

Introduction to LLMs and Agentic AI with Applications in Service Sciences

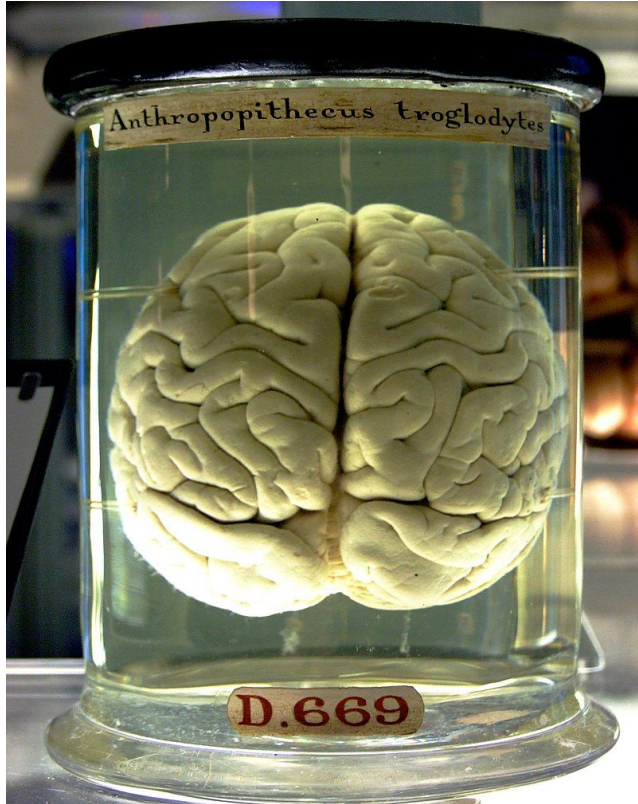
with Applications in Service Sciences

Tomáš Brázdil



M U N I

Neural Networks



Real vs artificial

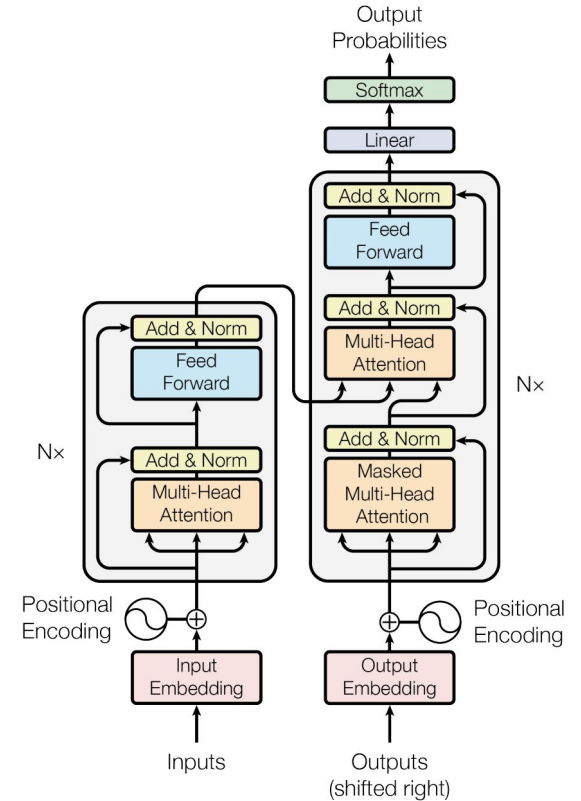
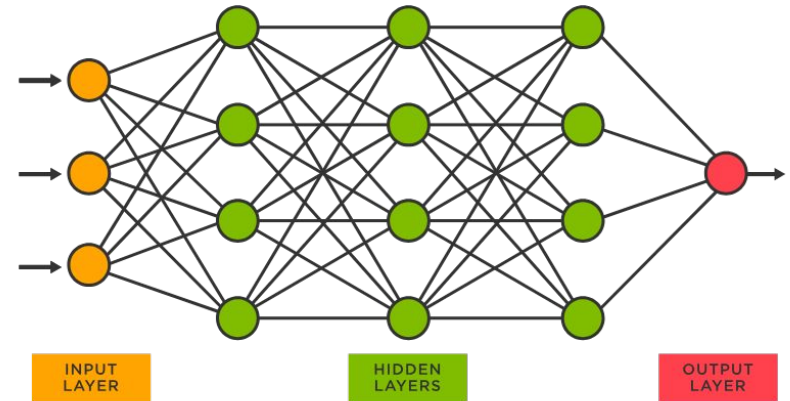
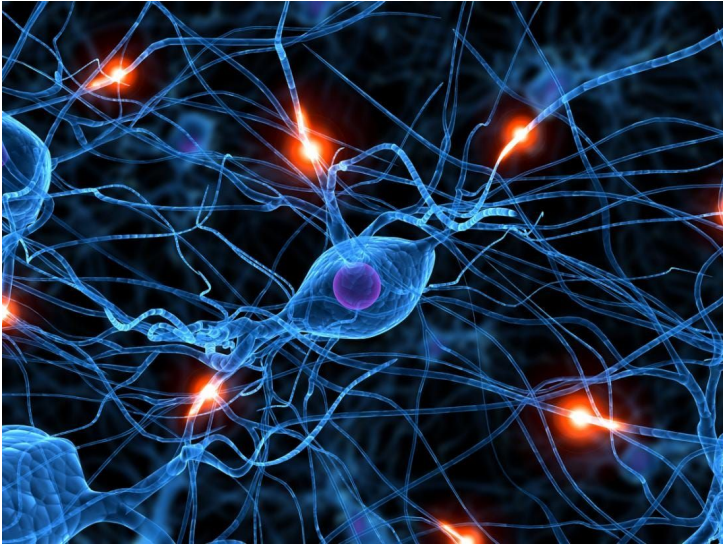


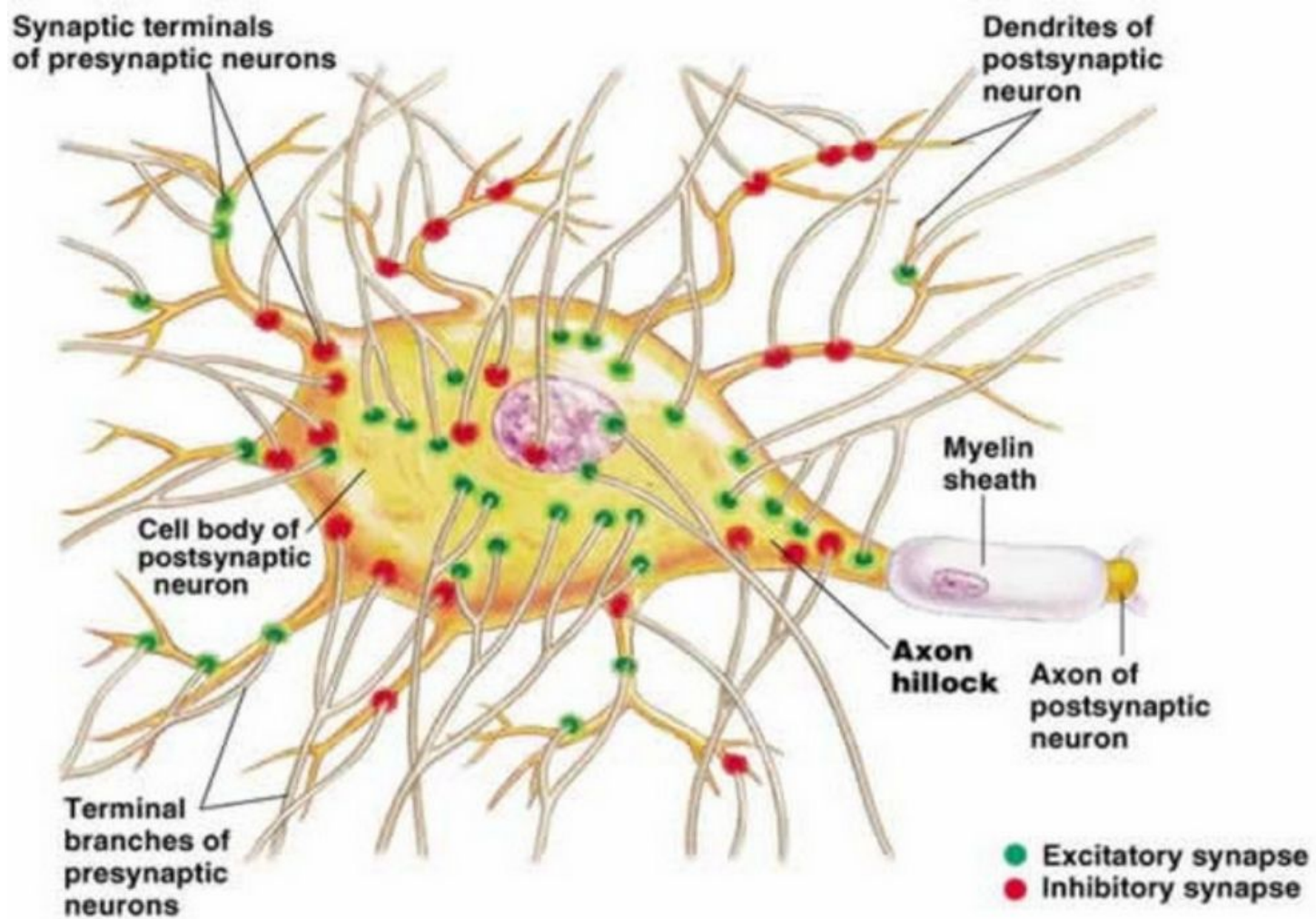
Figure 1: The Transformer - model architecture.

Neural Network

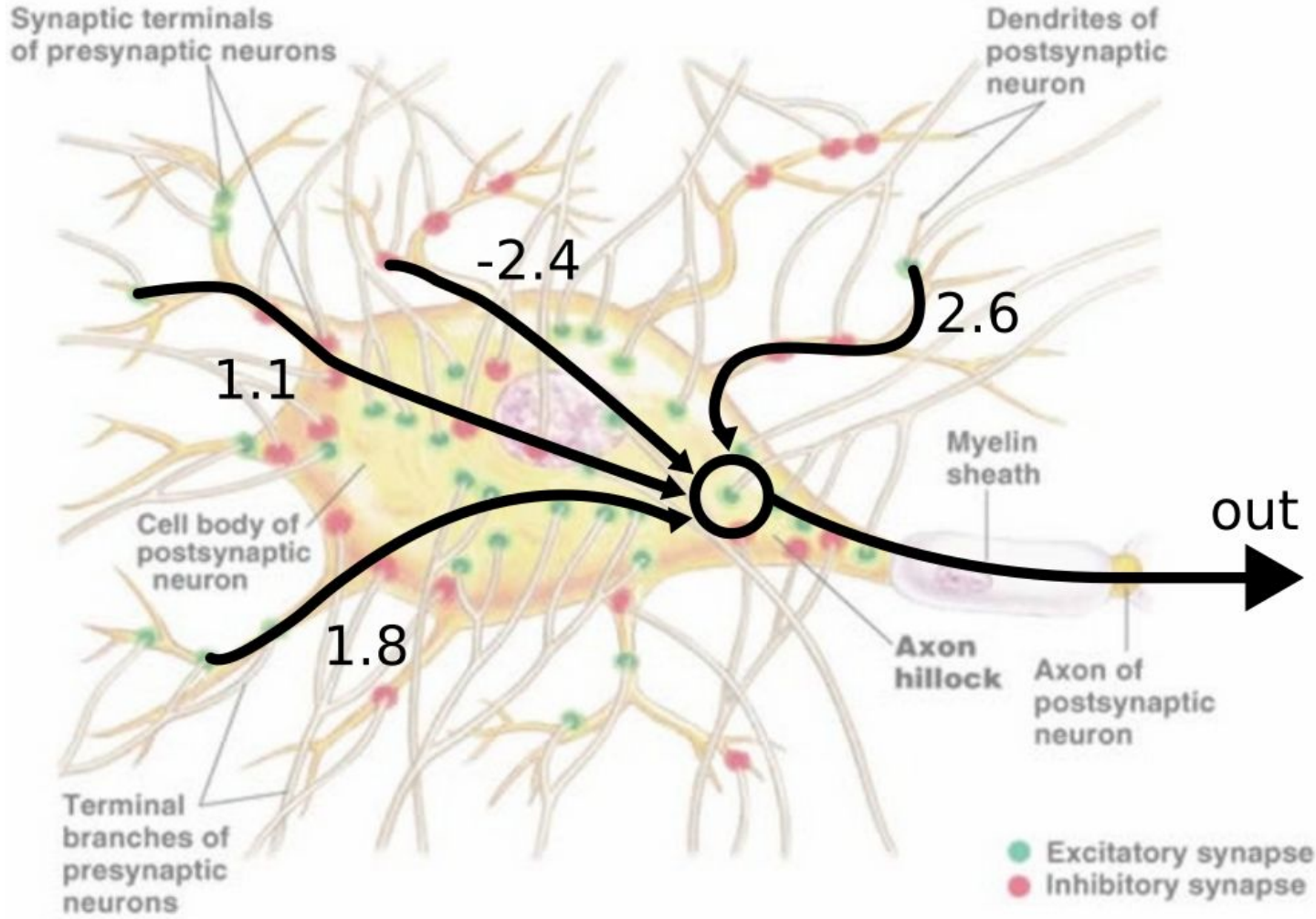


How does it work?

