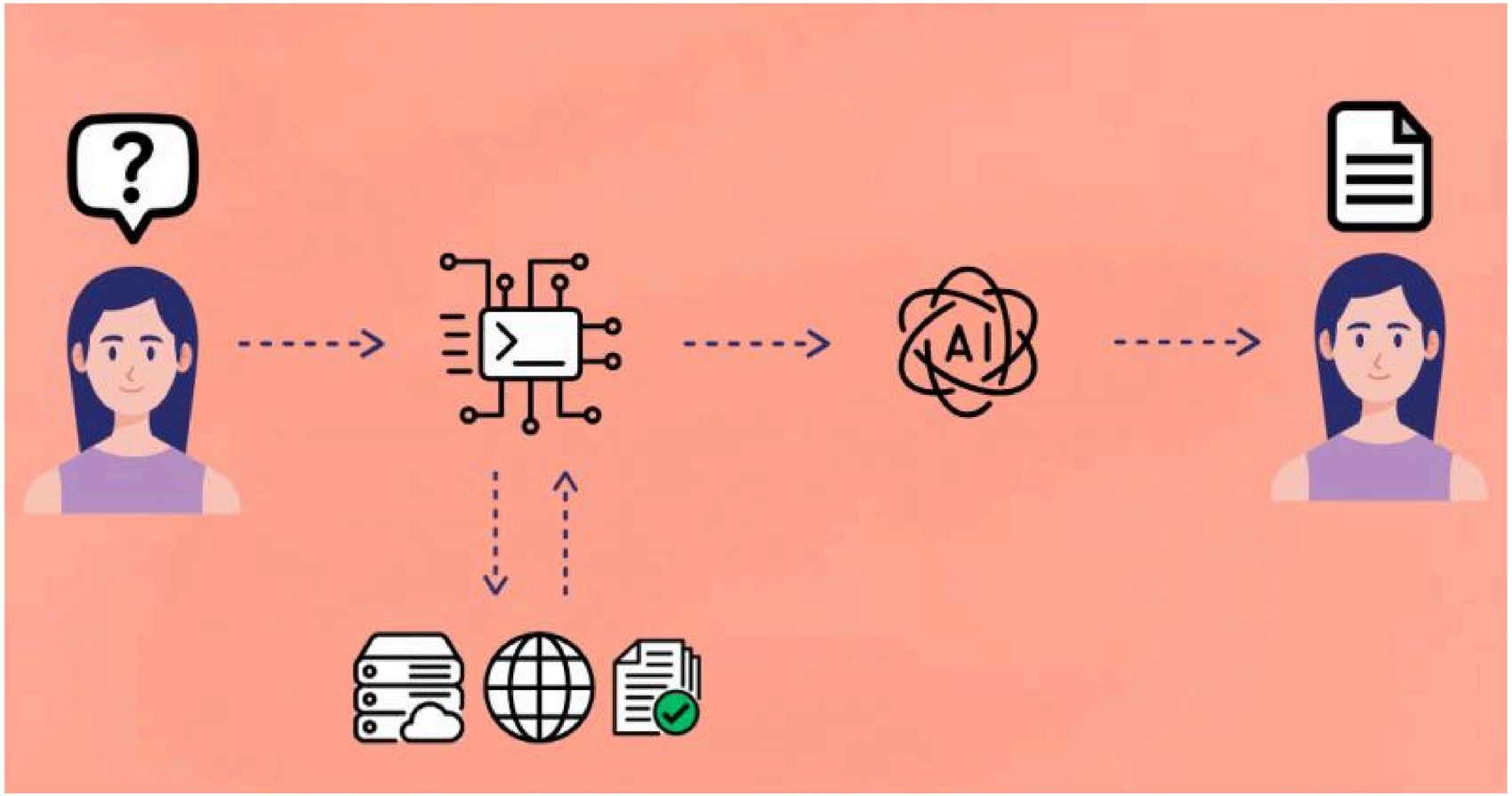
cosine similarity is the dot product divided by the magnitudes of the two vectors

- used for semantic search
- works well for high dimensions
- it is a normalized dot product
- says ⟶ length doesn't matter, semantic direction matters
- if the angle is the same, but lengths are different... ?
- if angle is same and lengths are the same... ?

