Practical Large Language Models: Using LLMs in Your Business With Python

Jonathan Mugan
@jmugan

Data Day Texas
January 27th, 2024

DeUmbra

Slides at https://www.jonathanmugan.com/WritingAndPress/

ChatGPT can teach you the nature of the universe

ChatGPT allows you to have a conversation with a textbook, which it also writes.

If you have a question, you only need to ask.

It got me over the conceptual hump that General Relativity merges space and time.

This is real. When you have a large mass, you don't "fall" by force. Instead, it curves space-time so your path through space-time, not just through space, takes you to the object.

It also drew me this picture.

But let's talk about something important

- Flexible search over your documents
- Question answering over your documents
- Exact calculations and decisions
- Considerations when using LLMs
- How LLMs Work

But let's talk about something important

- Flexible search over your documents
- Question answering over your documents
- Exact calculations and decisions
- Considerations when using LLMs
- How LLMs Work

Flexible document search

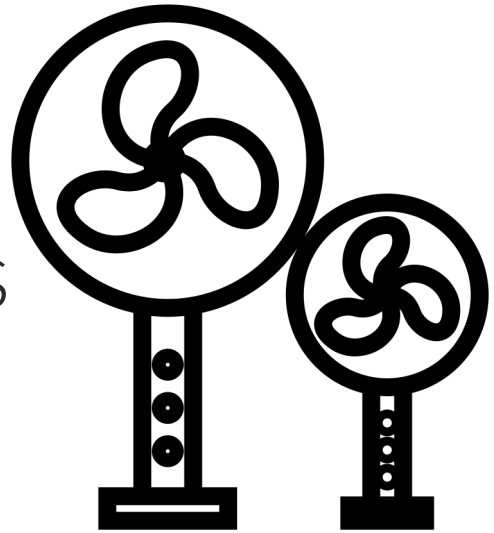
Vectors are great for search because you can't always think of the right keyword.

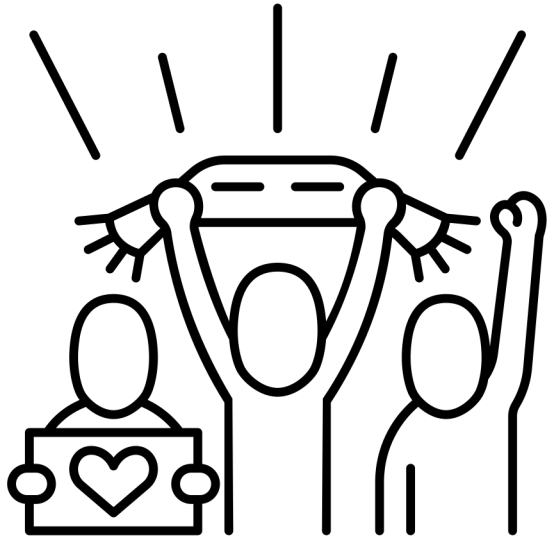
Also, as Led Zeppelin taught us, words can have more than one meaning.

Search: fans

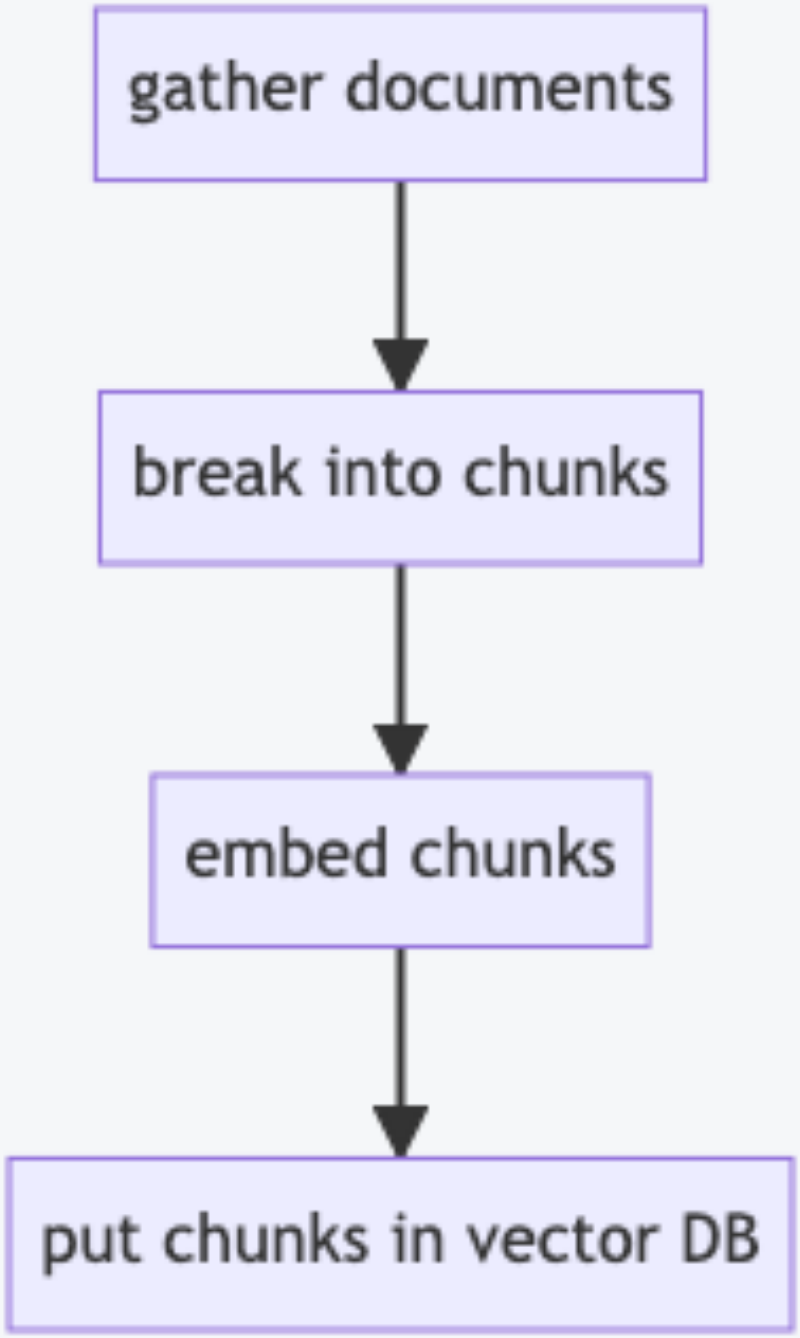
Search: sports fans

(will return docs about the Brewers with vector search)





Called "semantic search." Often trained by making embeddings of sentences closer together if they are related and farther apart if they are not.



Steps to prepare your documents

- textract if pdf https://textract.readthedocs.io/en/stable/python_package.html

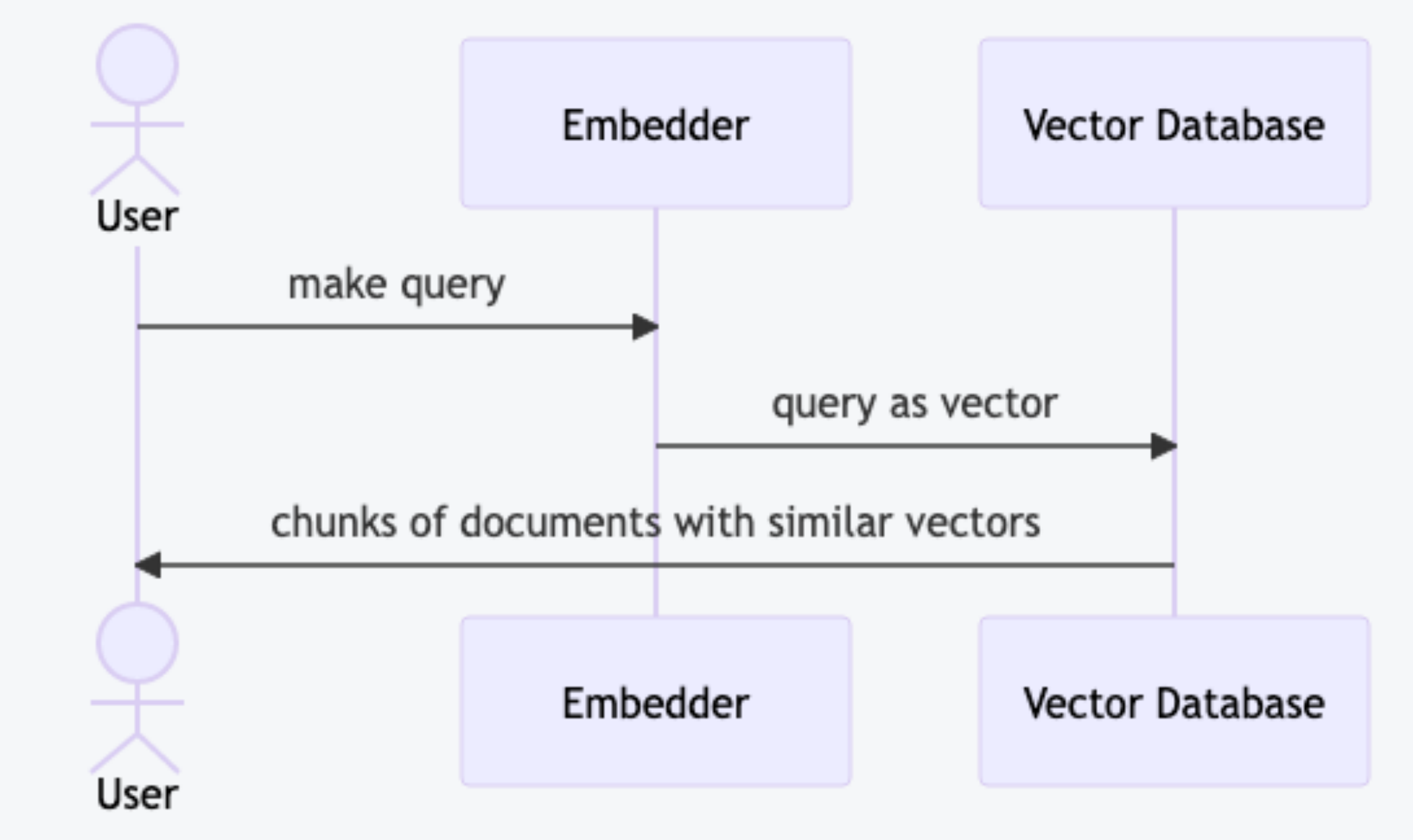
- LlamaIndex https://www.llamaindex.ai/
- LangChain https://www.langchain.com/

- Sentence-Transformers https://www.sbert.net/
- OpenAI https://platform.openai.com/docs/guides/embeddings
- TensorFlow Universal Sentence Encoder https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder