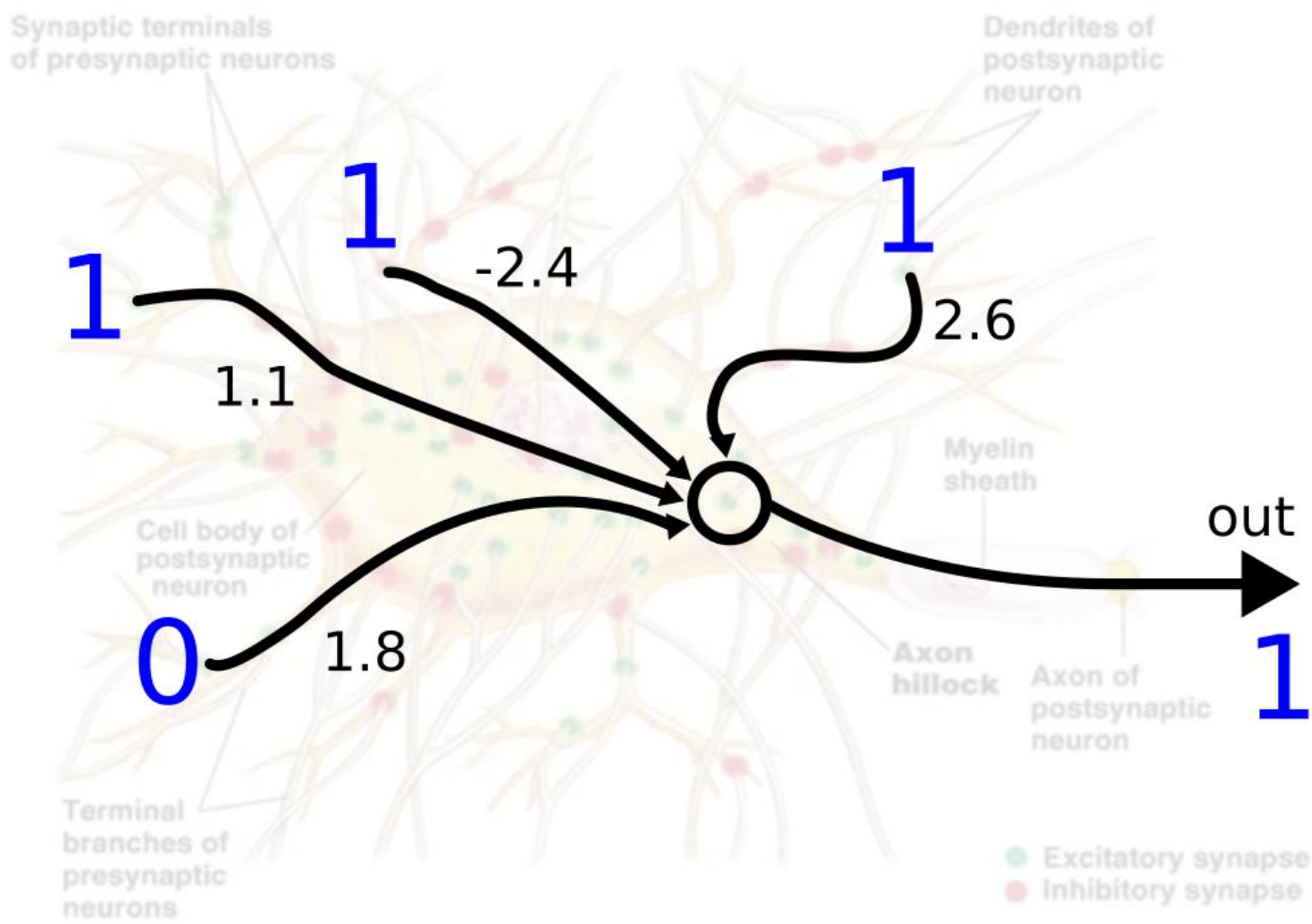
Neuron



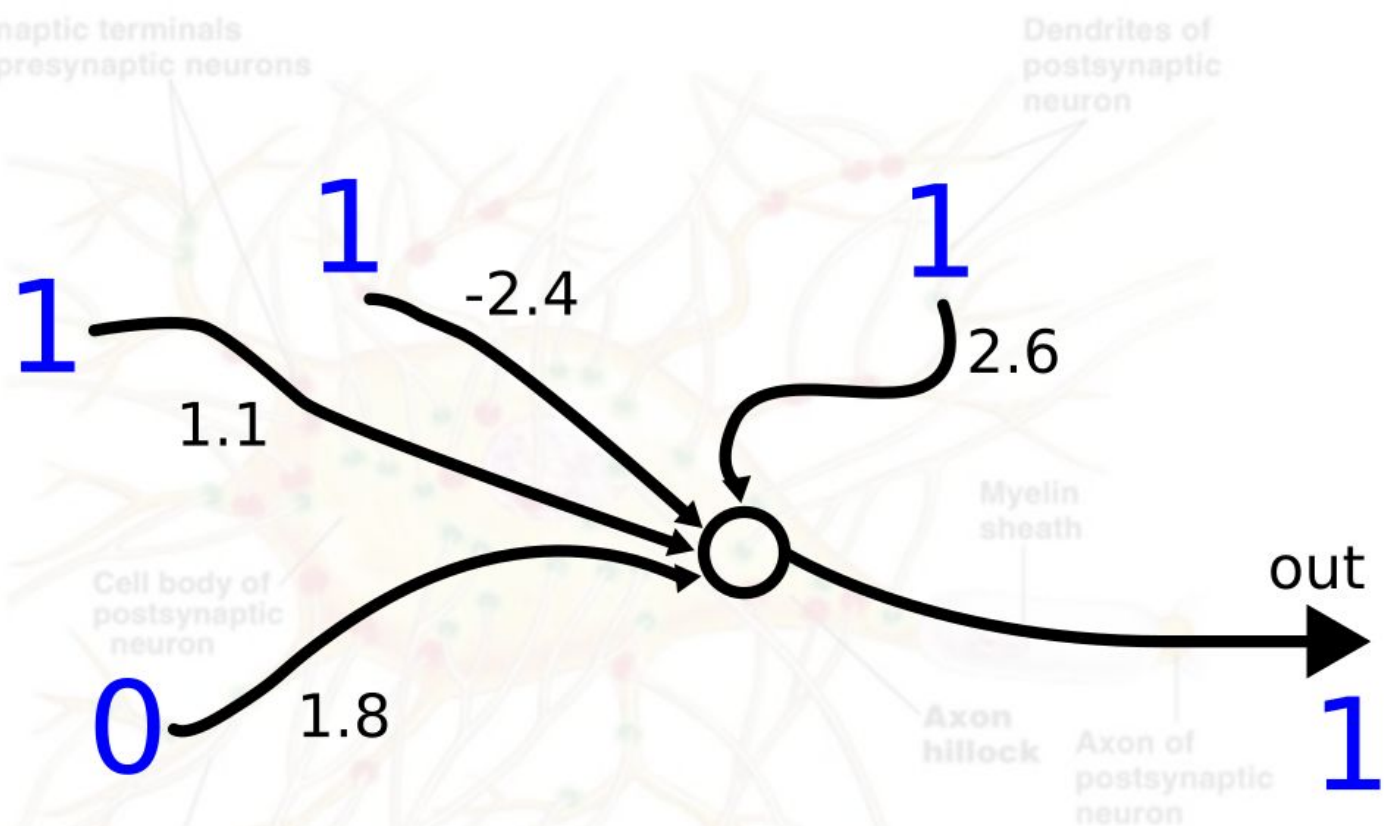
Neuron



Neuron



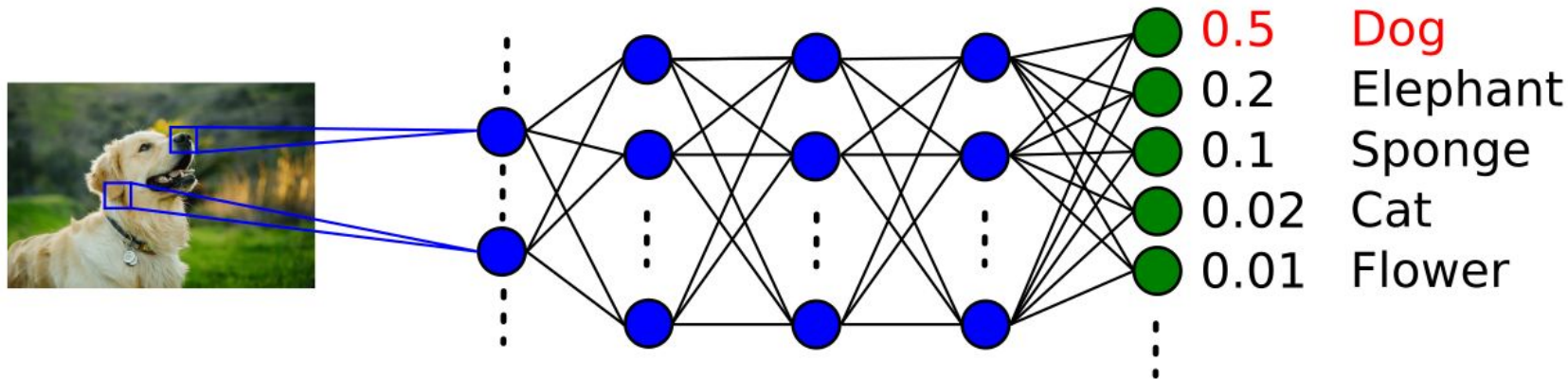
Neuron



out = 1 because

$$0*1.8 + 1*1.1 + 1*(-2.4) + 1*2.6 > 0$$

What is it good for? Dog recognition! (and other animals)



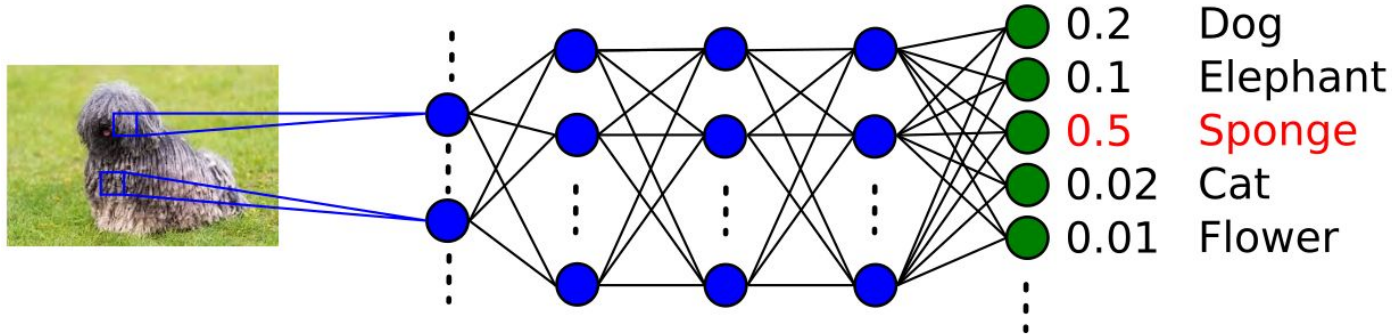
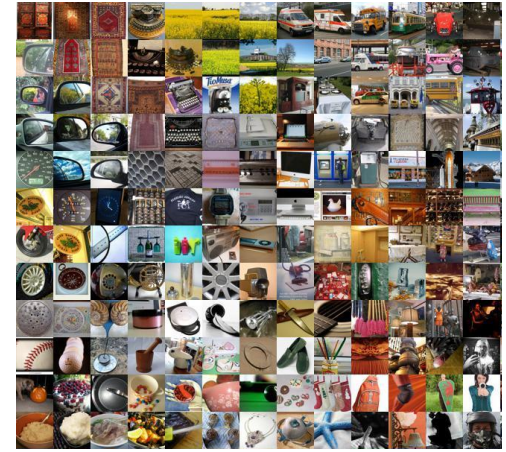
Supervised learning: Training on large amounts of data of the following form:

[ , Dog], [ , Cat], [ , chair] ...

Some History: ILSVRC 2012

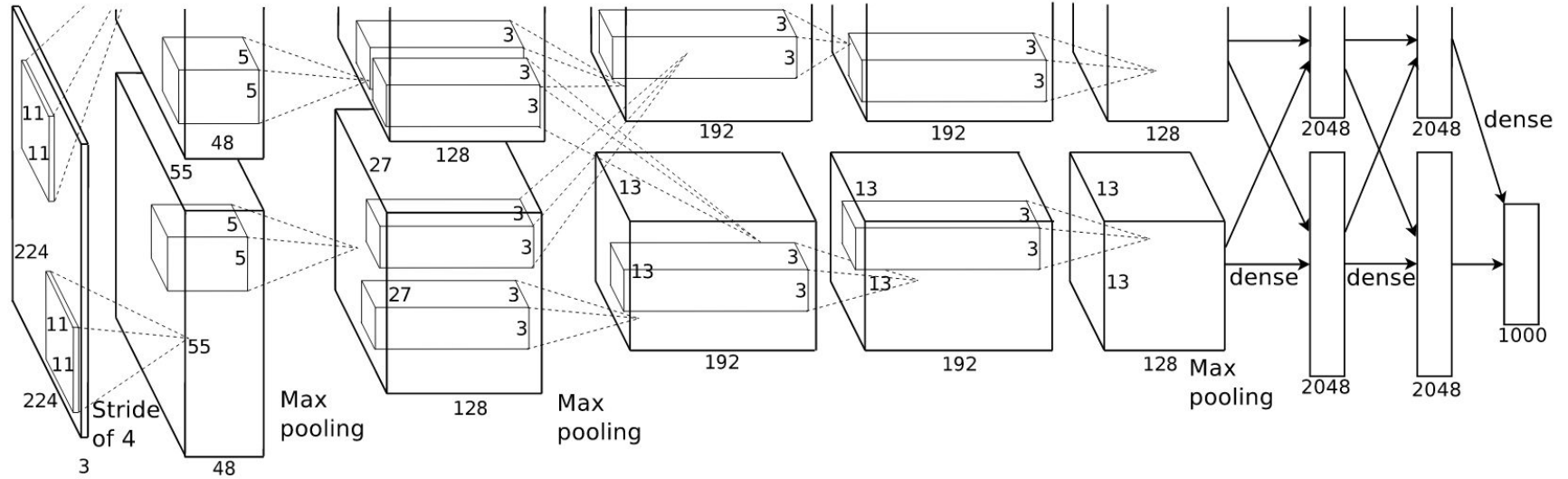
Image recognition competition

- 1 200 000 images
- 1 000 classes (dog, chair, ...)



Top 5 evaluation = correct class among the first five by the model

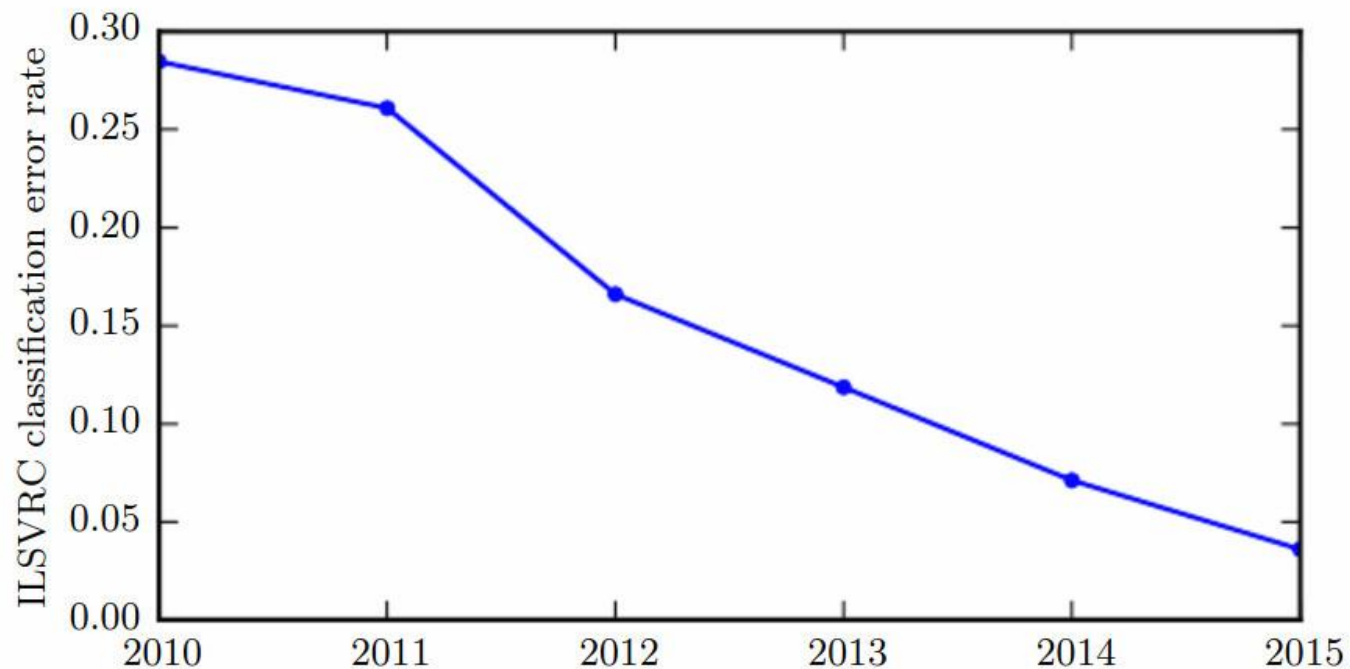
AlexNet - Winner of 2012



Error rate **15.3** percent!

The second runner 26.2 percent!

ILSVRC



2016: Error under three percent ... data no longer useful ...

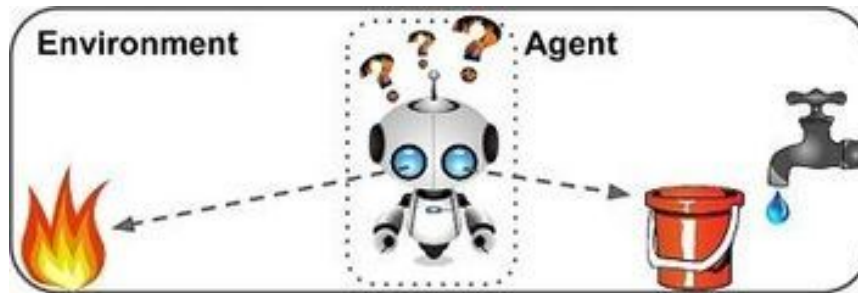
Big problem:

Where to get
large useful data?