$$\text{cosine_sim}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$



retrieval augmented generation



document
embed
chunk
index / store

→ **vector database**

query
embed query
semantic search
retrieve documents
generate

→ **final response**

another nice rag diagram

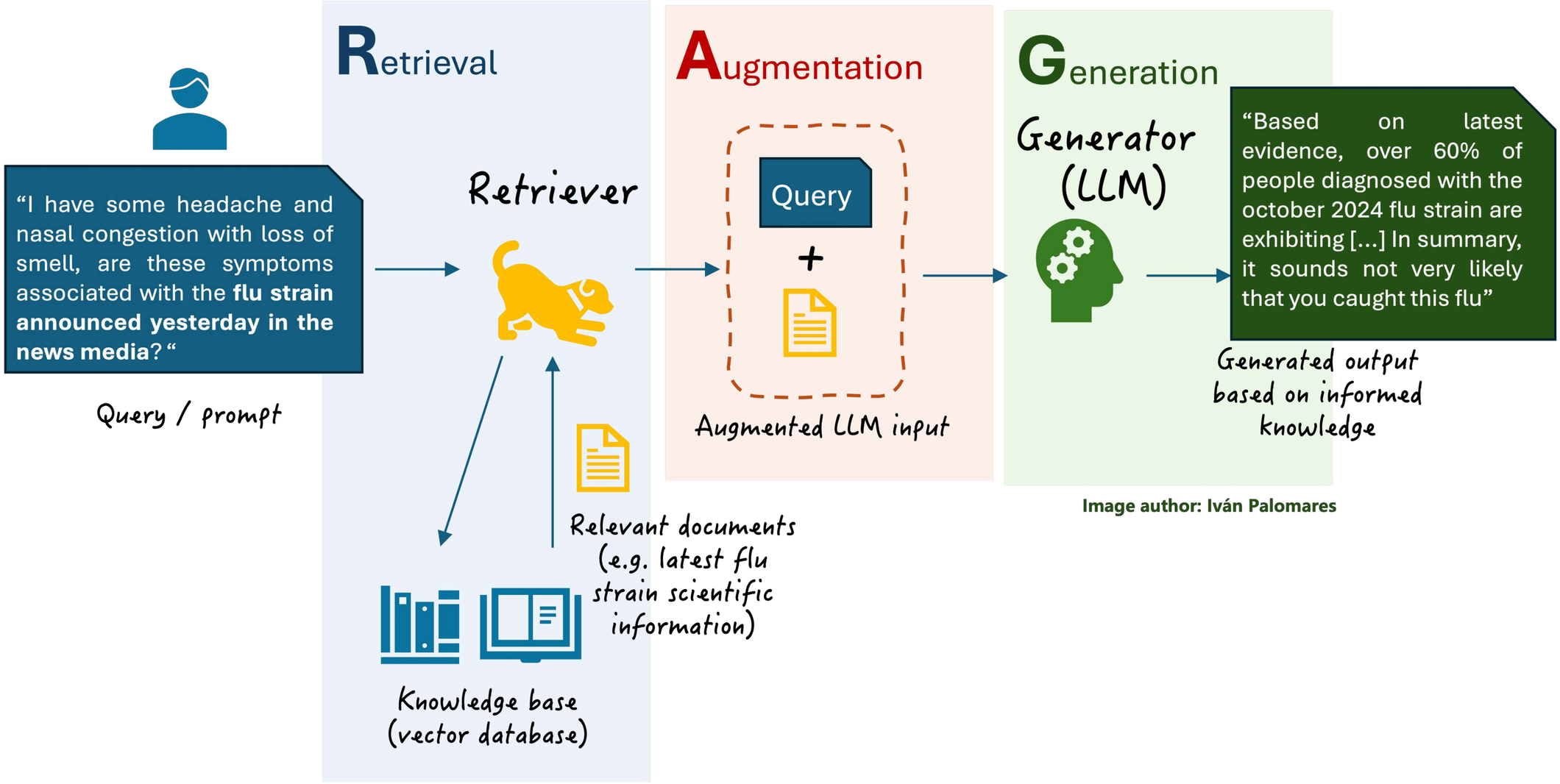


Image author: Iván Palomares

simple rag demo

google colab notebook

openai api key WILL NOT work after the workshop!

some other things you can try with the demo:

- changing chunking size
- use cosine similarity search by normalizing vectors to unit length and using faiss.IndexFlatIP(d)
- look at the other chunks and compare with what you think should be retrieved with a query

chunk pdf into pieces

```python
def chunk_text(text, chunk_size=500, overlap=100):
  # try changing the chunk_size !
    chunks = []
    start = 0
    while start < len(text):
        end = start + chunk_size
        chunk = text[start:end]
        chunks.append(chunk)
        start += chunk_size - overlap
    return chunks

chunks = chunk_text(text)
len(chunks)
```

```
100
```

simple rag demo (cont.)

google colab notebook

limitations of rag

- retrieval can return wrong or irrelevant chunks
- LLM hallucination
- chunking problems
    - too big: exceeds context window
    - too small: missing meaning, weak retrieval
    - PDF/HTML docs often chunk badly
- similarity search can match wrong concepts
- embeddings drift if documents change
- domain-specific embeddings sometimes needed
- context window limits