- Milvus https://milvus.io/
- Pinecone https://www.pinecone.io/
- Azure AI Search (formally Cognitive Search) https://learn.microsoft.com/en-us/azure/search/
- Memory with FAISS https://github.com/facebookresearch/faiss or scikit-learn https://scikit-learn.org/stable/

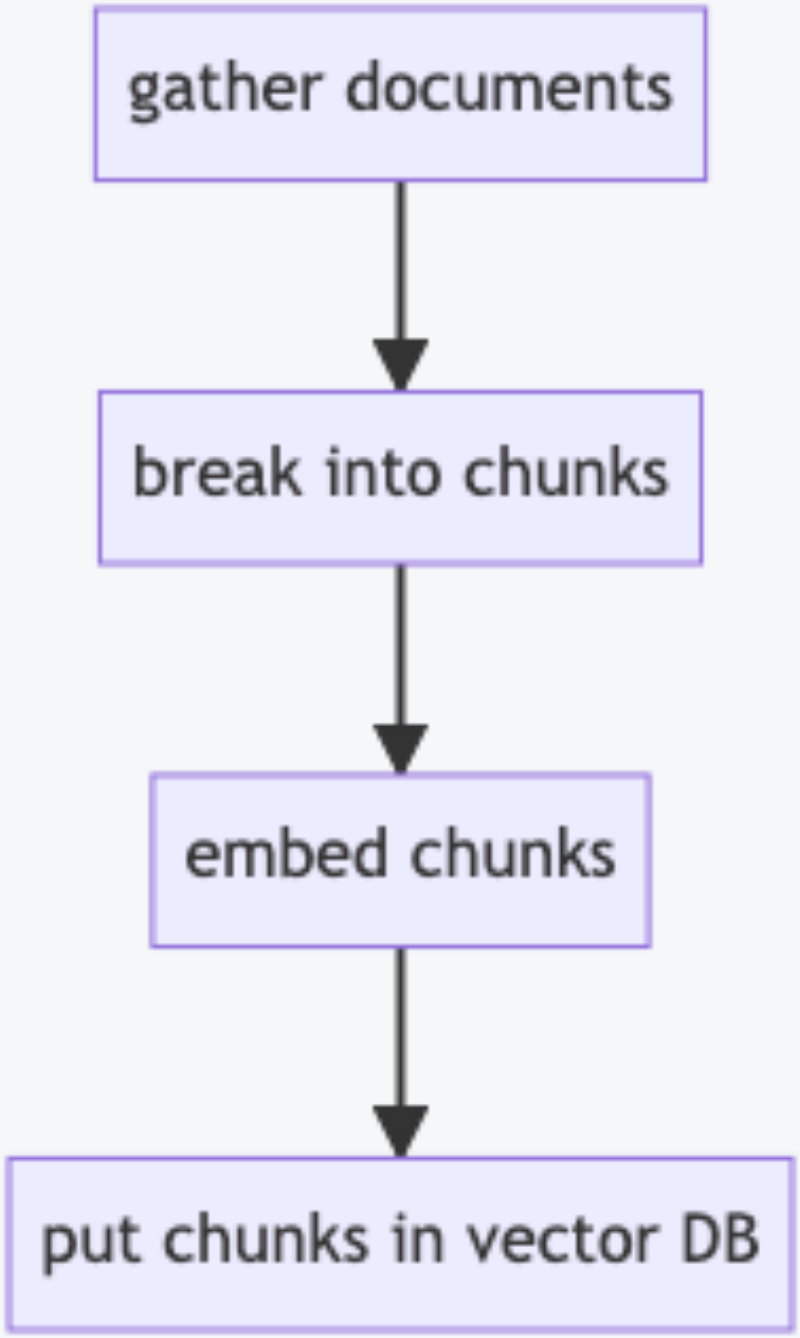
Search documents process

Can also rerank with an LLM and use hybrid search



Outline

- Flexible search over your documents
- Question answering over your documents
- Exact calculations and decisions
- Considerations when using LLMs
- How LLMs Work



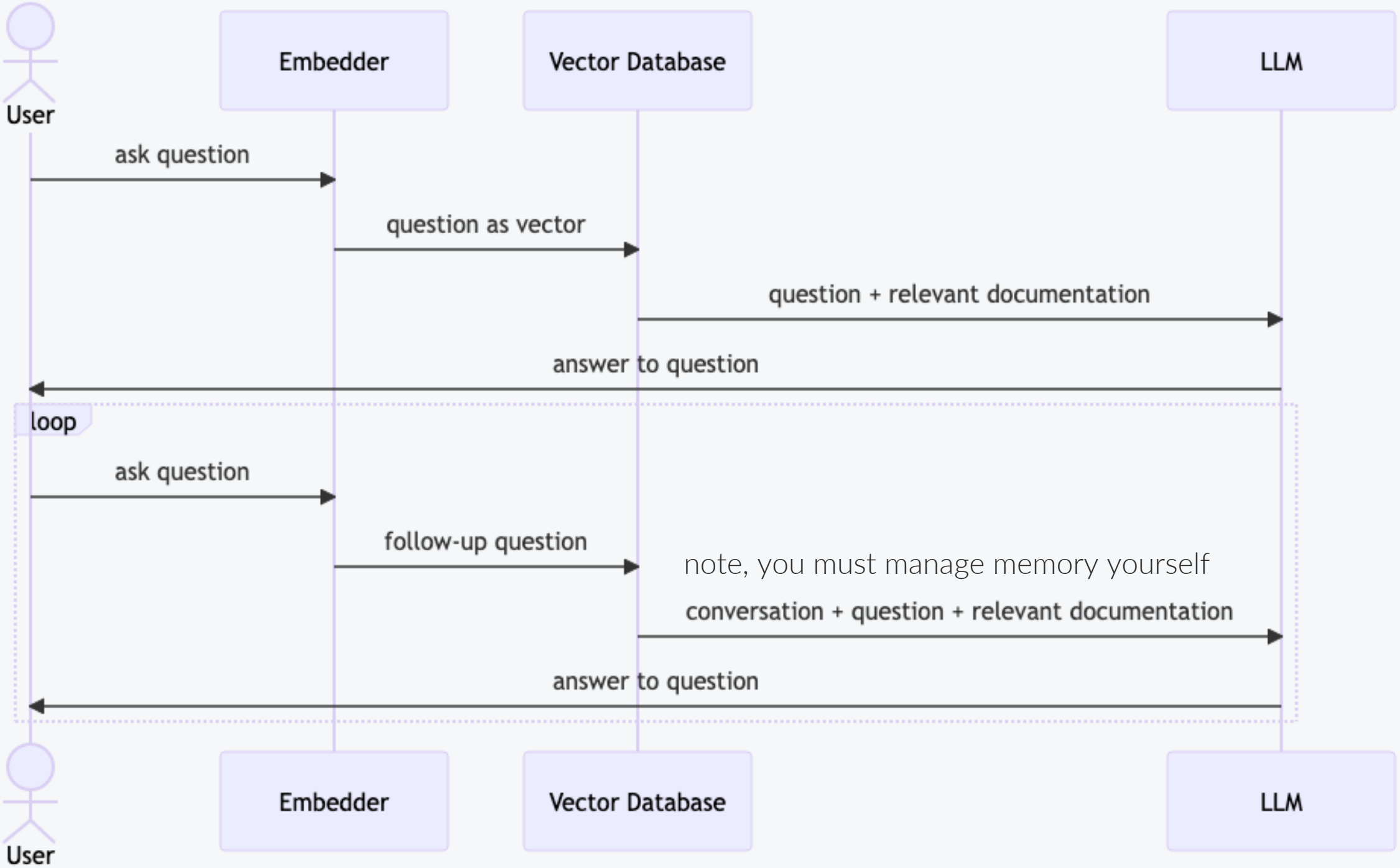
Keep same formatting steps

- textract if pdf https://textract.readthedocs.io/en/stable/python_package.html

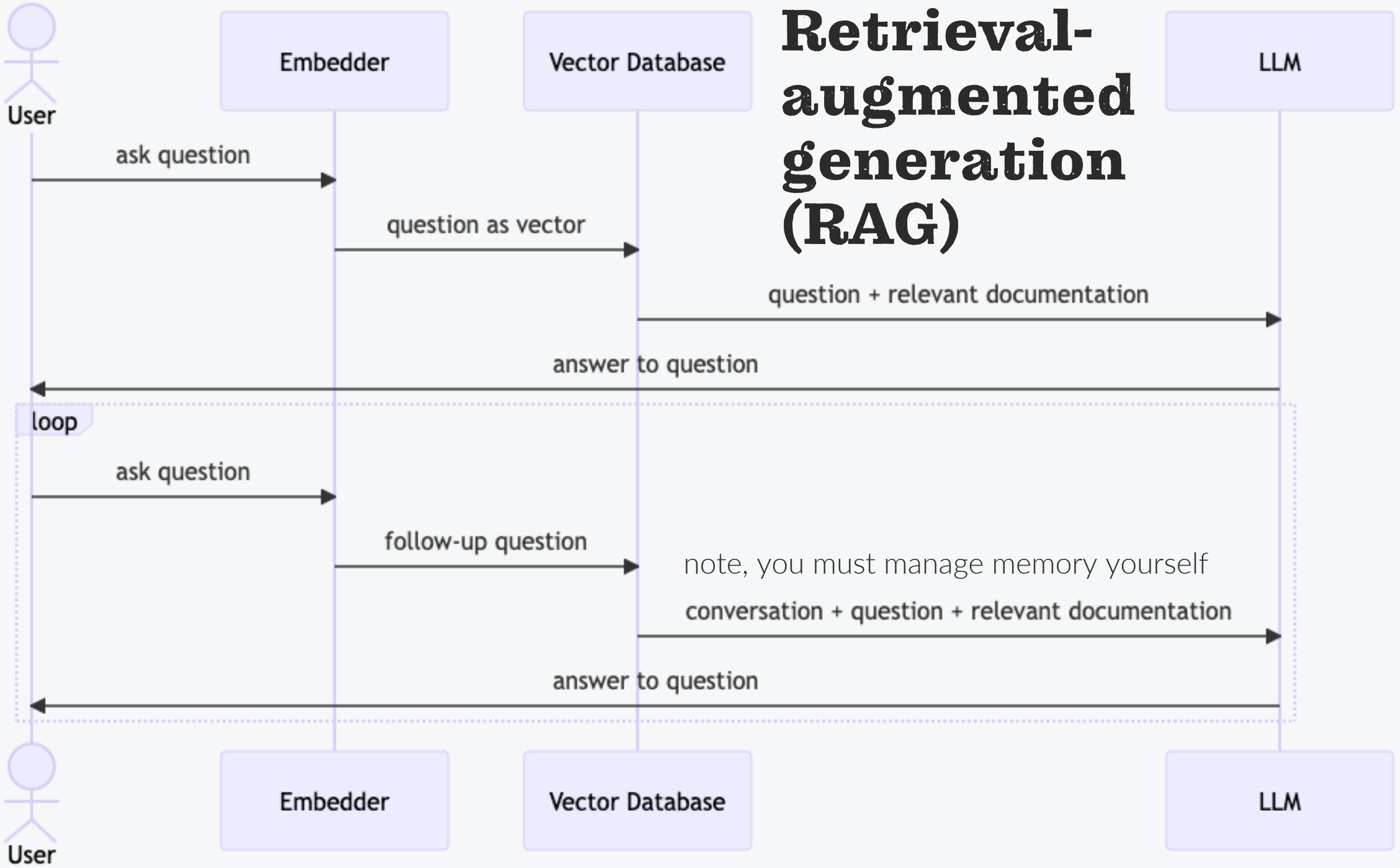
- LlamaIndex https://www.llamaindex.ai/
- LangChain https://www.langchain.com/