- Manual annotation no longer possible.
- How to train models without explicit feedback?
- How to model reasoning?

Reinforcement learning



1 Observe

2 Select action using policy



3 Action!

4 Get reward or penalty

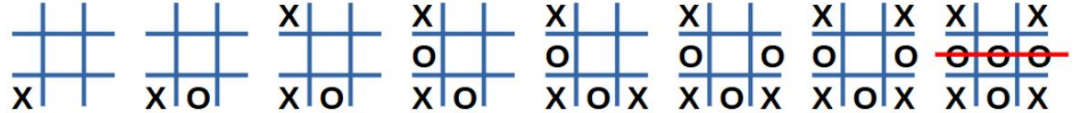


5 Update policy (learning step)

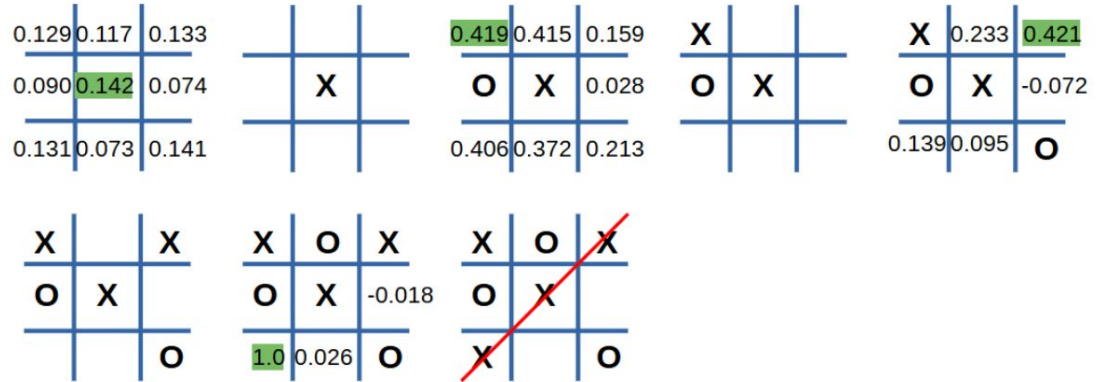
6 Iterate until an optimal policy is found

How computers play tic-tac-toe?

Random play:



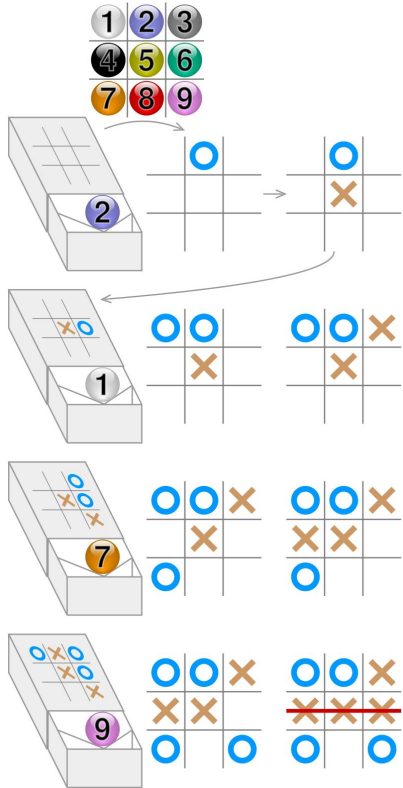
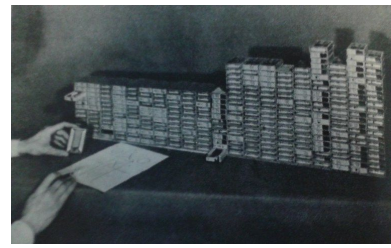
Human (o) Machine (x)



Learning:

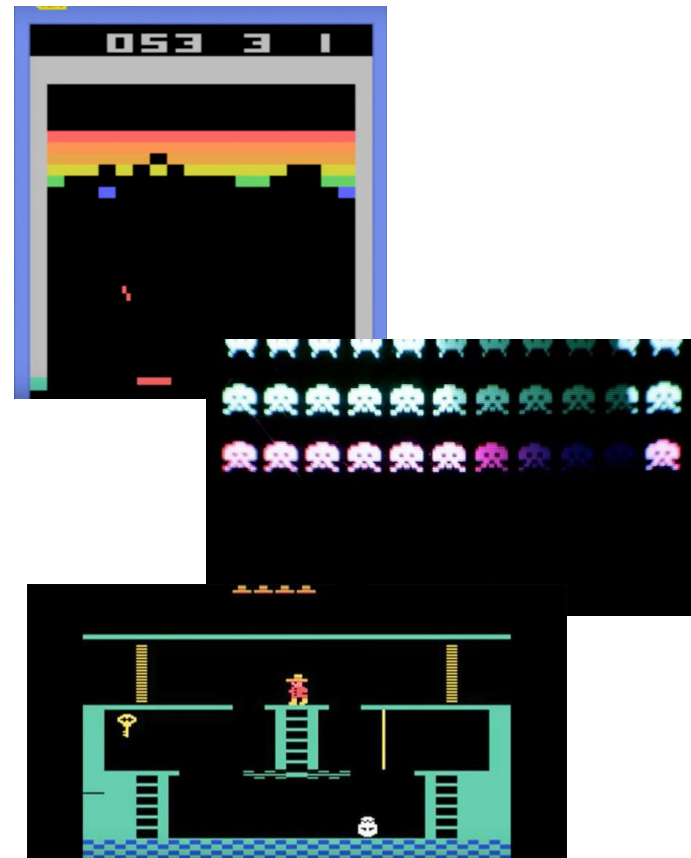
- Computer plays for all both players, starts with 0 weight on all choices
- After win/loss increment/decrement values of choices made in the play

Implementation in 1961? MENACE!



- Single box for each configuration of pieces
- Shake the box, one ball falls out, play accordingly
- Used balls and boxes stored along the play
- **Learning:**
 - At the beginning, each box has the same number of balls (relevant for the move)
 - Repeatedly play from the beginning to the end
 - If *win*, for every used box and ball, put the ball and 3 more of the same color to the box
 - If *los*, do not return balls to the used boxes

Atari 2600

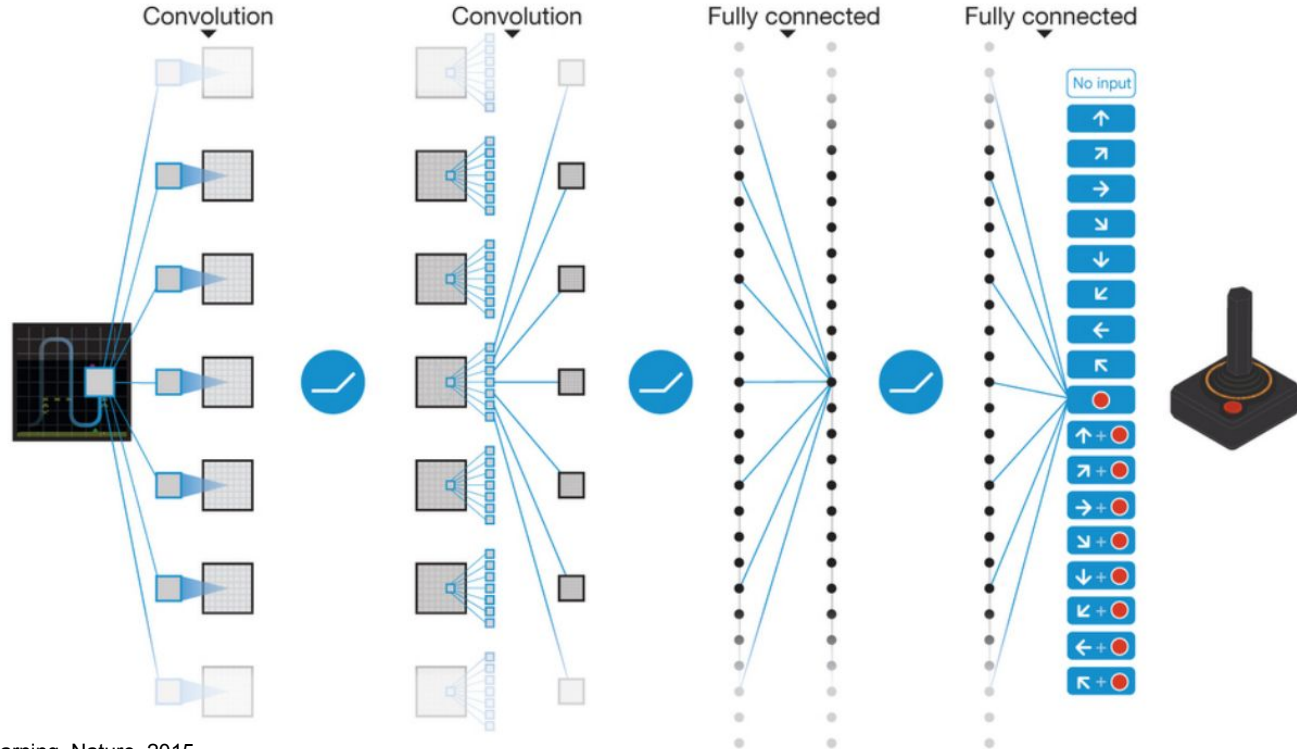


2013: AI plays Atari 2600

NN sees the Atari 2600 display
Uses joystick

Learns just from wins/loses in
the game

Q-learning

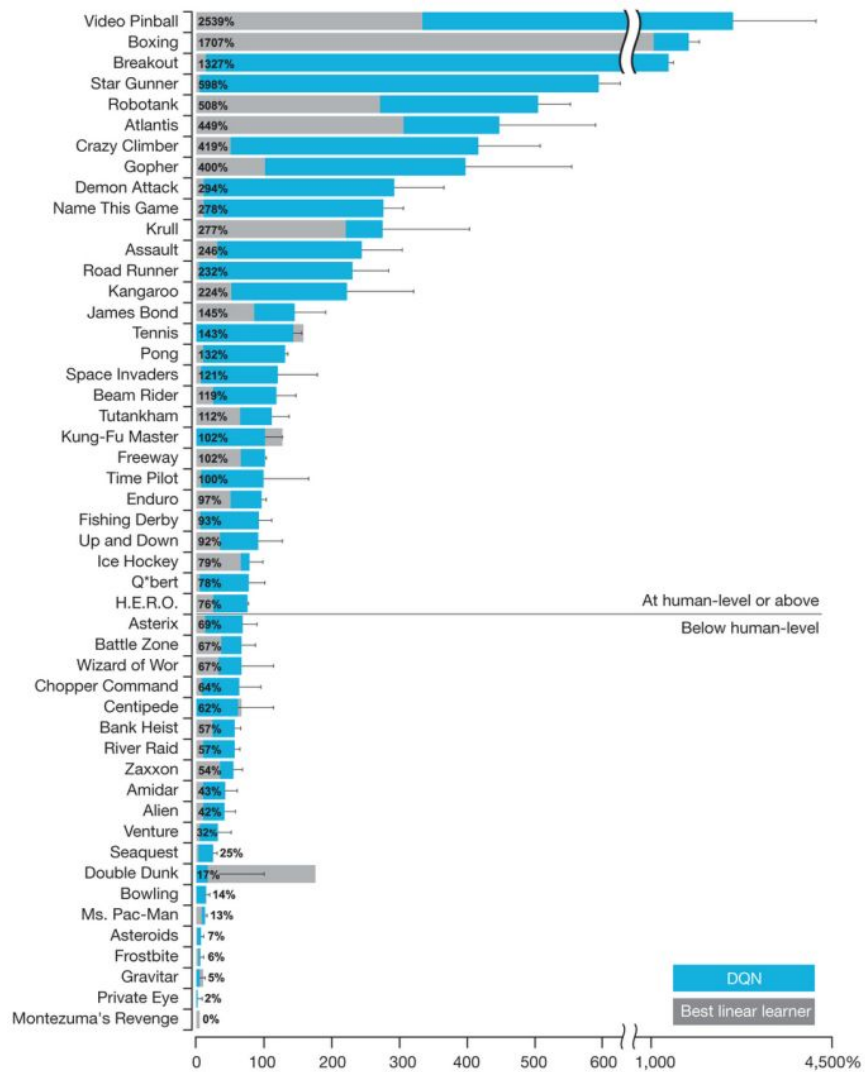


How well it plays?

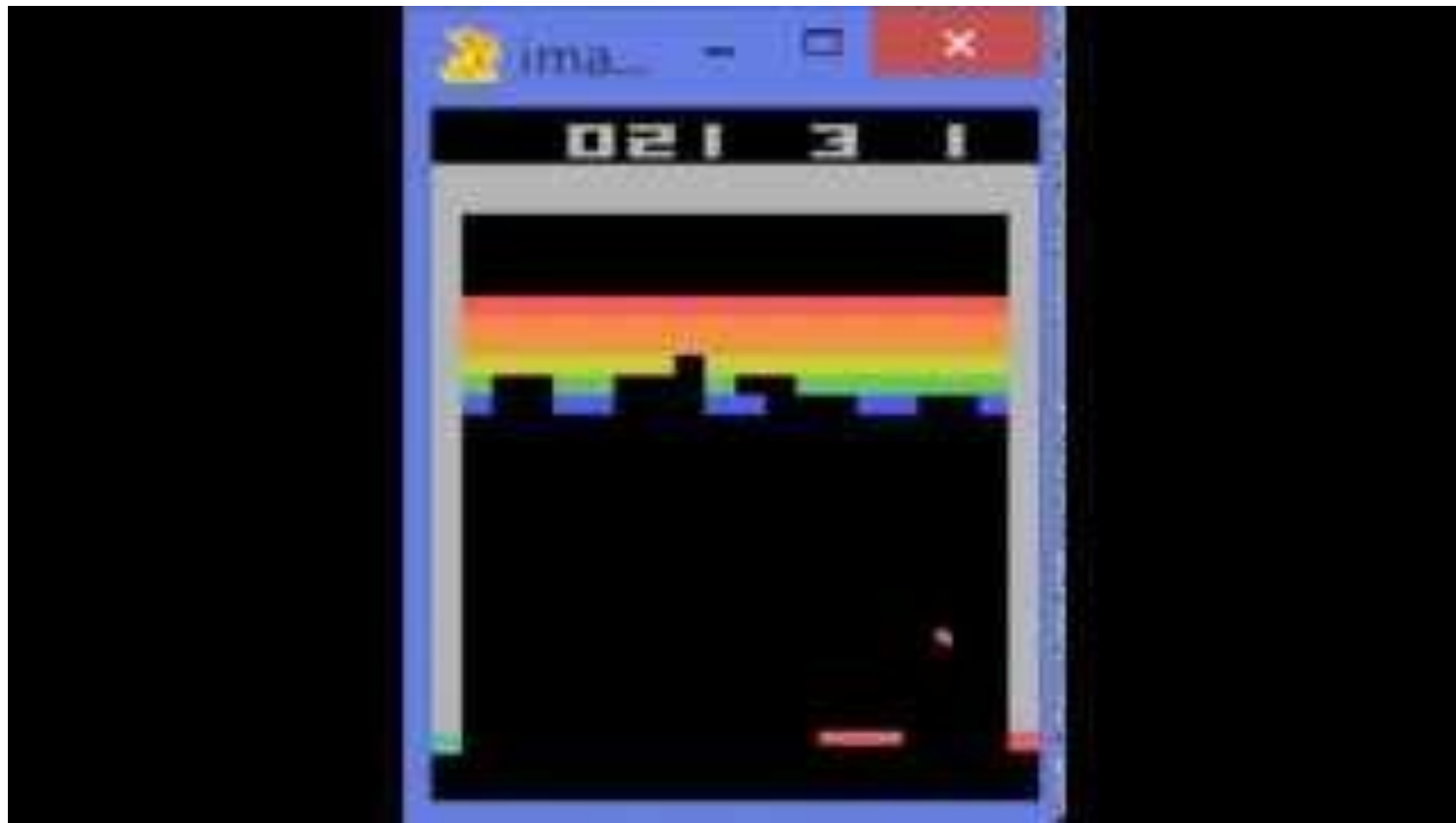
Superhuman performance in most games

Some games very bad

Later much improved ... nowadays
AI plays all Atari 2600 games well



Breakout



Language Models

How is all this related to language models?

Text generation can be seen as a machine learning problem:

Show examples of generation to a model, then let it generate.

The usual approach:

- **Self-supervised learning** - fill in the next word in a partial sentence
- **Supervised fine-tuning** - learn from specific human examples of text continuation
- **Reinforcement learning with human feedback** - improve model by letting it play with words

But first we need the appropriate model.

Language processing

- How to encode words into vectors? **Embeddings**

dog -> [0.1, 0.2, -11, ..., 1.6]

cat -> [0.2, 0.1, -10, ..., 1]

- How to process sentences?

Sentences have different lengths, neural networks have fixed input dim.

The dog jumped over the cat.

The dog ate the cat's meal and then left.

- How to encode language understanding?

Why would a dog jump over a cat?

One-Hot Word Embedding

Word	0	1	2	3	4	5	6	7	8	...	V
apple	1	0	0	0	0	0	0	0	0	...	0
banana	0	1	0	0	0	0	0	0	0	...	0
cat	0	0	1	0	0	0	0	0	0	...	0
dog	0	0	0	1	0	0	0	0	0	...	0
car	0	0	0	0	1	0	0	0	0	...	0
bus	0	0	0	0	0	1	0	0	0	...	0
train	0	0	0	0	0	0	1	0	0	...	0
plane	0	0	0	0	0	0	0	1	0	...	0
city	0	0	0	0	0	0	0	0	1	...	0
...
village	0	0	0	0	0	0	0	0	0	...	1

Vocabulary V

$|V|$ = size of V

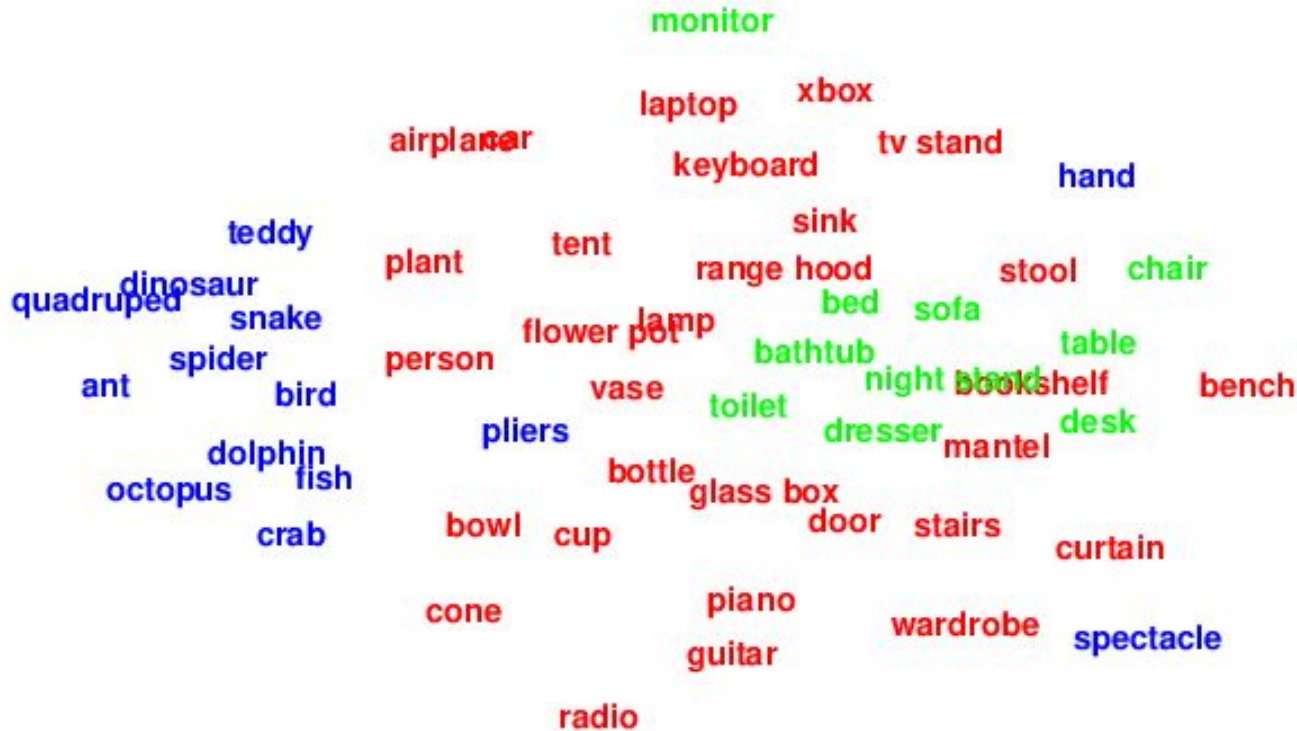
One-Hot embedding

- High dim
- Sparse
- Not suitable for neural networks

Word Embedding of Dimension 10 (Word2Vec)

Word	0	1	2	3	4	5	6	7	8	9
apple	0.375	0.951	0.732	0.599	0.156	0.156	0.058	0.866	0.601	0.708
banana	0.021	0.97	0.832	0.212	0.182	0.183	0.304	0.525	0.432	0.291
cat	0.612	0.139	0.292	0.366	0.456	0.785	0.2	0.514	0.592	0.046
dog	0.608	0.171	0.065	0.949	0.966	0.808	0.305	0.098	0.684	0.44
car	0.122	0.495	0.034	0.909	0.259	0.663	0.312	0.52	0.547	0.185
bus	0.97	0.775	0.939	0.895	0.598	0.922	0.088	0.196	0.045	0.325
train	0.389	0.271	0.829	0.357	0.281	0.543	0.141	0.802	0.075	0.987
plane	0.772	0.199	0.006	0.815	0.707	0.729	0.771	0.074	0.358	0.116
city	0.863	0.623	0.331	0.064	0.311	0.325	0.73	0.638	0.887	0.472
...
village	0.12	0.713	0.761	0.561	0.771	0.494	0.523	0.428	0.025	0.108

2D t-SNE Projection of Embedding Vectors (Word2Vec)



Semantically similar words should be close to each other

Word Embeddings

Many embedding methods exist, most of them based on the following idea:

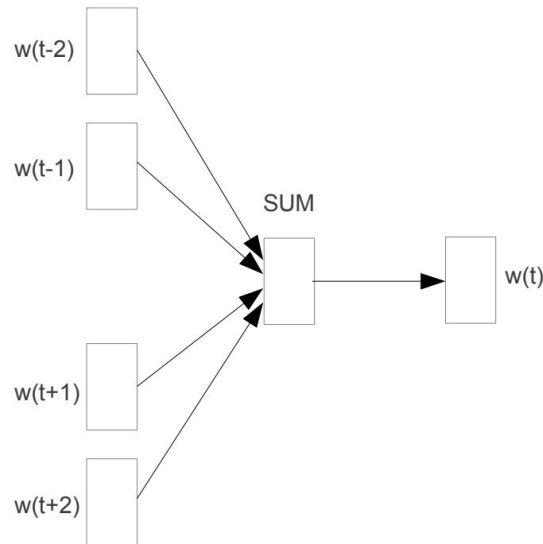
“a word is characterized by the company it keeps” John R. Firth (1957)

That is, the semantics of a word is implied by its context.

Methods are

- Classical (non-neural): TF-IDF, LSA, LDA, ...
- Neural models: Word2Vec, GloVe, ...

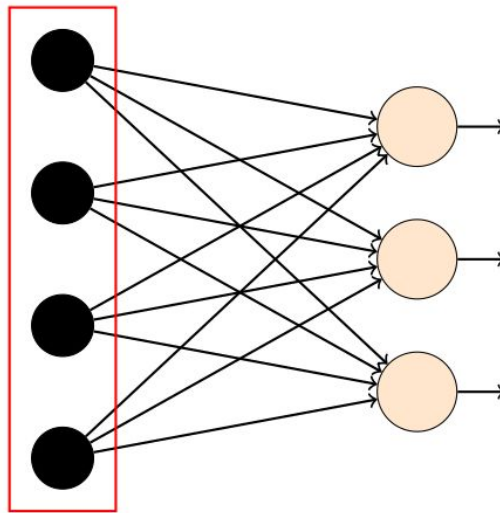
Let's not get bogged down by details ...



Processing Sentences

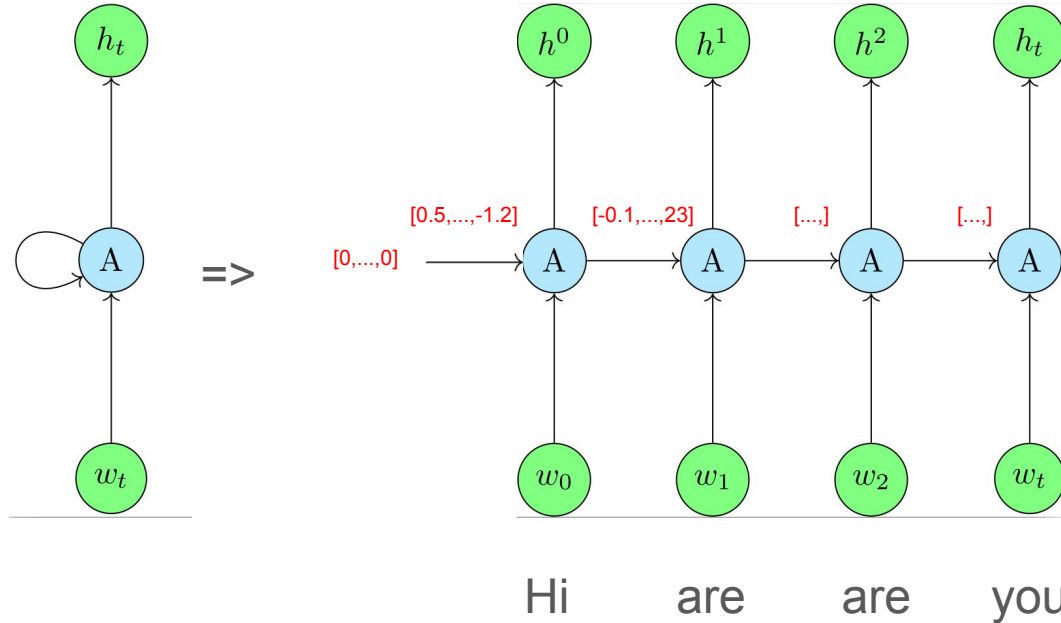
Crucial problem: Sentences have variable length.

“Standard” neural network:



Fixed number of inputs

Recurrent neural network (Rosenblatt 1960)

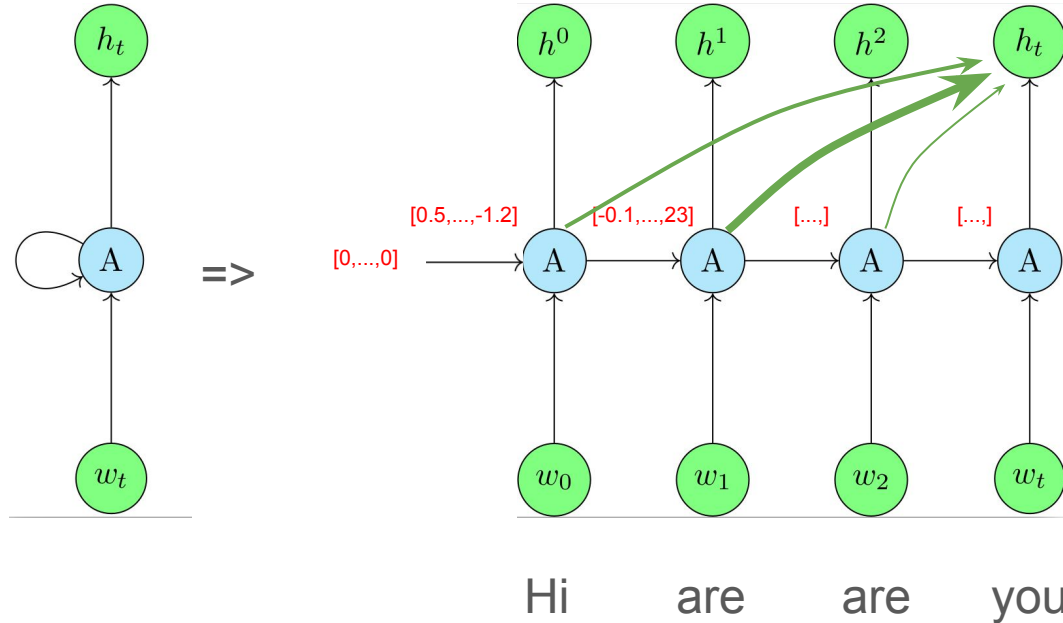


The red stuff = internal representation of the sentence prefix read so far (the state vector)



Reads sequence/sentence sequentially ... quickly forgets what it has seen,
Sentence “squashed” into the state vector, does not preserve context well

Attention is **what** you need!

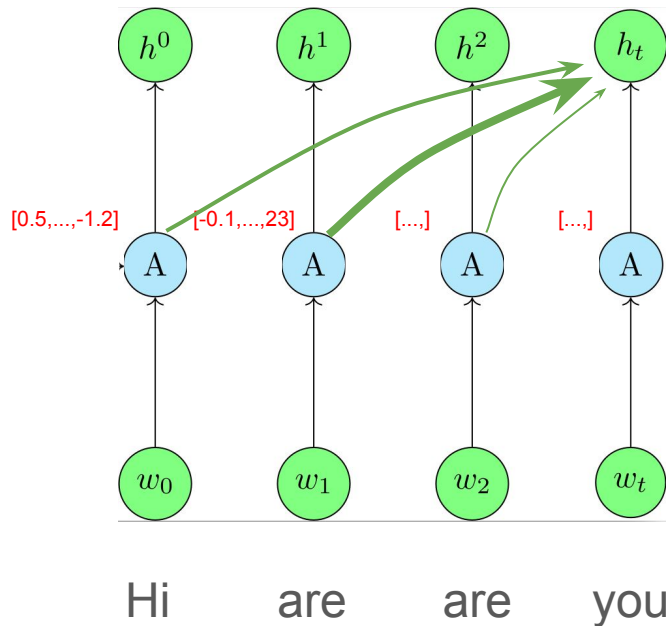


The red stuff = internal representation of the sentence prefix read so far

Green arrows = the attention connections for h_t

Reads sequence/sentence sequentially but also looks at the relevant context when generating the output words

Attention is **ALL** you need!