- Sentence-Transformers https://www.sbert.net/
- OpenAI https://platform.openai.com/docs/guides/embeddings
- TensorFlow Universal Sentence Encoder https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder

- Milvus https://milvus.io/
- Pinecone https://www.pinecone.io/
- Azure AI Search (formally Cognitive Search) https://learn.microsoft.com/en-us/azure/search/
- Memory with FAISS https://github.com/facebookresearch/faiss or scikit-learn https://scikit-learn.org/stable/


User
Embedder
Vector Database
LLM
ask question
question as vector
question + relevant documentation
answer to question
loop
ask question
follow-up question
note, you must manage memory yourself
conversation + question + relevant documentation
answer to question
User
Embedder
Vector Database
LLM

Retrieval-augmented generation (RAG)



Notes about RAG

- Since the LLM is trained on the whole internet, it will always answer. If your data is not sufficient it will answer anyway.
- LLMs have no history, you must manage it yourself

More here
https://twitter.com/jerryjliu0/status/1744407483719045583

Software frameworks

You can use software frameworks to make this process easier, but …

for each framework, you will need to work through the abstractions and how they fit together

- LlamaIndex https://www.llamaindex.ai/
- LangChain https://www.langchain.com/

Some experience with Azure

- Azure AI search (formerly known as Azure Cognitive Search) https://learn.microsoft.com/en-us/azure/search/search-what-is-azure-search
- Rest API interface https://learn.microsoft.com/en-us/rest/api/searchservice/
- GitHub Azure vector search samples
  - https://github.com/Azure/azure-search-vector-samples/tree/main/demo-python
- Python SDK
  - https://github.com/Azure/azure-sdk-for-python/blob/main/sdk/search/azure-search-documents
  - Python SDK documentation https://azuresdkdocs.blob.core.windows.net/$web/python/azure-search-documents/11.0.0/index.html
- Hybrid search! https://github.com/Azure/azure-search-vector-samples/blob/main/demo-python/code/azure-search-vector-python-sample.ipynb
- Nice description and video https://www.microsoft.com/en-us/research/blog/the-science-behind-semantic-search-how-ai-from-bing-is-powering-azure-cognitive-search
  - https://youtu.be/d_6ZNyV1MvA what is the capital of France
- Also see promptflow, a flow builder https://github.com/microsoft/promptflow
  - compare with Langflow https://github.com/logspace-ai/langflow

Structured Output

Sometimes you don't want answers in natural language; you want the answers in the form a computer can understand, like JSON.

ChatGPT can do that through the API.

```
"type": "function",
"function": {
    "name": "get_current_weather",
    "description": "Get the current weather",
    "parameters": {
        "type": "object",
        "properties": {
            "location": {
                "type": "string",
                "description": "The city and state, e.g. San Francisco, CA",
            },
            "format": {
                "type": "string",
                "enum": ["celsius", "fahrenheit"],
                "description": "The temperature unit to use. Infer this from
            },
        },
        "required": ["location", "format"],
    },
}
```

Example from https://cookbook.openai.com/examples/how_to_call_functions_with_chat_models

It's a tad convoluted. You tell it that you want to make a function call, and it returns the arguments you need. You then use the "arguments" as your JSON.

- Part of the API process is telling ChatGPT the data types of the "arguments"
- See https://platform.openai.com/docs/guides/function-calling and https://cookbook.openai.com/examples/how_to_call_functions_with_chat_models
- Another example, https://github.com/openai/openai-cookbook/blob/main/examples/Function_calling_with_an_OpenAPI_spec.ipynb

Structured Output (2)

Other tools (I haven't used these output parsers)

- LangChain Pydantic Parser
  - Last I checked it was just an instruction in the prompt https://python.langchain.com/docs/modules/model_io/output_parsers/pydantic
  - source code https://api.python.langchain.com/en/latest/_modules/langchain/output_parsers/pydantic.html#PydanticOutputParser
- LlamaIndex
  - https://docs.llamaindex.ai/en/stable/examples/finetuning/openai_fine_tuning_functions.html
  - https://docs.llamaindex.ai/en/stable/examples/output_parsing/df_program.html

Structured Output (3)

If you have access to the model, you can guide it by only allowing it to emit tokens that fit some structure you specify. (I haven't used these either.)

Guidance https://github.com/microsoft/guidance. It clamps down the tokens to make them fit the parameter you sent it, such as a JSON.

LMQL: Prompting is Programming
https://lmql.ai/
https://docs.lmql.ai/en/latest/
https://arxiv.org/pdf/2212.06094.pdf

But let's talk about something important

- Flexible search over your documents
- Question answering over your documents
- Exact calculations and decisions
- Considerations when using LLMs
- How LLMs Work

Exact calculations and decisions

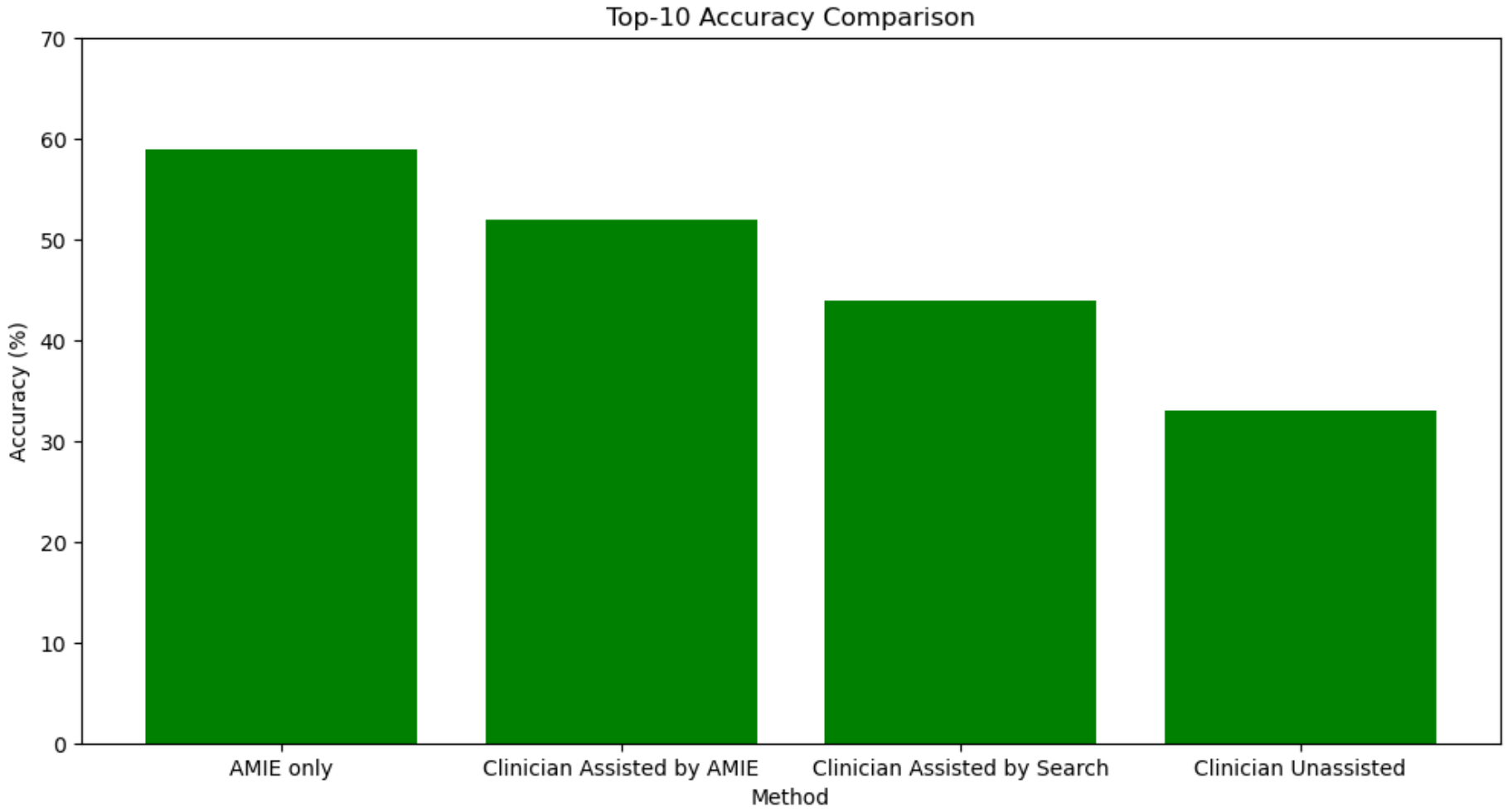
I’m not a superhero movie person, but my family is, so I’ve seen them all.

Anyway, we are close to this scene where Tony Stark works out time travel in *Avengers: Endgame*. Well, maybe not time travel, but we can now talk to computers, and they can infer what we want and make calculations on their own.

AI can help clinicians with medical cases

Articulate Medical Intelligence Explorer (AIME)

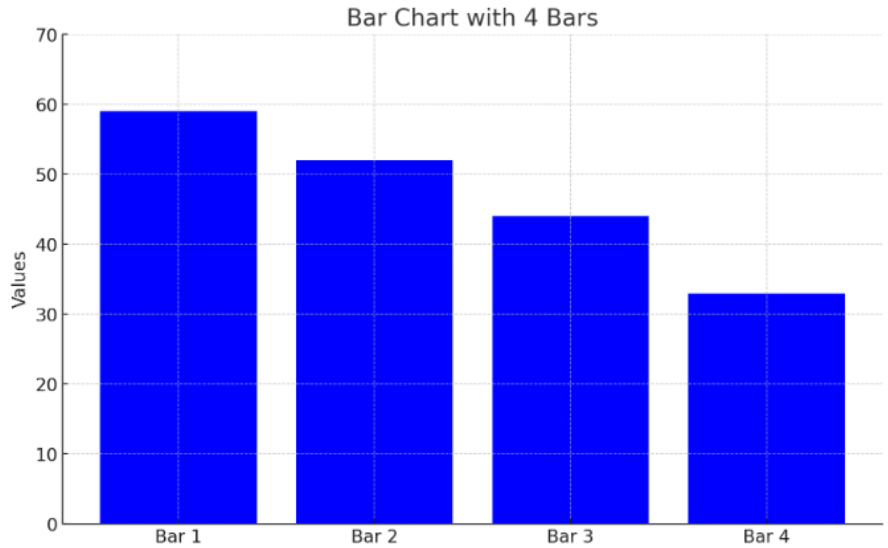
Adapted figure from
https://blog.research.google/2024/01/amie-research-ai-system-for-diagnostic_12.html



You

Draw me a bar chart with 4 bars: one at 59, one at 52, one at 44, and one at 33

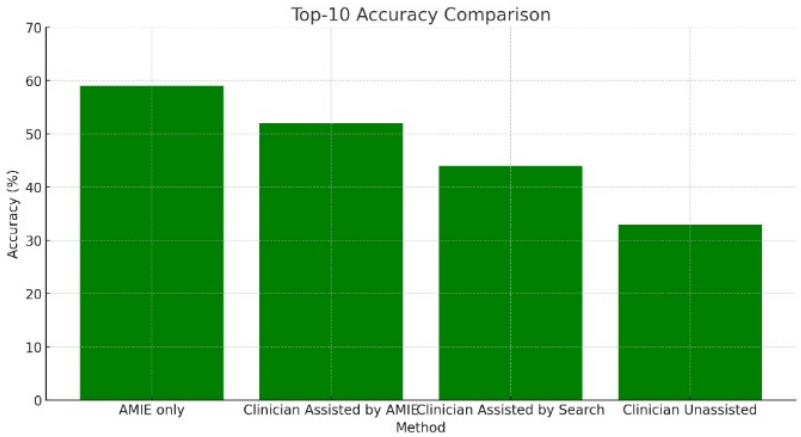
ChatGPT



You

Make the y axis accuracy and make "Bar 1" be "AMIE only", "Bar 2" be "Clinician Assisted by AMIE", "Bar 3" be "Clinician Assisted by Search" and "Bar 4 be "Clinician Unassisted". Make the title be "Top-10"

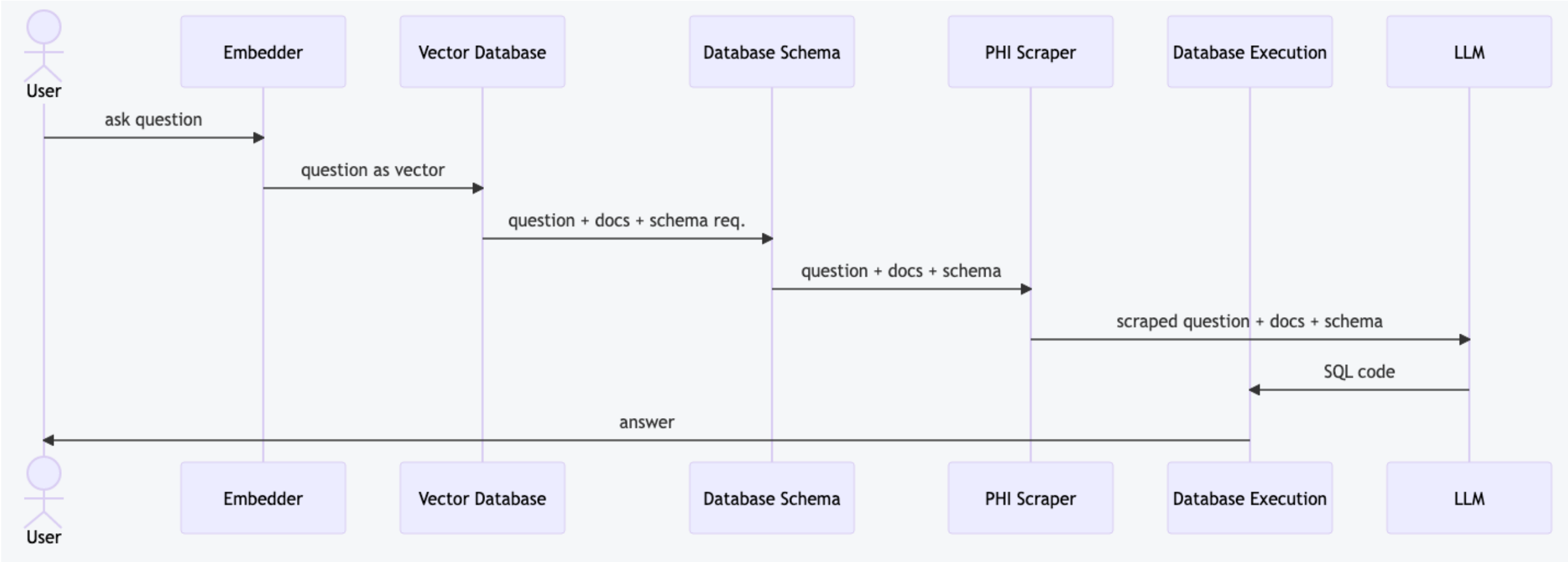
ChatGPT




```python
import matplotlib.pyplot as plt

# Updating data for the bar chart with new label
bar_labels = ['AMIE only', 'Clinician Assisted b
bar_values = [59, 52, 44, 33]

# Creating the updated bar chart
plt.figure(figsize=(12, 6))
plt.bar(bar_labels, bar_values, color='green')

# Adding titles and labels
plt.title('Top-10 Accuracy Comparison')
plt.xlabel('Method')
plt.ylabel('Accuracy (%)')
plt.ylim(0, 70)  # Setting y-axis limit for bett

# Display the plot
plt.show()
```

Tool use example: SQL

- It can generate Python code and also SQL.
- It can use the structured output to call other tools that have well-defined interfaces.



But let's talk about something important

- Flexible search over your documents
- Question answering over your documents
- Autonomous calculations and decisions
- Considerations when using LLMs
- How LLMs Work

Hallucination

- Because LLMs don't have a grounded understanding of the world, they are prone to make up answers.
  - See https://thegradient.pub/grounding-large-language-models-in-a-cognitive-foundation/
- Some people prefer the term confabulation
- Best if there is a way to check answers quickly or automatically (depends on your domain)

Mitigate through prompting:

- Tell it in the prompt, if you don't know say you don't know (only helps a little)

Mitigate through fine-tuning

- LoRA and QLoRA
- AIME used instruction tuning—fine tuning on a particular set of instructions.

Privacy

People still working out when they can send data to the cloud.

As a workaround, you can use Presidio to automatically identify and replace PII with randomly generated values.

- https://microsoft.github.io/presidio/samples/python/customizing_presidio_analyzer
- https://microsoft.github.io/presidio/analyzer/adding_recognizers
- https://github.com/microsoft/presidio

Easiest place to start is to use it with LangChain

- https://python.langchain.com/docs/guides/privacy/presidio_data_anonymization

You Can Run LLMs Locally

GPU memory is the bottleneck

Mitigate Quantization with https://github.com/TimDettmers/bitsandbytes/

7B are increasingly good, such as Mistral
https://openpipe.ai/blog/mistral-7b-fine-tune-optimized
https://news.ycombinator.com/item?id=38712802

Local models and finetuning models works better if your domain is niche, but for domains that require general common sense I've found that one of the big models, such as GPT3.5 or GPT4, is best.

Good place to find out what the good models are

🏆 LMSYS Chatbot Arena Leaderboard

| Vote | Blog | GitHub | Paper | Dataset | Twitter | Discord |

LMSYS Chatbot Arena is a crowdsourced open platform for LLM evals. We've collected over **200,000** human preference votes to rank LLMs with the Elo ranking system.

Arena Elo | Full Leaderboard

Total #models: **54**. Total #votes: **213576**. Last updated: Jan 9, 2024.

Contribute your vote 🗳️ at chat.lmsys.org! Find more analysis in the notebook.

Rank	Model	Arena Elo	95% CI	Votes	Organization	License
1	GPT-4-Turbo	1249	+14/-13	23069	OpenAI	Proprietary
2	GPT-4-0314	1190	+14/-14	16237	OpenAI	Proprietary
3	GPT-4-0613	1160	+14/-12	20884	OpenAI	Proprietary
4	Mistral Medium	1150	+15/-13	6586	Mistral	Proprietary
5	Claude-1	1149	+15/-13	16956	Anthropic	Proprietary
6	Claude-2.0	1131	+14/-13	11204	Anthropic	Proprietary
7	Mixtral-8x7b-Instruct-v0.1	1123	+15/-13	12469	Mistral	Apache 2.0
8	Gemini Pro (Dev)	1120	+18/-18	1898	Google	Proprietary
9	Claude-2.1	1119	+14/-12	20883	Anthropic	Proprietary
10	GPT-3.5-Turbo-0613	1116	+13/-13	26583	OpenAI	Proprietary

https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

“Cheese shop” sketch by Monte Python gives you a sense of the experience of renting cloud GPUs

Cheese shop on getting GPUs

- How about a little A100?
  - I’m afraid we are fresh out of A100 sir.
- Never mind, how are you on V100?
  - Only at the first of the week, sir
- Tsk, Tsk, well, a T4 if you please.
  - Ohhh … the cat’s eaten it.

It’s really like that. You down all these menus of choices and configuration and just when you are ready to deploy, you find out you can’t for some reason.

Serverless GPUs:
I’ve had good luck with https://gradient.ai/

Find sketch here

https://www.youtube.com/watch?v=Hz1JWzyvv8A&t=3s

But let's talk about something important

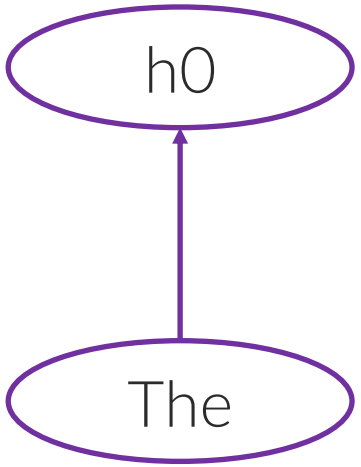
- Flexible search over your documents
- Question answering over your documents
- Exact calculations and decisions
- Considerations when using LLMs
- How LLMs Work

Three technologies enabled ChatGPT

1. Language Modeling
2. Transformers
3. Instruction Tuning

It began with machine translation

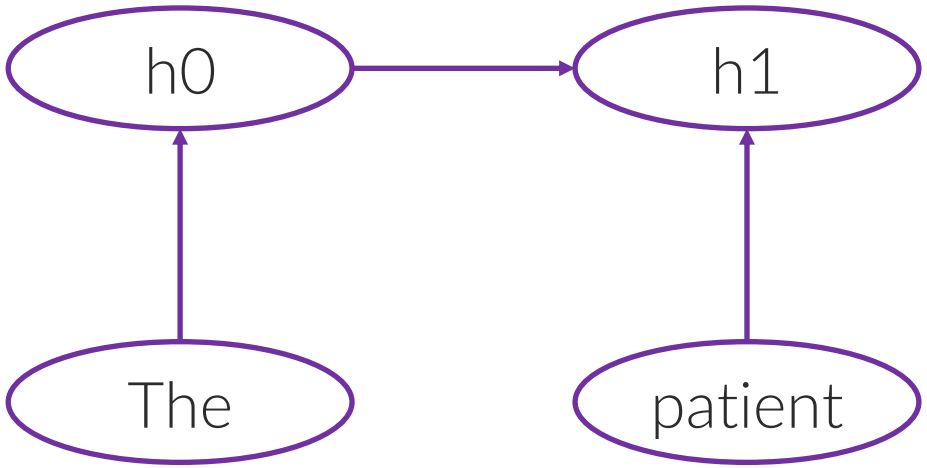
"The patient fell."