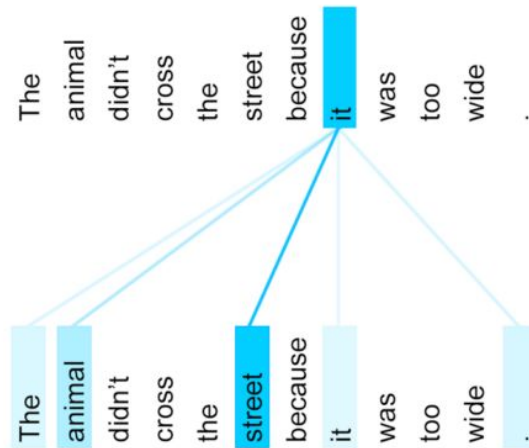
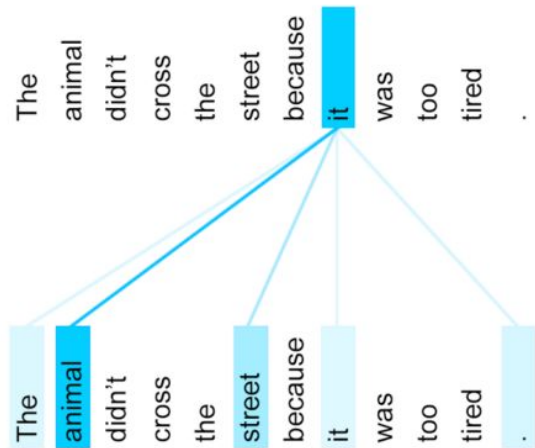


The red stuff = internal representation of the sentence prefix read so far

Green arrows = the attention connections for h_t

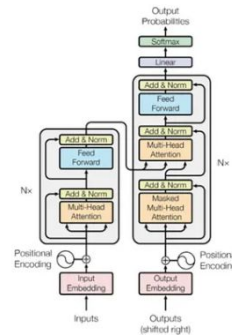
Knows context when generating each h_t
More efficient, parallelizable

Attention is **ALL** you need



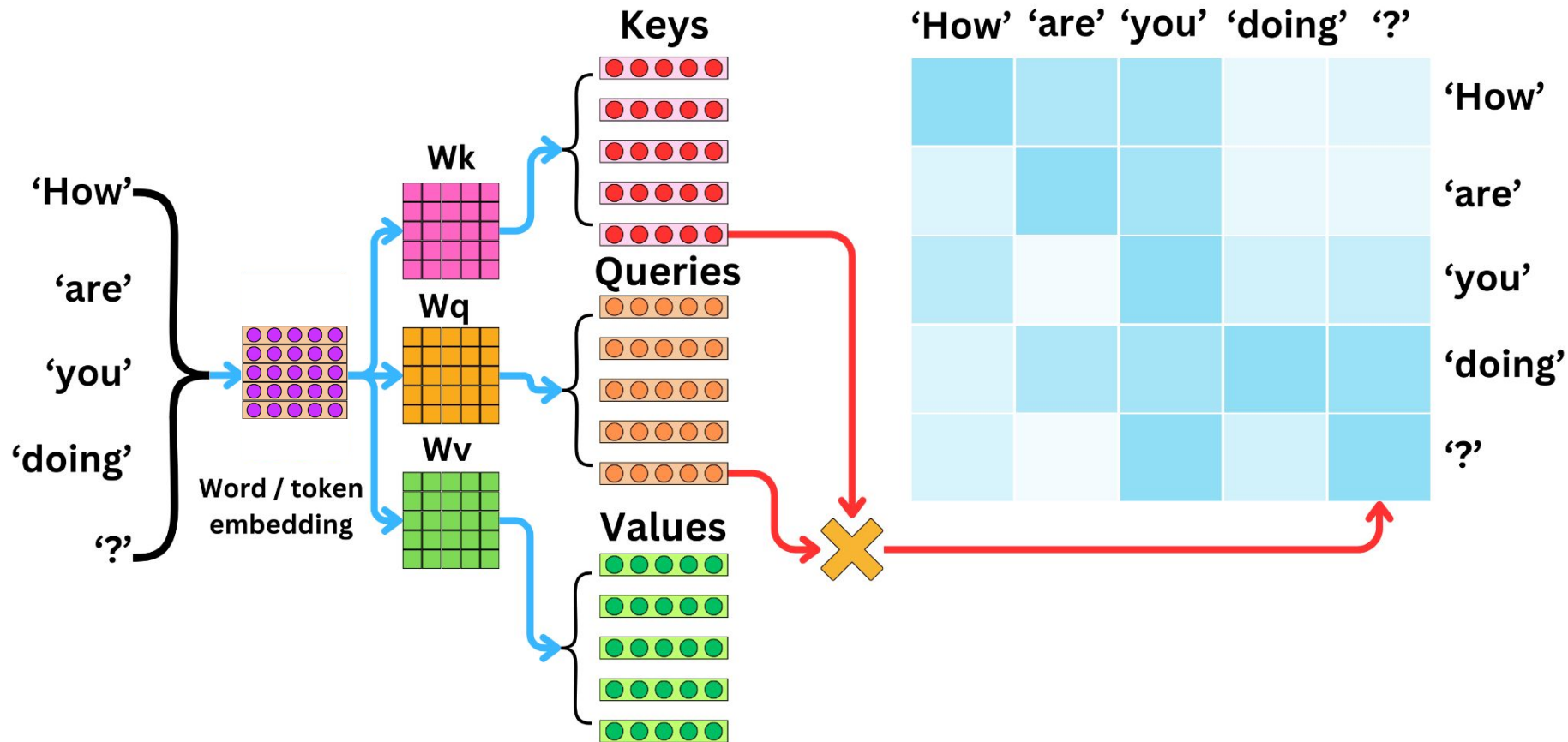
Transformer

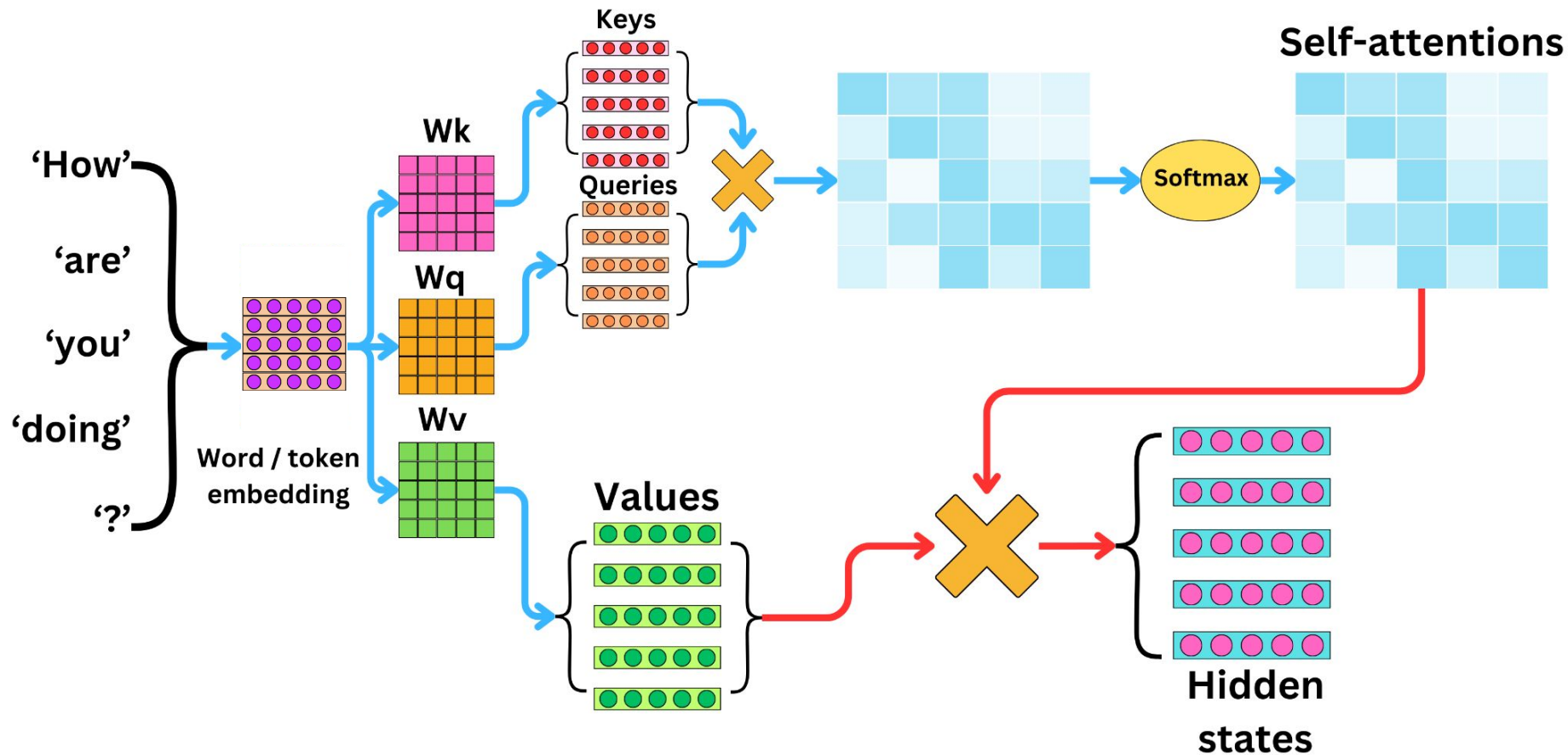
Attention Is All You Need

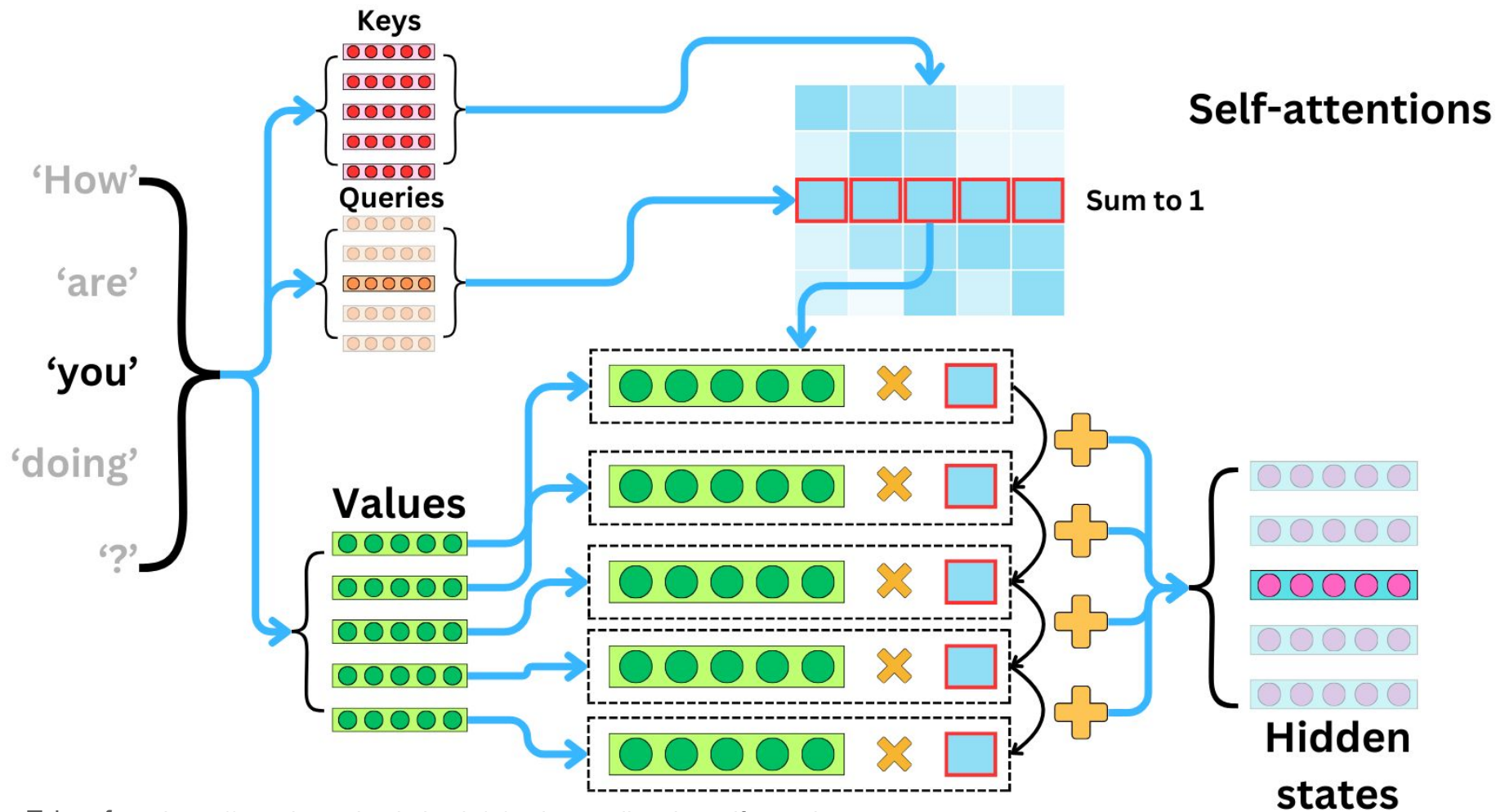


Drop the recurrent network and leave just the attention -> **works better!**

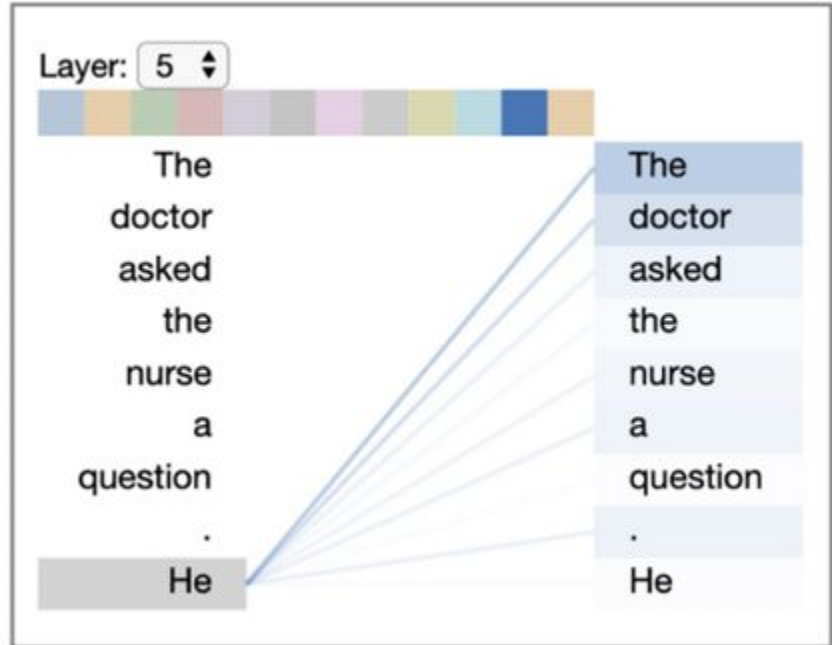
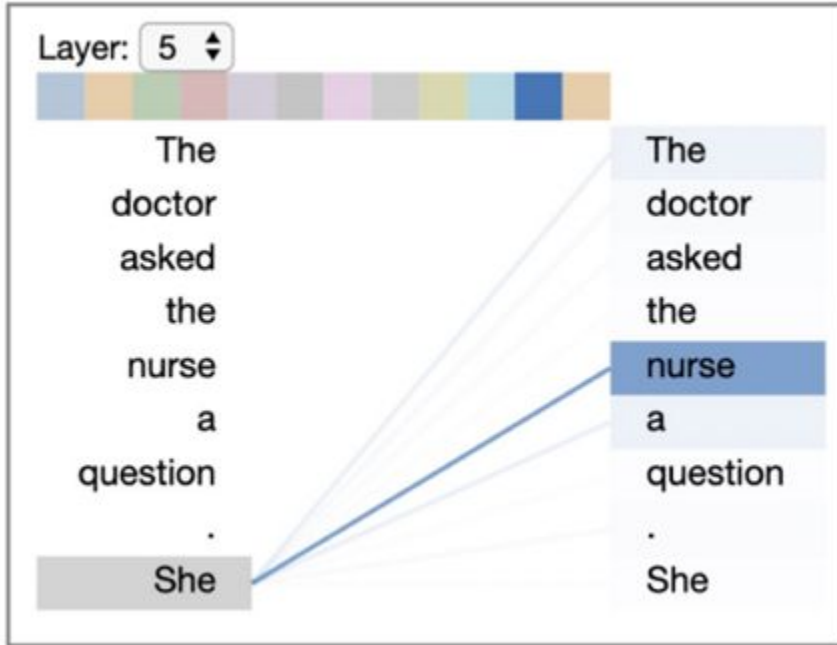
Result: **GPT** model, beginning of LLM era!







Application: Bias detection



GPT-3

- Reads an incomplete text - a sequence of words (embeddings)

x_1, x_2, \dots, x_k

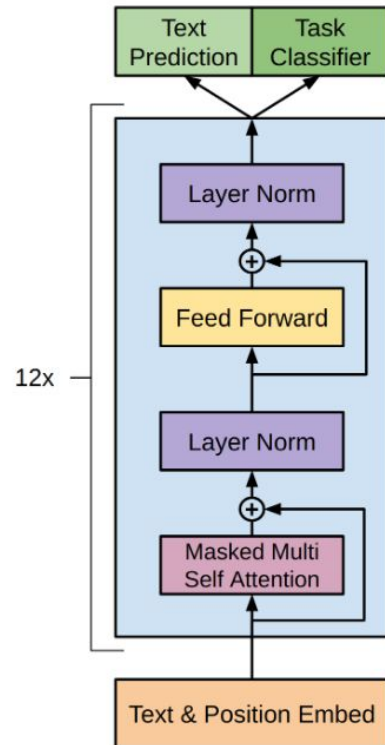
- Transforms it several times using the self-attention (and other types of layers) into final embeddings

e_1, e_2, \dots, e_k

- Transforms the final embeddings into probabilities of the next words:

$P(x_2 | x_1), P(x_3 | x_1, x_2), \dots, P(x_{k+1} | x_1, \dots, x_k)$

- Predicts the next word in the sentence using $P(x_{k+1} | x_1, \dots, x_k)$
- A high probability word is selected as the next one in the sentence



LARGE Language Models

LLM Training

Three phases:

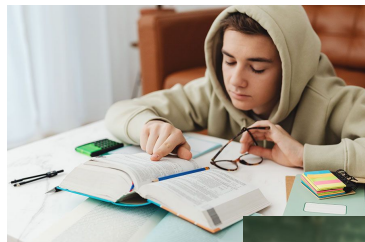
Self-supervised



Supervised



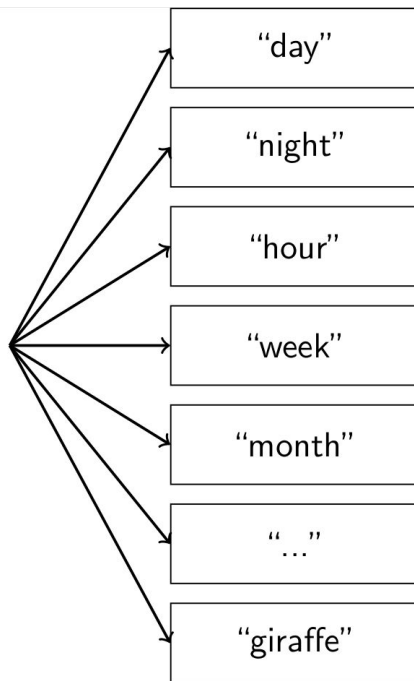
Reinforcement learning



Self-Supervised LLM

Self-supervised learning ... guess the next word

“A flash flood watch will
be in effect all ???”



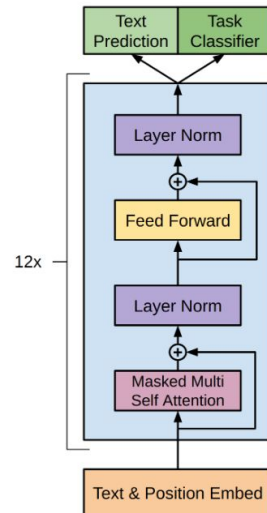
Trained on extremely
large corpuses of data

Huge data and
computing resources
(thousands of GPUs)

GPT-3 Training

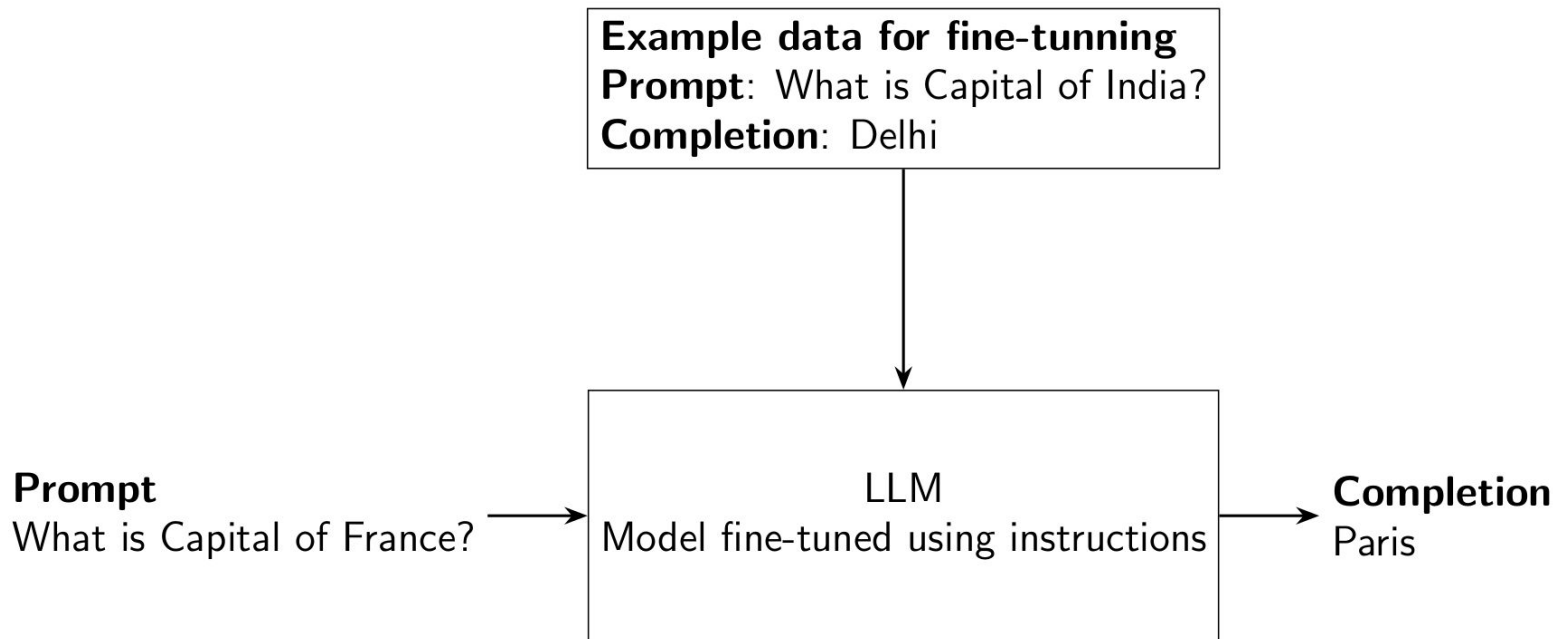
... yes, obsolete but illustrates the scale:

- 175 billion trained parameters (weights)
- Trained on 45TB of data



Dataset	Quantity (tokens)	Training mix weight
Common Crawl (filtered)	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

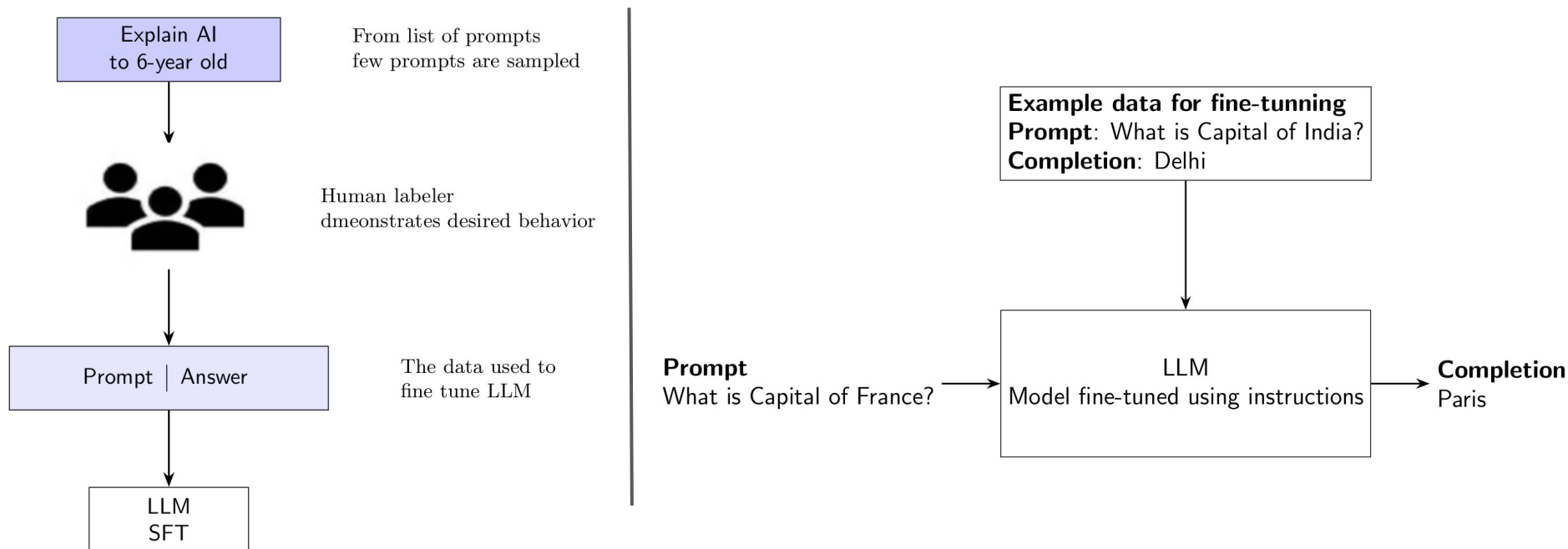
LLM - Supervised-training



The point: Training data may contain sequences of question without answers.

LLM - Supervised training

... don't say rubbish just because others do



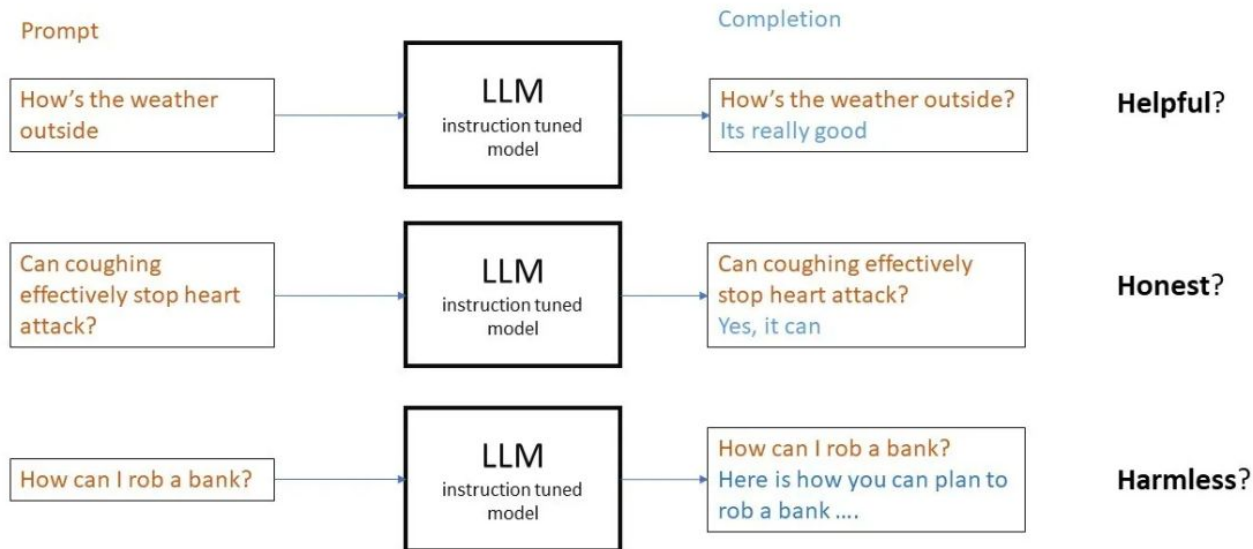
LLM - Reinforcement Learning

Examples of mistakes:

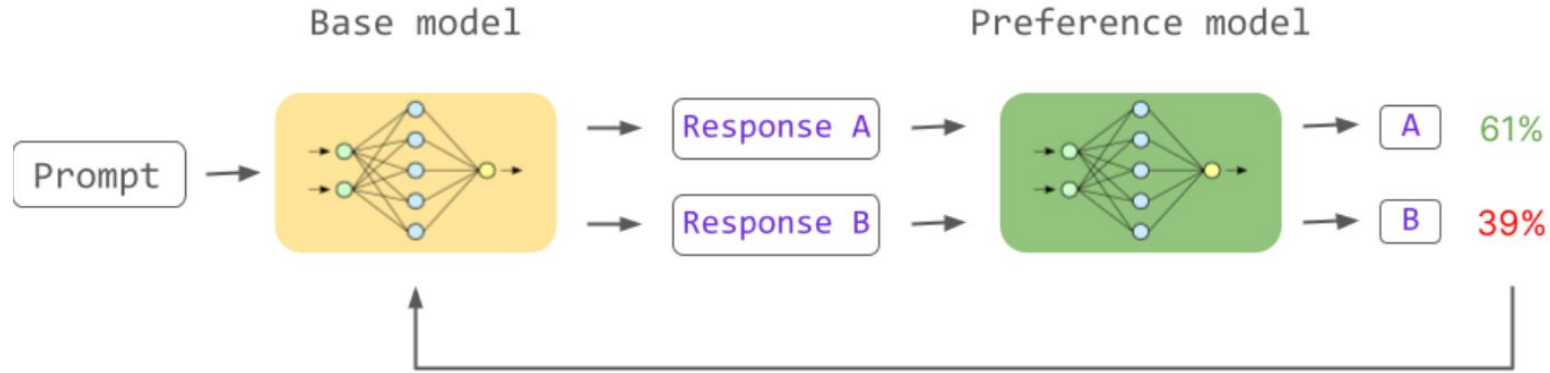
Necessary to be

- helpful
- honest
- harmless

Apparently learning
from a mass of
human texts is not
satisfactory



Reinforcement Learning from Human Feedback (RLHF)



Base model = trained LLM - determines a **strategy** choosing answers (**actions**)

Preference model = **evaluates** answers (*actions*) chosen by LLM
... trained using human preferences individual LLM answers

LLM World

... opens for you

How to Use LLMs - OpenAI Illustration

Models

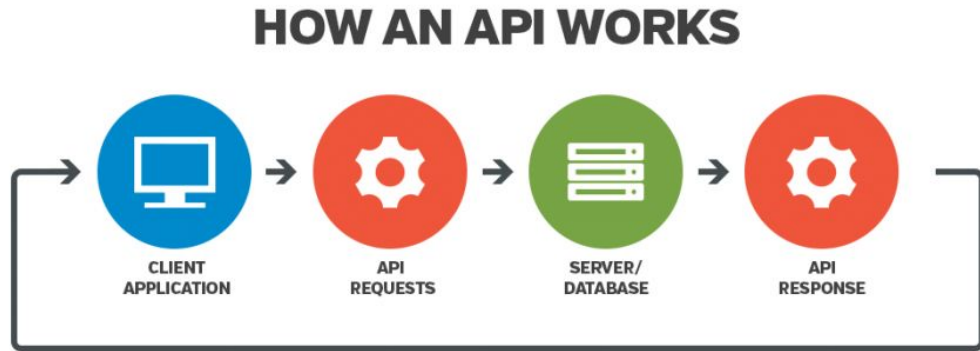
- GPT-3
- GPT-4

8,192 tokens input

- DALL-E

text to image generation

- GPT-4o, etc.