Using a recurrent neural network (RNN).

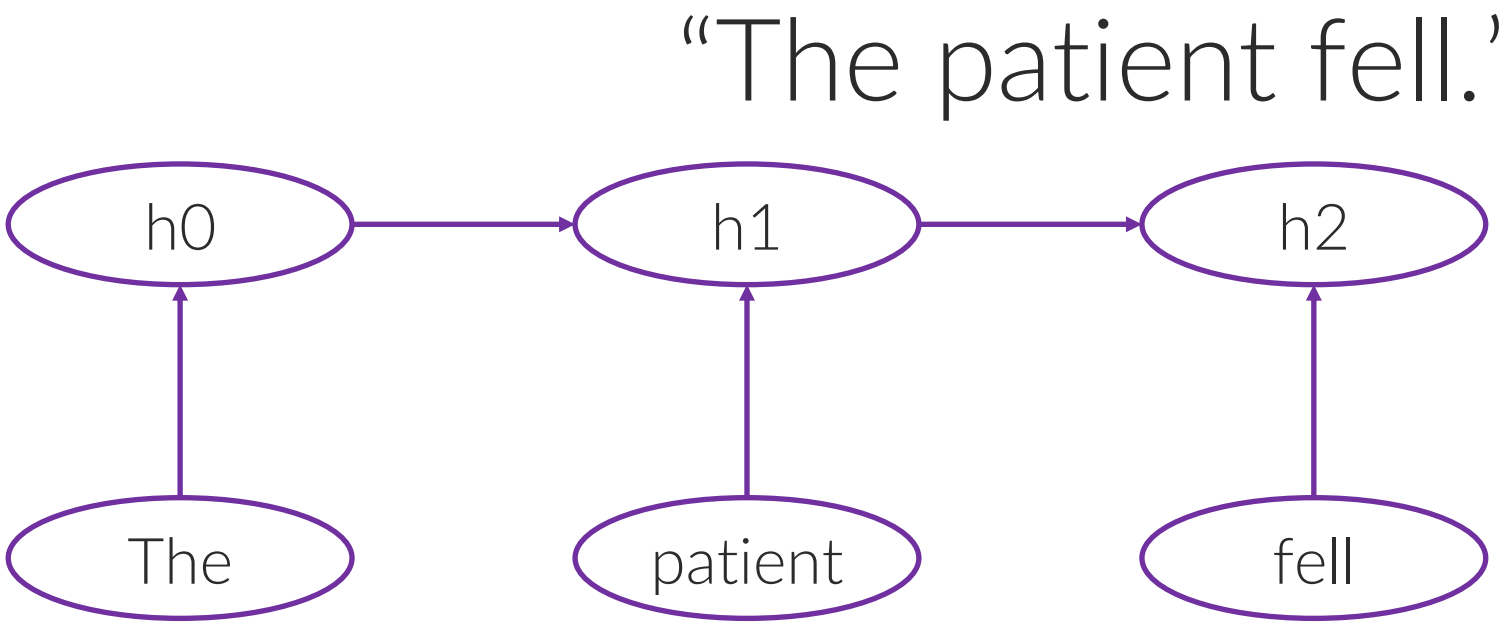
Encoding sentence meaning into a vector

"The patient fell."



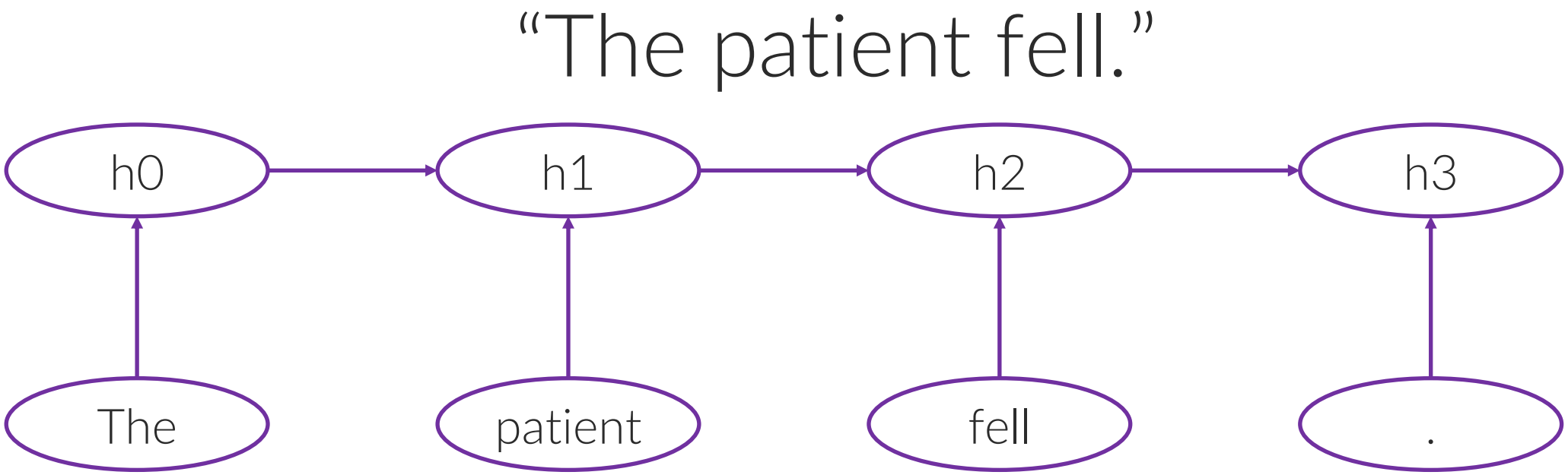
Using a recurrent neural network (RNN).

Encoding sentence meaning into a vector



Using a recurrent neural network (RNN).

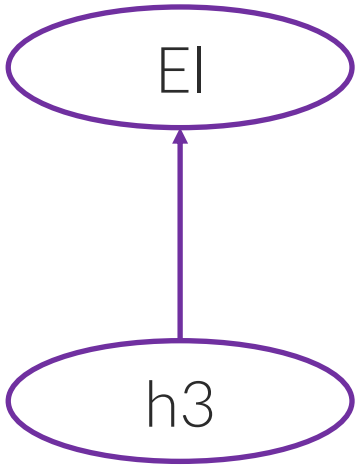
Encoding sentence meaning into a vector



RNN is like a hidden Markov model but doesn't make the Markov assumption and benefits from a vector representation.

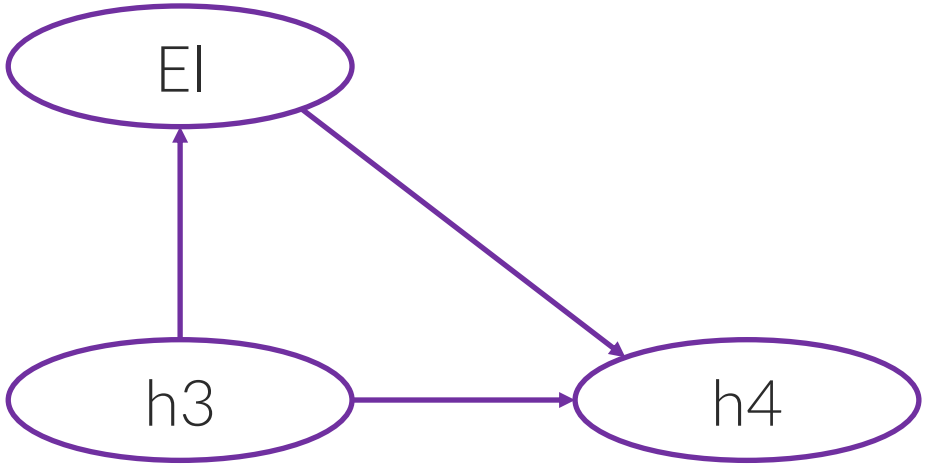
Decoding sentence meaning

Machine translation.



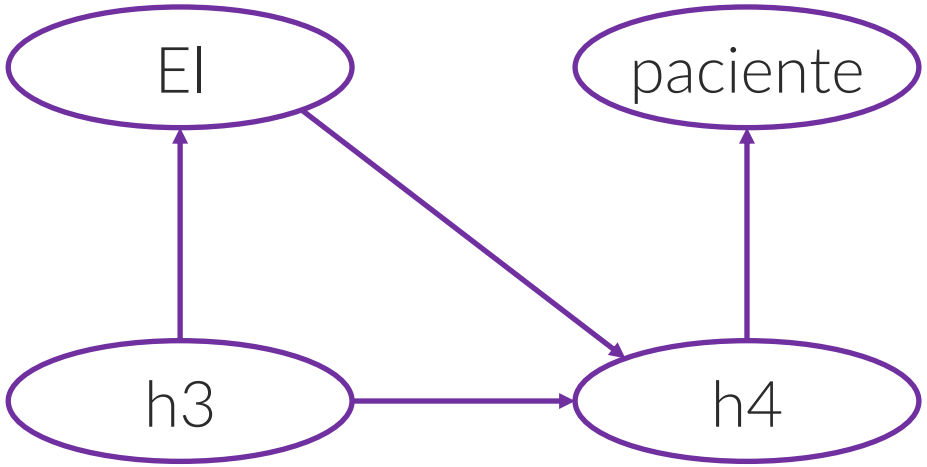
Decoding sentence meaning

Machine translation.



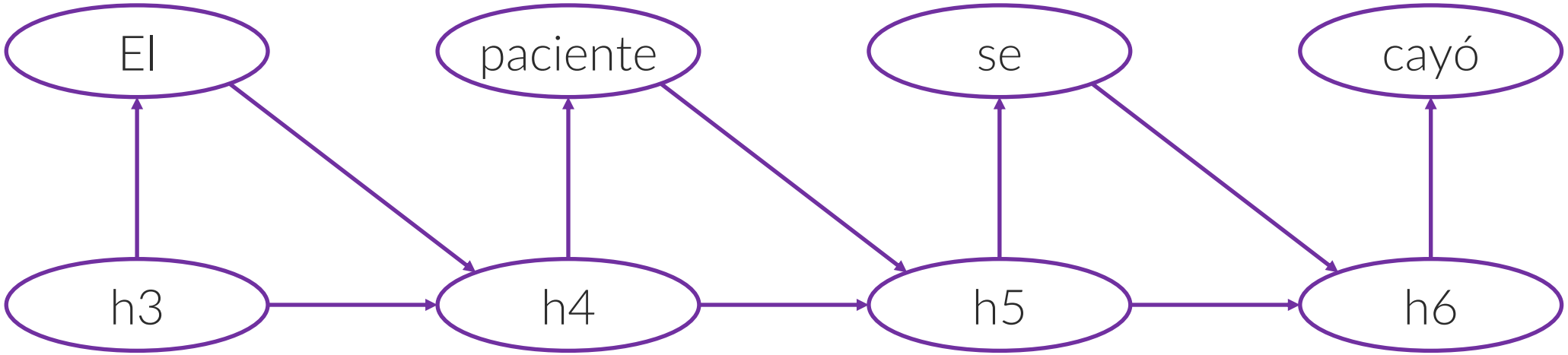
Decoding sentence meaning

Machine translation.



Decoding sentence meaning

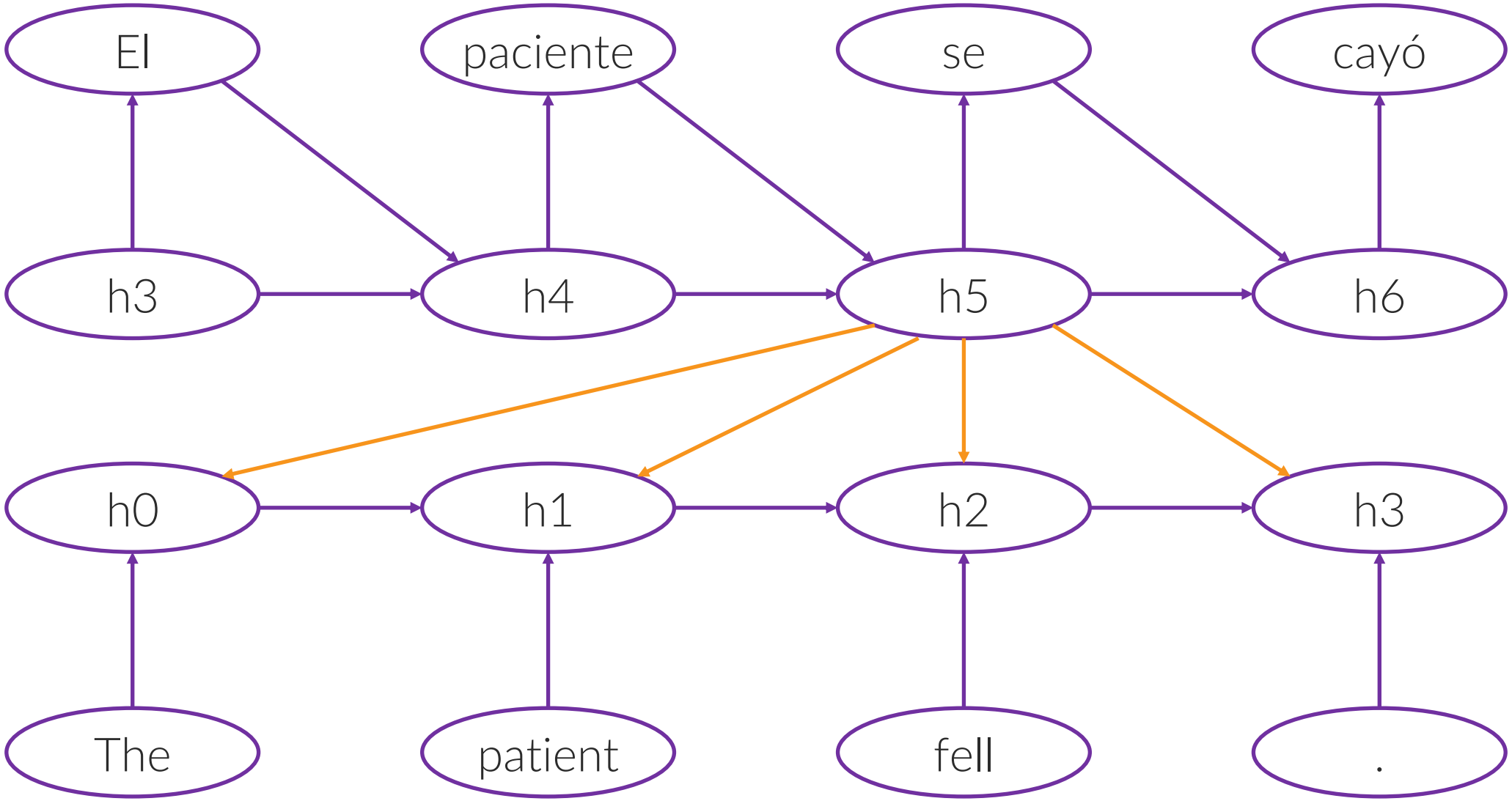
Machine translation.



[Cho et al., 2014]

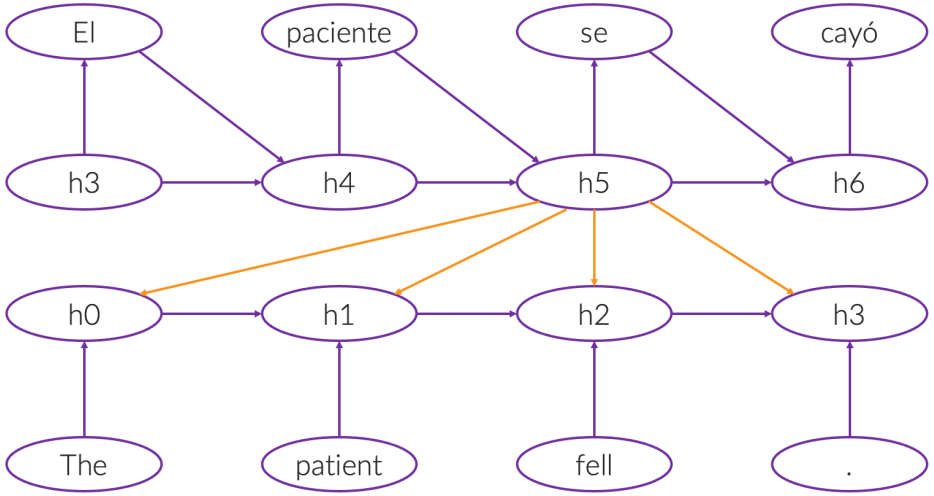
- It keeps generating until it generates a stop symbol.
- It used a kind of interpolation from a huge set of training data.

Attention [Bahdanau et al., 2014]

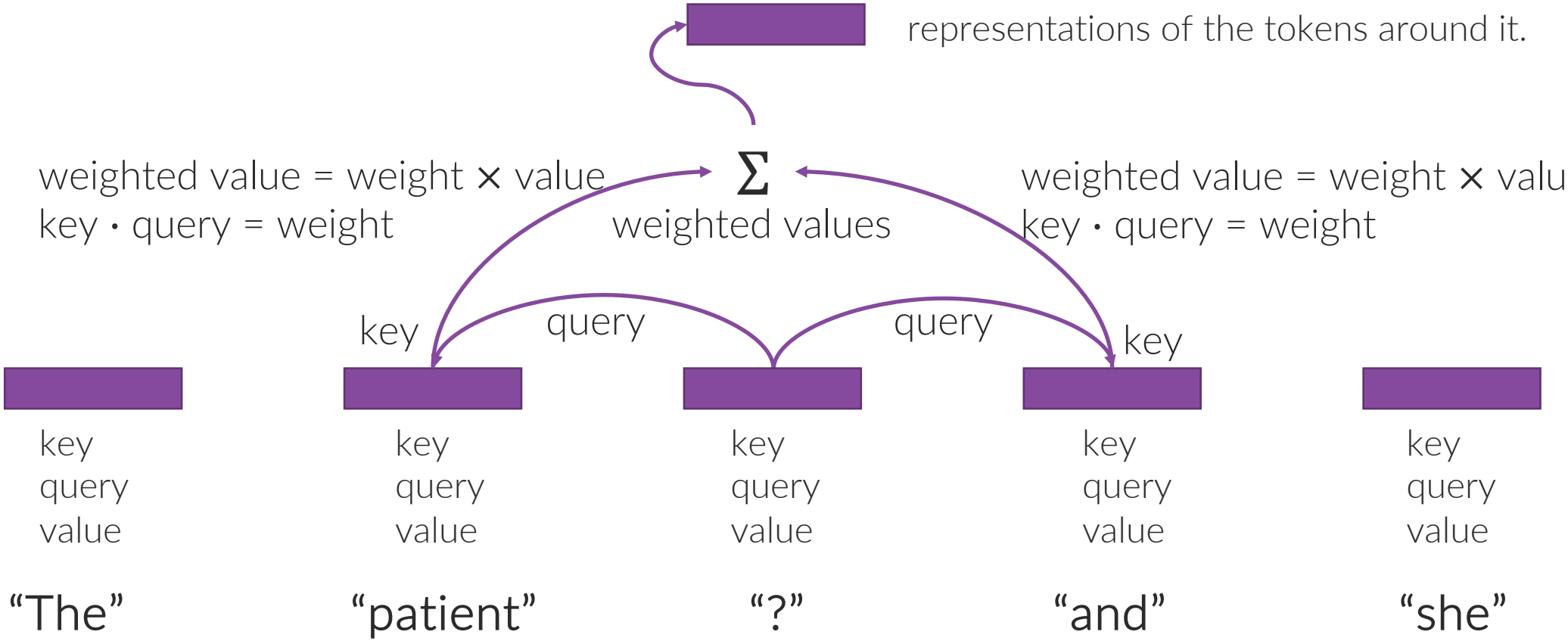


Transformers: Attention is all you need

https://arxiv.org/abs/1706.03762



How transformers think: tokens with keys, queries, and values



Recall from matrix multiplication

$X^{m \times n} Y^{n \times o} = Z^{m \times o}$

Sizes

v vocab size

d embedding size

l length of text input in tokens

Learned Weight Matrices

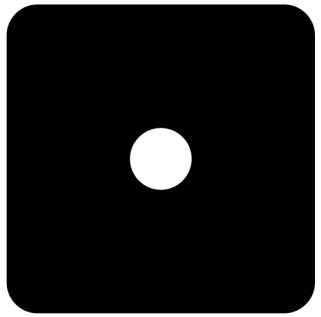
$E^{v \times d}$ embedding matrix

$W_Q^{d \times d_k}$ query matrix

$W_K^{d \times d_k}$ key matrix

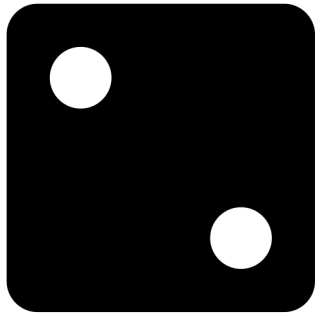
$W_V^{d \times d_v}$ value matrix

$W^{d_v \times d}$ linear



Use embedding matrix to get data

$$X^{l \times d}$$

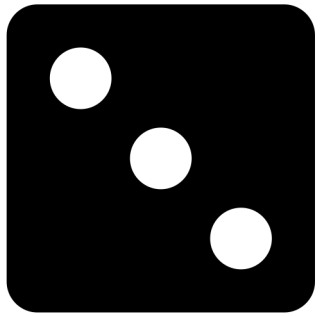


Compute queries, keys, and values

$$Q^{l \times d_k} = X^{l \times d} W_Q^{d \times d_k}$$

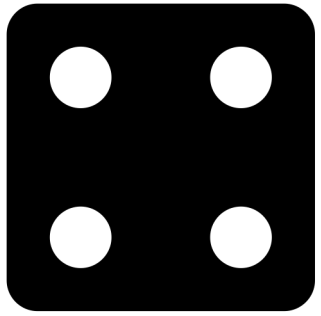
$$K^{l \times d_k} = X^{l \times d} W_K^{d \times d_k}$$