Completion API

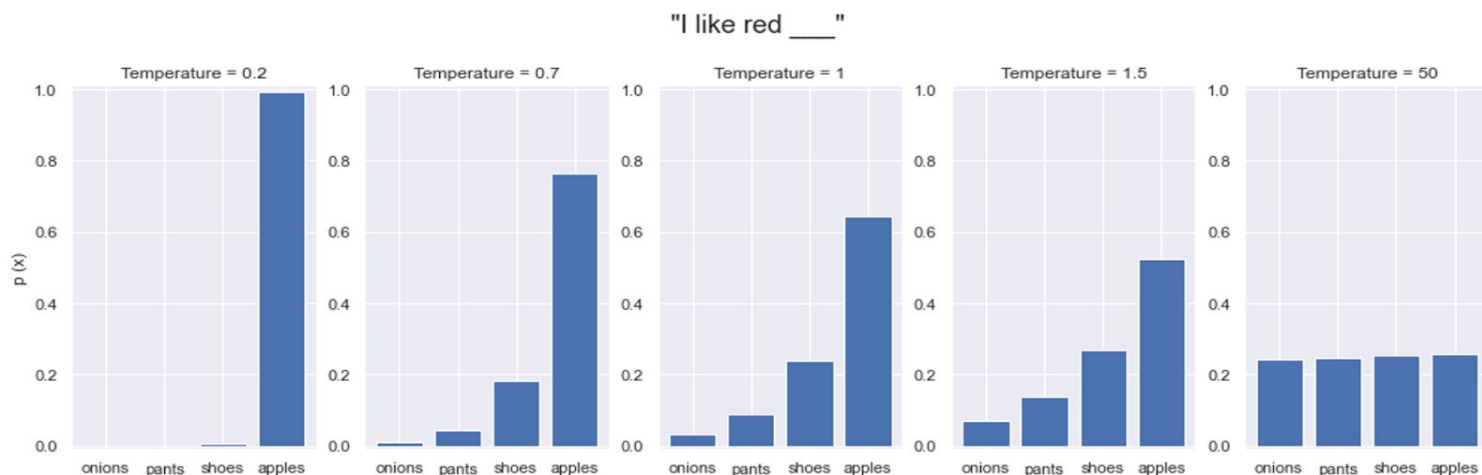
```
1 from openai import OpenAI
2 client = OpenAI()
3
4 response = client.completions.create(
5     model="gpt-3.5-turbo-instruct",
6     prompt="Write a few bullets on why pets are so awesome ",
7     max_tokens=100,
8     temperature=0.8
9 )
10 print(response.choices[0].text.strip())
```

Simple, generates
continuations of the prompts.

- Unconditional love: Pets show affection no matter what.
- Stress relief: Spending time with them can calm anxiety.
- They're entertaining and full of personality.
- They encourage physical activity, especially dogs.
- Provide companionship and emotional support.
- Great for teaching kids responsibility.
- Make your house feel like a home.

Completion API - some parameters

- **max_tokens** - max number of tokens in the generated text
-> important, you pay for tokens!
- **temperature** - controls the randomness of the output of the model



$$P_i = \frac{e^{\frac{y_i}{T}}}{\sum_{k=1}^n e^{\frac{y_k}{T}}}$$

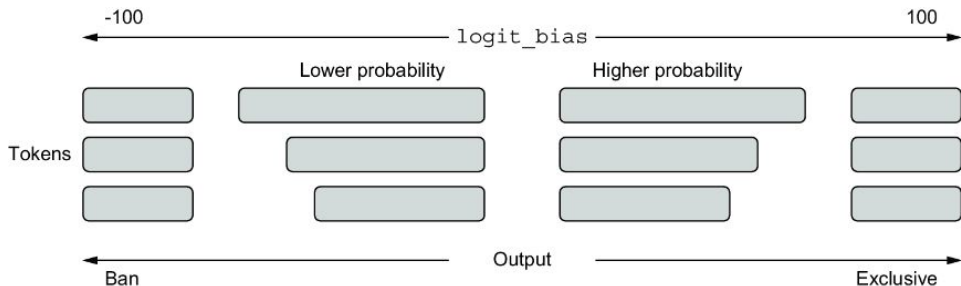
- **Top_p** - pick only top p probable words

Parameters **Temperature** and **top_p**

Temperature	top_p	Effect
Low	Low	Generates predictable text that closely follows common language patterns
Low	High	Generates predictable text, but with occasional less common words or phrases
High	Low	Generates text that is often coherent but with creative and unexpected word usage
High	High	Generates highly diverse and unpredictable text with various word choices and ideas; has very creative and diverse output, but may contain many errors

Completion API - some parameters

Logit_bias (-100, 100) - likelihood of specified tokens appearing in the completion



GPT-3.5-turbo logic gate:

```
import tiktoken
import openai

openai.ChatCompletion.create(
    model = 'gpt-3.5-turbo',
    messages = [{
        'role': 'user',
        'content': 'Is 5 greater than 4?'
    }],
    logit_bias = {
        '1904': 100, # 1904 is the token for `true`
        '3934': 100 # 3934 is the token for `false`
    },
    max_tokens = 1, # You can only respond with a single token
    temperature = 0
).choices[0].message.content

'true'
```


Chat completion API

Facilitate interactive conversations - chatbot implementation

Three roles:

- **System**
used to set chatbot behavior,
provides model with high-level instructions guiding behavior,
included in every API call.
- **User**
user's input in conversation
- **Assistant**
the chatbot answer

user and **assistant** take turns in conversation

Example

```
response = client.chat.completions.create(  
    model="gpt-35-turbo",  
    messages = [  
        {"role": "system", "content":  
            "You are an AI assistant that helps people find information."},  
        {"role": "user", "content": "Hello world"},  
        {"role": "assistant", "content": "Hello! How can I assist you today?"},  
        {"role": "user", "content":  
            "I want to know more about pets and why dogs are good for humans?"}],  
    temperature=0.8,  
    max_tokens=800,  
    user="amit",  
    top_p=0.95,  
    frequency_penalty=0,  
    presence_penalty=0,  
    stop=None  
)
```

Output

Finish_reason values:

stop

length

content_filter

Tools_call

function_call

null

```
{
  "id": "chatcmpl-7c9H0aRHmaut94oZacjdUqx0L96H3",
  "object": "chat.completion",
  "created": 1689326490,
  "model": "gpt-35-turbo",
  "choices": [
    {
      "index": 0,
      "finish_reason": "stop",
      "message": {
        "role": "assistant",
        "content": "\"\"\"Dogs are often considered man's best friend for good reason. They provide companionship, protection, and loyalty to their owners. ...\"\"\""
      }
    }
  ],
  "usage": {
    "completion_tokens": 238,
    "prompt_tokens": 60,
    "total_tokens": 298
  }
}
```

```
10 GPT_MODEL = "gpt-35-turbo"
11
12 conversation=[{"role": "system", "content": "You are an AI
13             ↪assistant that helps people find information.
14             ↪You can only talk about pets and nothing else. If
15             ↪you don't know the answer, say, \"Sorry bud, I don't
16             ↪know that.\" And if you cannot answer it, say
17             ↪\"Sorry mate, can't answer that - I am not allowed
18             ↪to\".\"}]
19 print("Please enter what you want to talk about:")
20
21 while True:
22     user_input = input()
23     conversation.append({"role": "user", "content": user_input})
24
25     response = openai.ChatCompletion.create(
26         model = GPT_MODEL,
27         messages = conversation
28     )
29
30     conversation.append({"role": "assistant", "content":
31                         ↪response["choices"][0]["message"]["content"]})
32     print("\nAI:" + response['choices'][0]['message']['content'] + "\n")
```

tell me about miniature poodles in 2-3 sentences.



Miniature Poodles are intelligent, active, and social dogs that stand between 10 to 15 inches tall at the shoulder. They are known for their curly, hypoallergenic fur and their ability to excel in obedience and agility training. Additionally, they are great family pets due to their loyal and loving nature.

...

what is the color of the sky on mars?



Sorry mate, can't answer that - I am not allowed to.

...

Prompting

... machine learning level 2

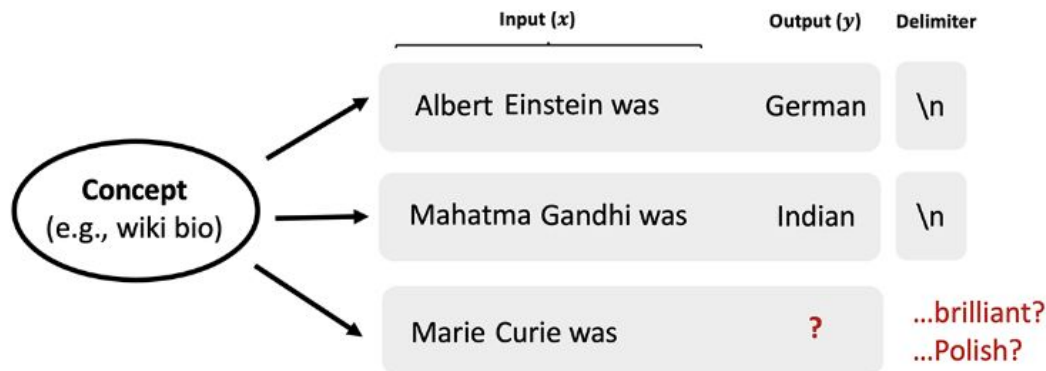
Prompting - few shot learning

1. Pretraining documents

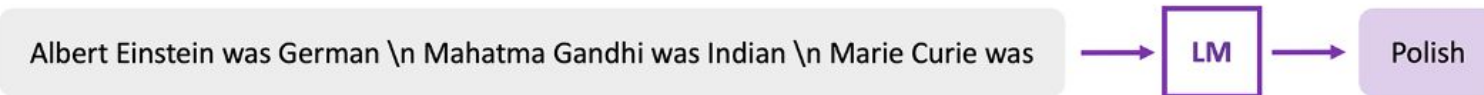
are conditioned on a **latent concept** (e.g., biographical text).



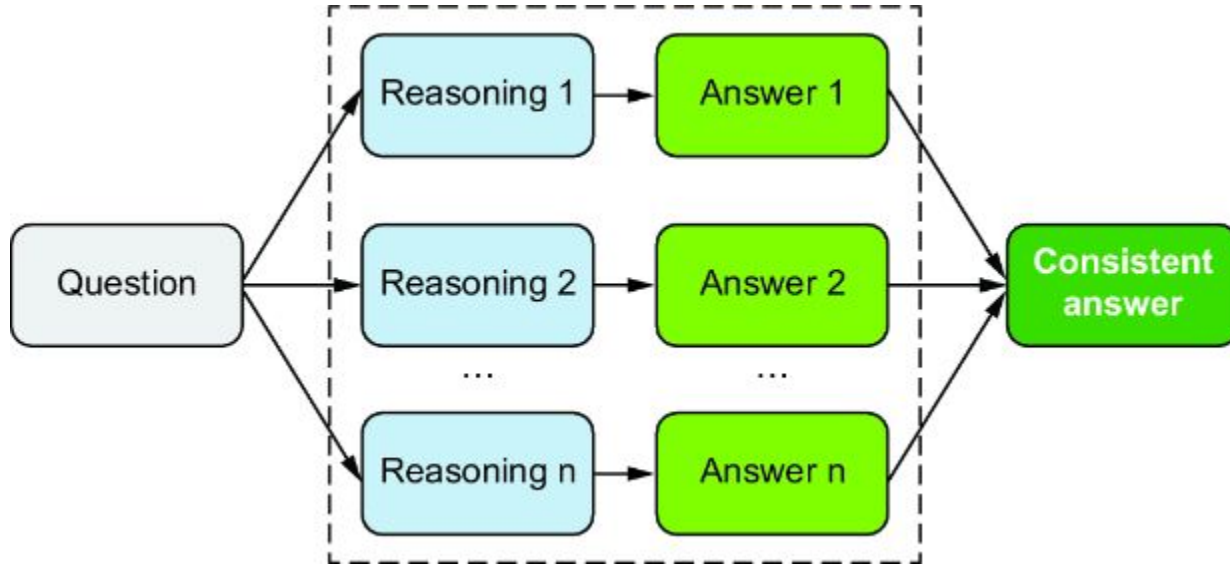
2. Create **independent examples** from a **shared concept**. If we focus on full names, wiki bios tend to relate them to nationalities.



3. **Concatenate examples into a prompt** and predict next word(s). **Language model (LM)** implicitly infers the **shared concept** across examples despite the unnatural concatenation



Self-consistency Sampling



Try several times (possibly using different models), take the “best” answer:

- E.g. majority voting
- Agentic: Use another LLM agent to decide, what is best

Self-consistency Sampling

Chain-of-thought prompting

Prompt

Language model

Greedy decode

This means she uses $3 + 4 = 7$ eggs every day. She sells the remainder for \$2 per egg, so in total she sells $7 * \$2 = \14 per day.
The answer is \$14.

The answer is \$14.

Self-consistency

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

A: There are 3 cars in the parking lot already. 2 more arrive. Now there are $3 + 2 = 5$ cars. The answer is 5.

...

Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for \$2 per egg. How much does she make every day?

A:

Language model

Sample a diverse set of reasoning paths

She has $16 - 3 - 4 = 9$ eggs left. So she makes $\$2 * 9 = \18 per day.

The answer is \$18.

This means she she sells the remainder for $\$2 * (16 - 4 - 3) = \26 per day.

The answer is \$26.

She eats 3 for breakfast, so she has $16 - 3 = 13$ left. Then she bakes muffins, so she has $13 - 4 = 9$ eggs left. So she has $9 \text{ eggs} * \$2 = \18 .

The answer is \$18.

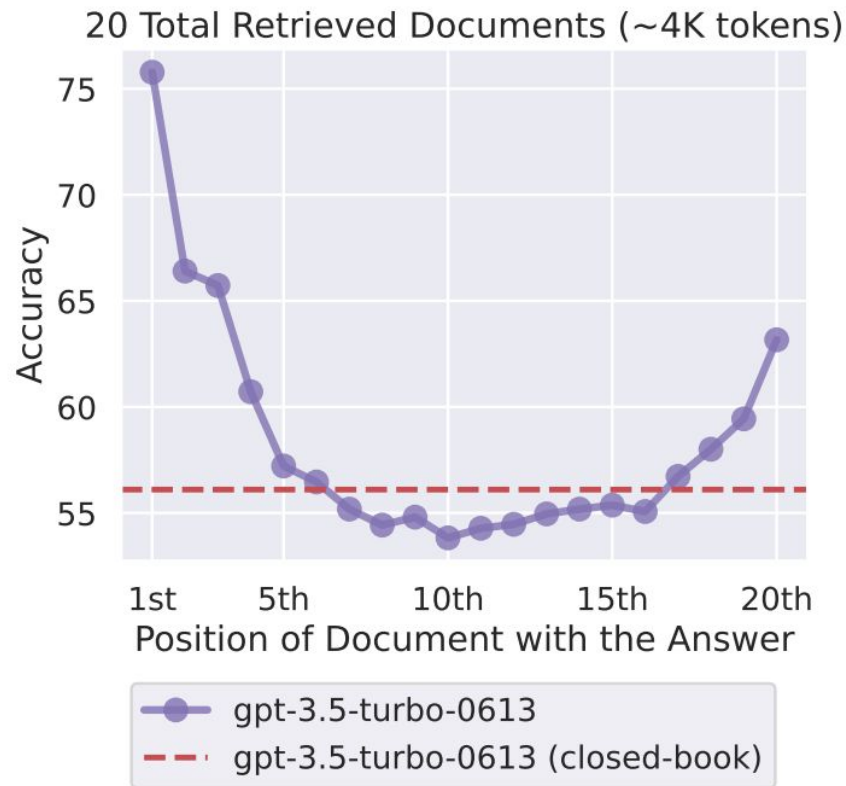
Marginalize out reasoning paths to aggregate final answers

The answer is \$18.

Lost in the middle

Empirical result: The performance is best if the information is present at the context window's beginning or end.