$$V^{l \times d_v} = X^{l \times d} W_V^{d \times d_v}$$



Compute attention weights

$$A^{l \times l} = Q K^T$$

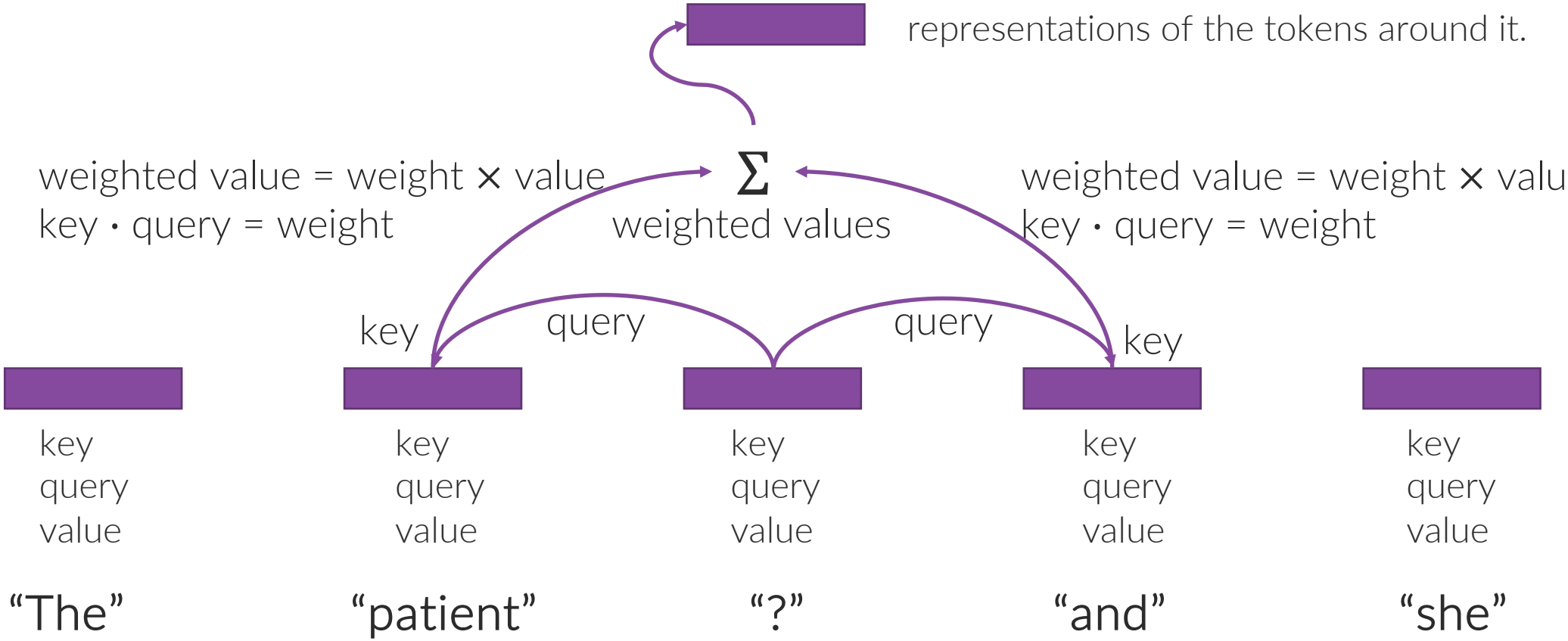


Use weights to get values and resize for next layer

$$\hat{X}^{l \times d_v} = A^{l \times l} V^{l \times d_v}$$

$$X_{next}^{l \times d} = \hat{X}^{l \times d_v} W^{d_v \times d}$$

How transformers think: tokens with keys, queries, and values



GPT-3: 96 heads, 1248 embedding size, 48 layers, plus details like positional encoding

Instruction Tuning: Reinforcement Learning with Human Feedback (RLHF)

Train an evaluation model to determine how good an output is.

1. Have humans rate outputs.
2. Train an evaluation model on those ratings. In RL, that evaluation model is called a *reward function*.

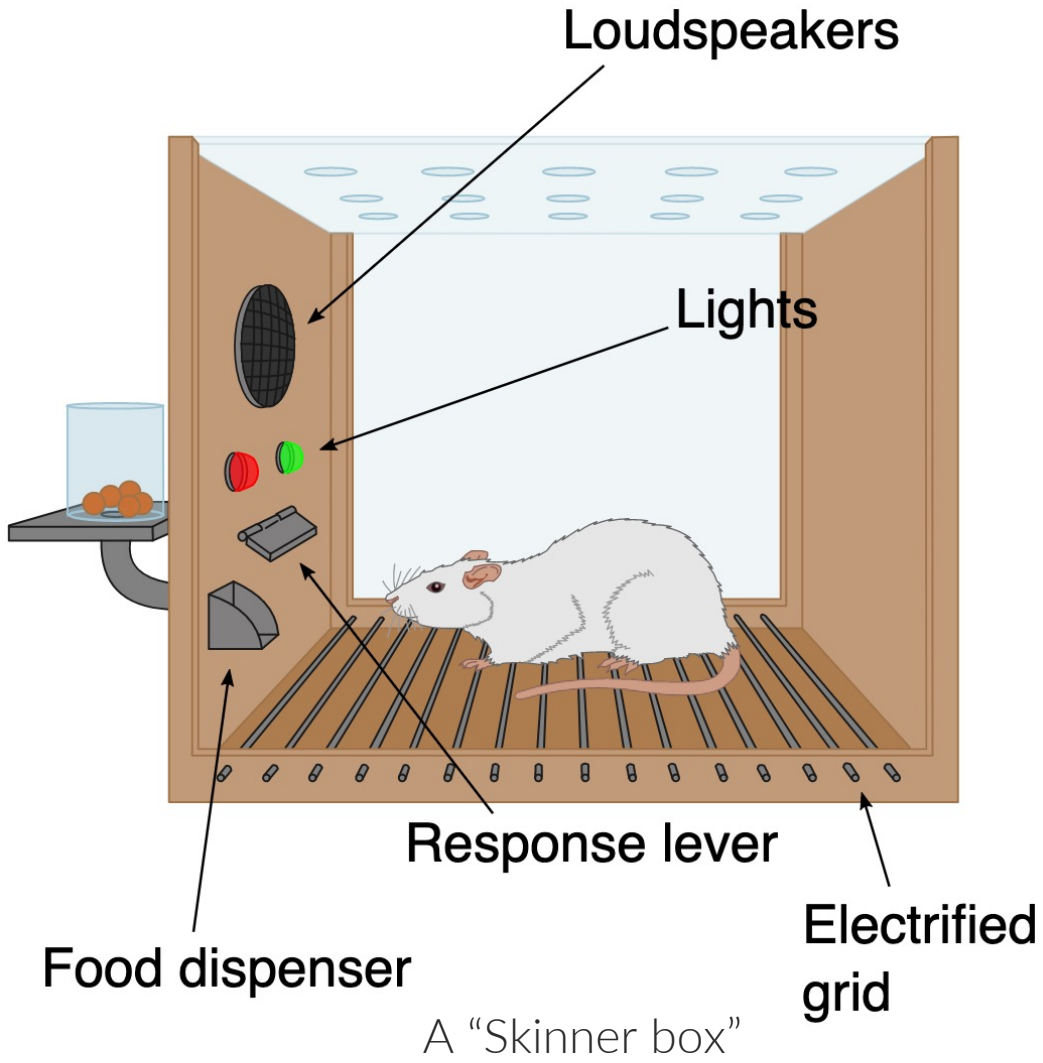
Use that evaluation model to guide autonomous learning.

- Begin with the language model trained on the internet.

RL is a gradual stamping in of behavior

Reinforcement learning: the first 100 years

- Some behaviors arise more from a gradual stamping in [Thorndike, 1898].
- Became the study of Behaviorism [Skinner, 1953] (see Skinner box on the right).
- Formulated into artificial intelligence as Reinforcement Learning [Sutton and Barto, 1998].

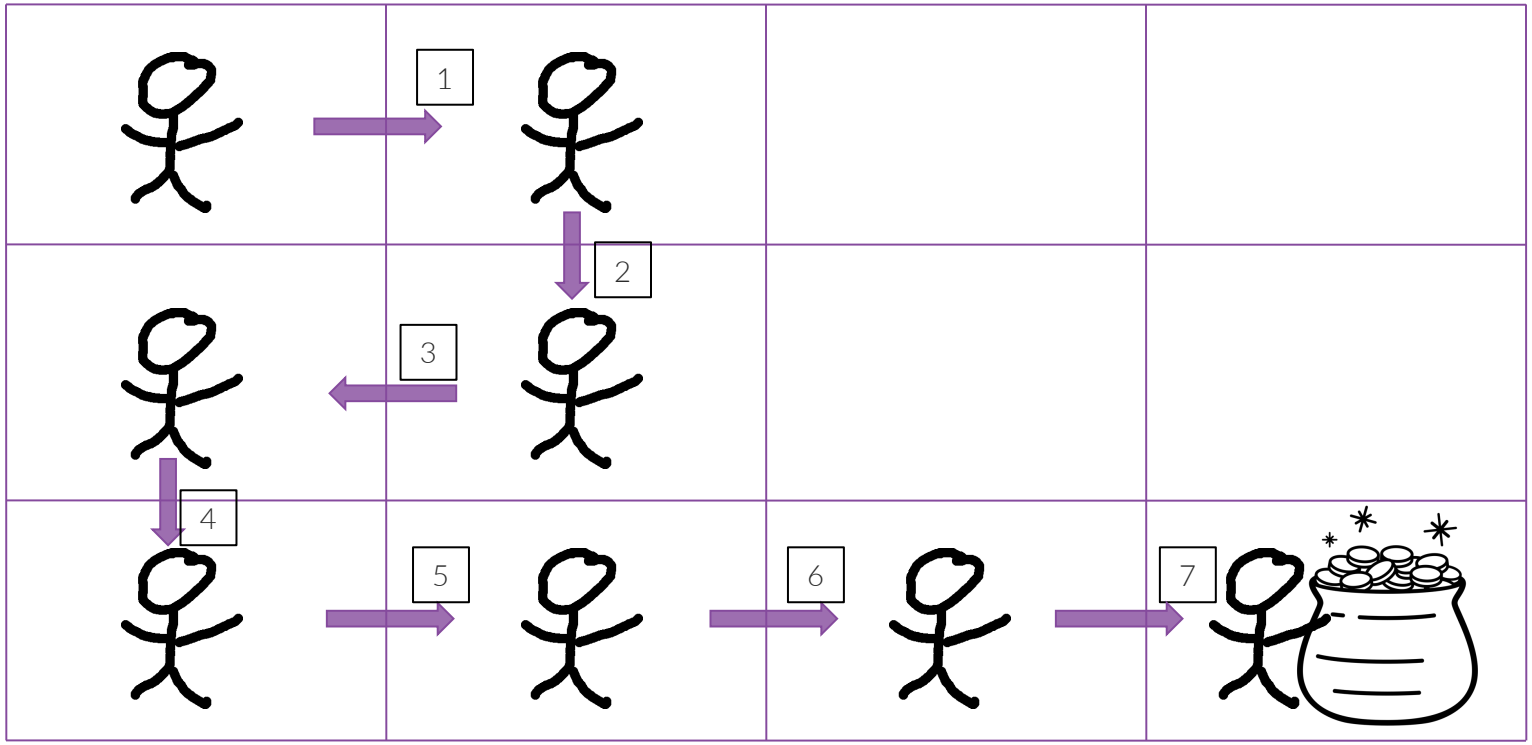


A "Skinner box"

RL in a nutshell: begin with random exploration

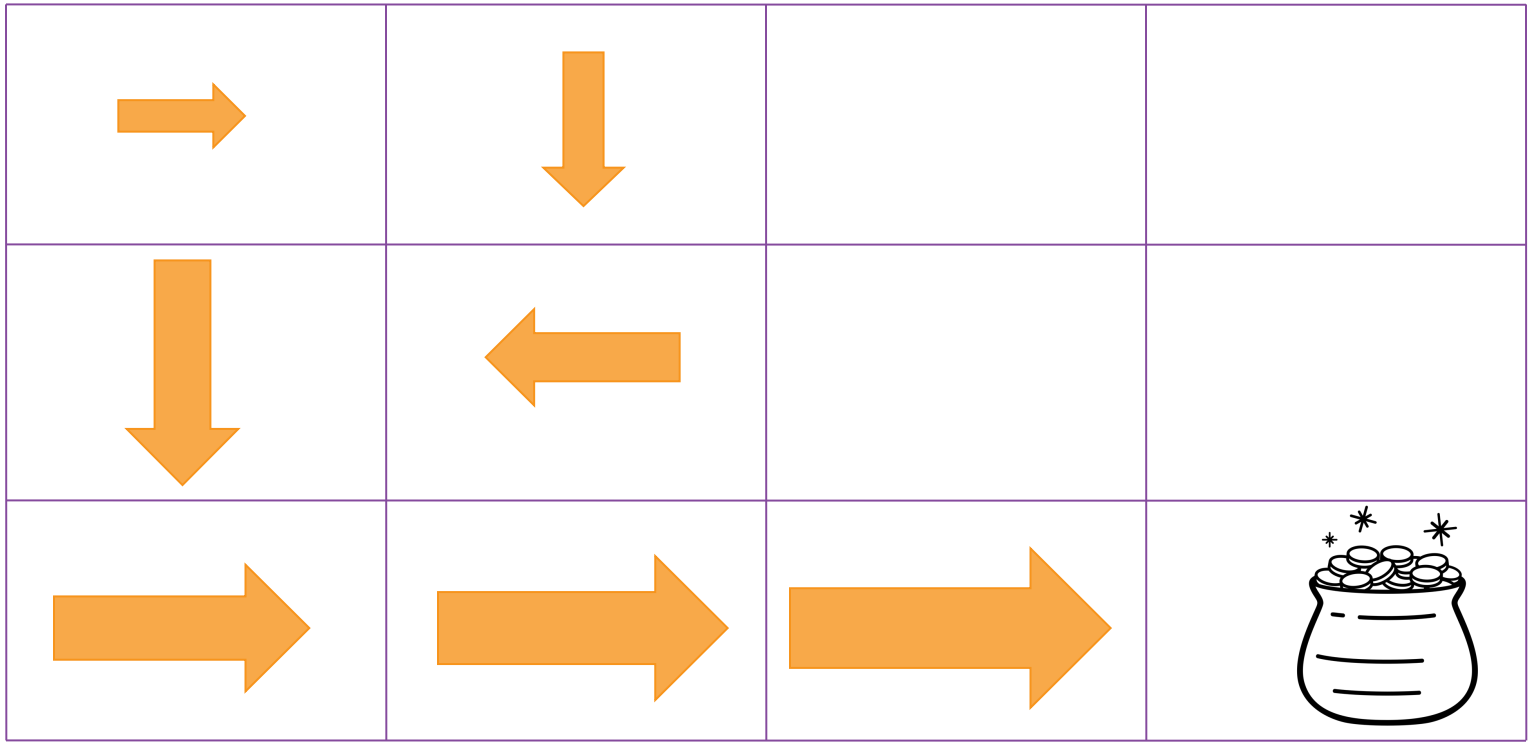
In reinforcement learning, the agent often begins by randomly exploring until it reaches its goal.

RL in a nutshell: begin with random exploration



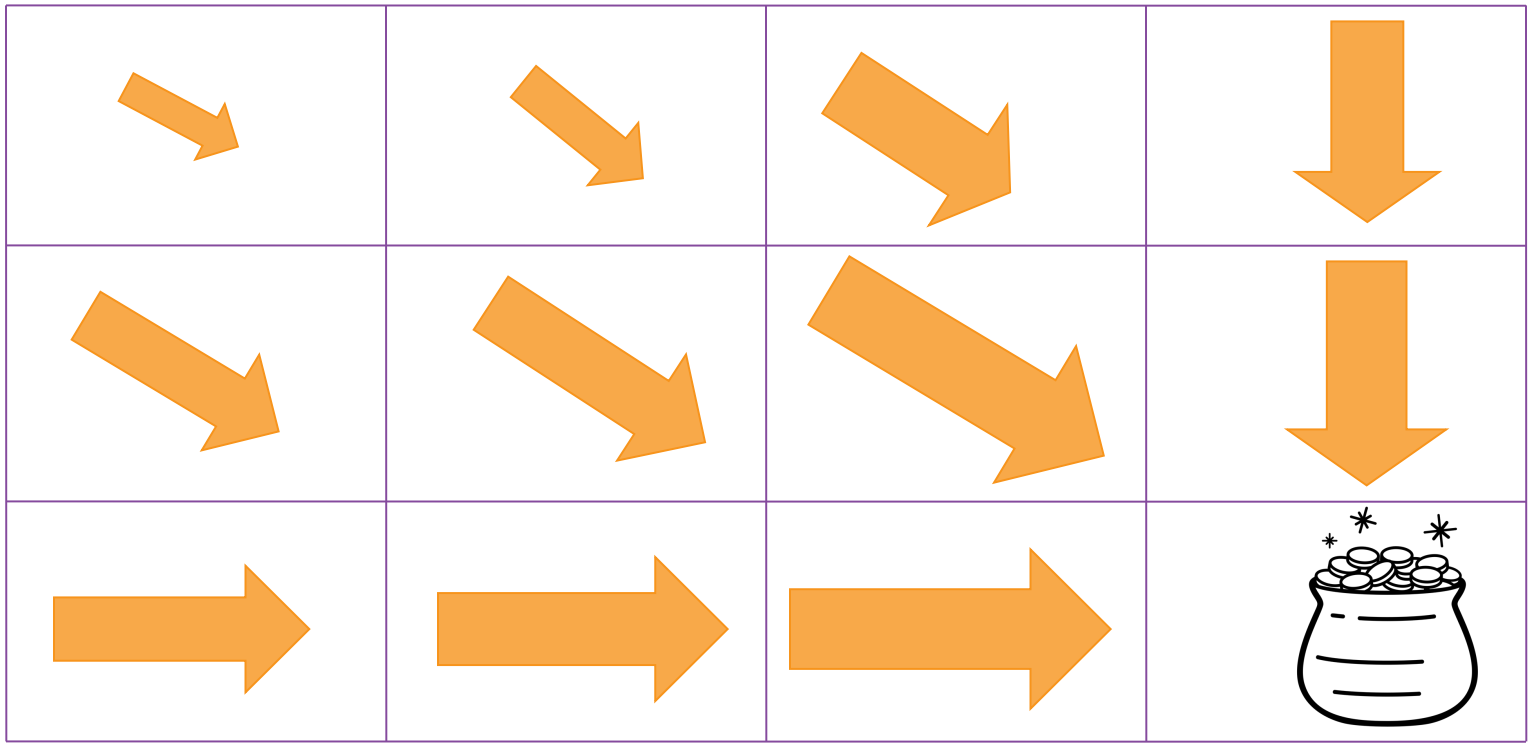
In reinforcement learning, the agent often begins by randomly exploring until it reaches its goal.

RL in a nutshell: remember what got you there



- When it reaches the goal, credit is propagated back to its previous states.
- Simplest case: the agent learns the function $Q^{\pi}(s, a)$, which gives the cumulative expected discounted reward of being in state s and taking action a and acting according to policy π thereafter. Modern uses PPO.

RL in a nutshell: learn a policy for behavior



Eventually, the agent learns the value of being in each state and taking each action and can therefore always do the best thing in each state. This behavior is then represented as a policy.

Instruction Tuning: Reinforcement Learning with Human Feedback (RLHF)

Train an evaluation model to determine how good an output is.

1. Have humans rate outputs.
2. Train an evaluation model on those ratings. In RL, that evaluation model is called a *reward function*.

Use that evaluation model to guide autonomous learning.

- Begin with the language model trained on the internet.

RLHF: Reinforcement Learning with Human Feedback
https://openai.com/research/instruction-following
https://arxiv.org/pdf/2203.02155.pdf

201 West 5th Street,
Suite 1575
Austin, TX. 78701