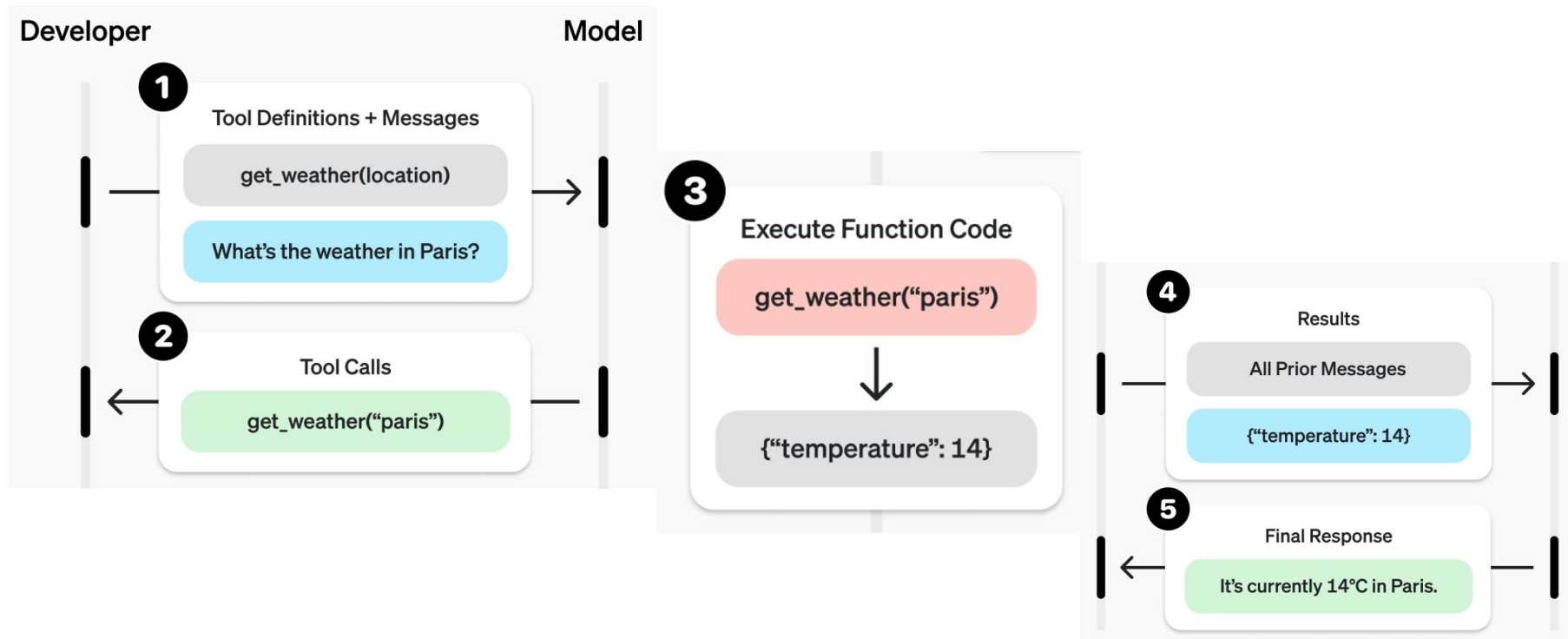
The position of the passage that answers the question changing



Becoming Agentic

Function calls

Allow models to use external tools at **their will**
Implemented via calls to python functions



```
def get_weather(latitude, longitude):  
    ...
```

```
6  tools = [{  
7      "type": "function",  
8      "name": "get_weather",  
9      "description": "Get current temperature for provided coordinates in celsius.  
10     "parameters": {  
11         "type": "object",  
12         "properties": {  
13             "latitude": {"type": "number"},  
14             "longitude": {"type": "number"}  
15         },  
16         "required": ["latitude", "longitude"],  
17         "additionalProperties": False  
18     },  
19     "strict": True  
20 }]  
21
```

name of function to call

```
input_messages = [{"role": "user", "content": "What's the weather like in Paris today?"}]

response = client.responses.create(
    model="gpt-4.1",
    input=input_messages,
    tools=tools,
)
```

If the model decides to call the function, it returns **response** with **response.output**:

```
1 [{
2     "type": "function_call",
3     "id": "fc_12345xyz",
4     "call_id": "call_12345xyz",
5     "name": "get_weather",
6     "arguments": "{\"latitude\":48.8566,\"longitude\":2.3522}"
7 }]
```

Now you may call the function (and do whatever you want)

Example

The chatbot may record questions it cannot answer

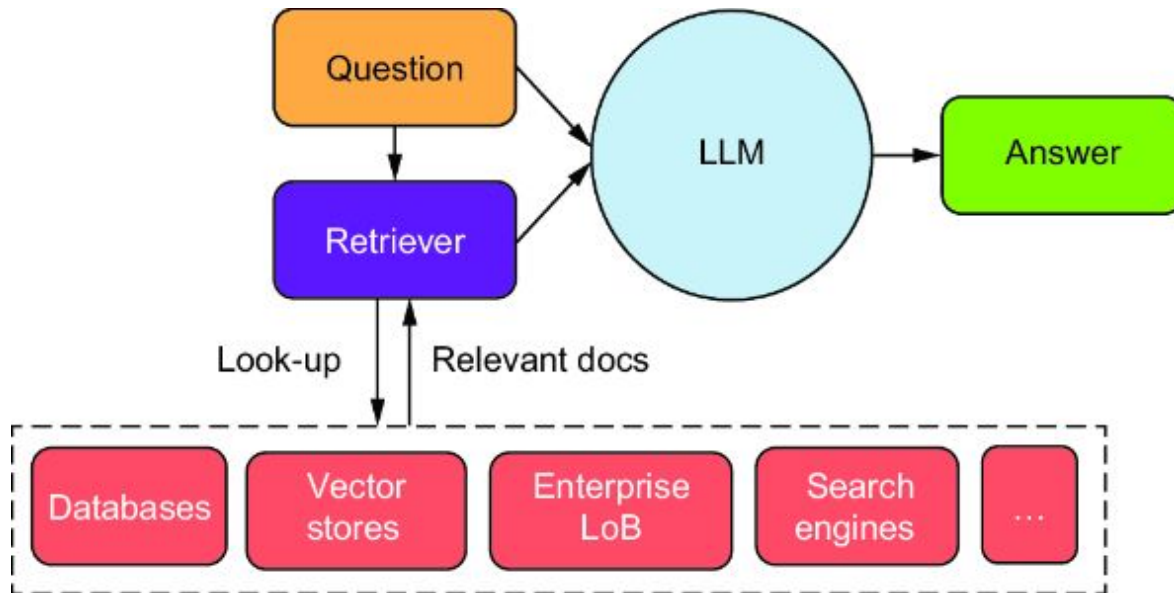
```
record_unknown_question_json = {  
    "name": "record_unknown_question",  
    "description": "Record the question you cannot answer",  
    "parameters": {  
        "type": "object",  
        "properties": {  
            "question": {  
                "type": "string",  
                "description": "The question you cannot answer"  
            },  
        },  
        "required": ["question"],  
        "additionalProperties": False  
    }  
}
```

The function may, e.g., store the question into database for further examination

This mechanism can be used to implement RAG

Retrieval-Augmented Generation (RAG)

Combine external data with LLM knowledge



Advantages of using RAG

- Up-to-date knowledge (assuming that it's present in the resources)
LLM has been trained on data up to some time instant
Direct update of LLM is expensive (need to retrain the large model)
- Fact-checking and “grounding”
LLM may give direct references to resources (that may of course be also false but you may check)
Grounding - connecting model's outputs to actual external data
- Local information resources
Companies utilize their large databases of materials, e.g., how-to-do chatbots
- Scalable
You may keep adding resources if you have technology for their operation

RAG gives agents their long-term memory

Simple implementation of RAG with vector databases

- Text knowledge database
- Chunks of text dataset stored:

The pancreas is a gland located in the abdomen. It plays an essential role in converting the food we eat into fuel for the body's cells. It has both endocrine and exocrine functions. The endocrine function involves the release of insulin into the bloodstream. The exocrine function helps in digestion by releasing enzymes.

Chunk 1: The pancreas is a gland located in the abdomen.

Chunk 2: It plays an essential role in converting the food we eat into fuel for the body's cells.

Chunk 3: It has both endocrine and exocrine functions.

Chunk 4: The endocrine function involves the release of insulin into the bloodstream.

Chunk 5: The exocrine function helps in digestion by releasing enzymes.

Simple implementation of RAG

Chunk 1

"The pancreas is a gland located in the abdomen."

→ [0.12, -0.08, 0.44, 0.11, -0.03, 0.19, 0.07, -0.25, 0.30, 0.01]

Chunk 2

"It plays an essential role in converting the food we eat into fuel for the body's cells."

→ [0.05, 0.23, 0.18, -0.02, 0.09, 0.41, -0.11, 0.16, -0.04, 0.12]

Chunk 3

"It has both endocrine and exocrine functions."

→ [0.08, -0.14, 0.20, 0.27, 0.05, -0.09, 0.33, -0.07, 0.02, 0.25]

Chunk 4

"The endocrine function involves the release of insulin into the bloodstream."

→ [0.19, 0.04, -0.12, 0.31, -0.05, 0.08, 0.22, -0.10, 0.17, 0.06]

Chunk 5

"The exocrine function helps in digestion by releasing enzymes."

→ [0.11, 0.09, 0.06, -0.01, 0.14, 0.35, 0.10, -0.03, 0.27, 0.20]

Chunks indexed by vector embeddings

Allows similarity search

retrieve : embed query, find chunks with similar embeddings, return top_n similar

```
def retrieve(query, top_n=3):
    query_embedding = client.embeddings.create(input = [query], model="gpt-4.1").data[0].embedding
    # temporary list to store (chunk, similarity) pairs
    similarities = []
    for chunk, embedding in VECTOR_DB:
        similarity = cosine_similarity(query_embedding, embedding)
        similarities.append((chunk, similarity))
    # sort by similarity in descending order, because higher similarity means more relevant chunks
    similarities.sort(key=lambda x: x[1], reverse=True)
    # finally, return the top N most relevant chunks
    return similarities[:top_n]
```

Put the relevant chunks into the instruction_prompt for the chatbot

```
input_query = 'Ask me a question: '
retrieved_knowledge = retrieve(input_query)

instruction_prompt = f'''You are a helpful chatbot.
Use only the following pieces of context to answer the question. Don't make up any new information:
{'\n'.join([f' - {chunk}' for chunk, similarity in retrieved_knowledge])}
'''
```

Finally, call the model:

```
input_messages = [  
    {'role': 'system', 'content': instruction_prompt},  
    {'role': 'user', 'content': input_query},  
]  
  
response = client.responses.create(  
    model="gpt-4.1",  
    input=input_messages  
)
```







Now all this can be iterated, various resources used, memory updated, etc.

There are various memory types, vector indexed, SQL databases, etc.

Chunking is an art in itself: Short or long chunks? Depends on the text, context, etc.

Example: Code Generation

Already many LLM based tools exist

Engine	Provider	Model	Features	
GitHub Copilot	GitHub + OpenAI	GPT-4 (via Azure)	IDE integration (VS Code, JetBrains), inline code suggestions, context-aware completion	
CodeWhisperer	Amazon	Proprietary	Integrated with AWS, supports Python/Java/JavaScript, security scanning	
ChatGPT Code Interpreter	OpenAI	GPT-4o / GPT-4-turbo	Advanced reasoning and coding, can run Python code, math and data analysis	
Gemini Code Assist	Google	Gemini 1.5 Pro	IDE plugins, strong integration with Google Cloud and Vertex AI	
Claude for Code	Anthropic	Claude 3 Opus	Large context window (200k+ tokens), excels at code review and understanding	
Cursor	Cursor.sh	GPT-4 / Claude / OSS	Code-focused IDE, agentic workflow features, Copilot alternative	

Cursor

- LLM based (primarily Claude but supports others)
- Autocompletion & chat based
- LLM integrated with the editor
- Features
 - Can understand your codebase
 - Runs terminal commands
 - Error detector
 - References the codebase (shows relevant places in your code that may solve the problem)
 - ...

See yourself <https://www.cursor.com/features>

Fun with web page code (Cursor)

My initial prompt: *I want you to create web portal, locally hosted, which has a standard layout of marketing web pages and contains a "HIT ME" button which makes a text "Have been hit" appear on the page in red color!*

The LLM generated HTML, CSS and JavaScript code underlying a server and the web-page (a slight overkill for a web page but fair enough)

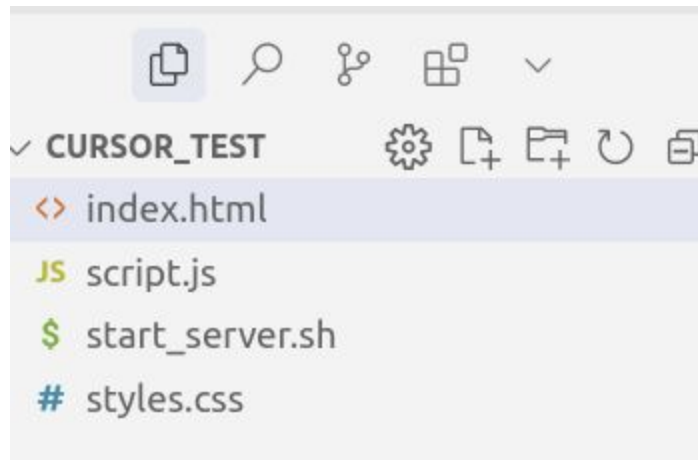
The code had two easily corrected typos in CSS

The LLM instructed me to open the html page

My next prompt: *give me a script that allows me to execute this on linux self hosted*

Created a sh script `start_server.sh`, applied `chmod +x start_server.sh` and told me to run the script

(that I managed, who knows what it would have done if I would not be able to execute the script)



```
JS script.js > ...
1 document.addEventListener('DOMContentLoaded', () => {
2     const hitMeButton = document.getElementById('hitMeButton');
3     const hitText = document.getElementById('hitText');
4
5     hitMeButton.addEventListener('click', () => {
6         hitText.textContent = 'Have been hit';
7         hitText.style.color = 'red';
8
9         // Add a small animation effect
10        hitText.style.transform = 'scale(1.1)';
11        setTimeout(() => {
12            hitText.style.transform = 'scale(1)';
13        }, 200);
14    });
15 };
```

```
< index.html x $ start_server.sh JS script.js # styles.css
< index.html > html > body > main
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Marketing Portal</title>
7     <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10 <header>
11     <nav>
12         <div class="logo">Marketing Portal</div>
13         <ul>
14             <li><a href="#home">Home</a></li>
15             <li><a href="#about">About</a></li>
16             <li><a href="#services">Services</a></li>
17             <li><a href="#contact">Contact</a></li>
18         </ul>
19     </nav>
20 </header>
21
22 <main>
23     <section class="hero">
24         <h1>Welcome to Our Marketing Portal</h1>
25         <p>Discover amazing opportunities and solutions for your business</p>
26         <button id="hitMeButton" class="hit-me-button">HIT ME</button>
27         <p id="hitText" class="hit-text"></p>
28     </section>
29
30     <section id="about" class="content-section">
31         <h2>About Us</h2>
32         <p>We are a leading marketing agency dedicated to helping businesses grow</p>
33     </section>
34
35     <section id="services" class="content-section">
36         <h2>Our Services</h2>
37         <div class="services-grid">
38             <div class="service-card">
39                 <h3>Digital Marketing</h3>
40                 <p>Comprehensive digital marketing solutions for your business</p>
41             </div>
42             <div class="service-card">
43                 <h3>Content Creation</h3>
44                 <p>Engaging content that resonates with your audience.</p>
45             </div>
46             <div class="service-card">
47                 <h3>SEO Optimization</h3>
48                 <p>Improve your online visibility and reach.</p>
49             </div>
50         </div>
51     </section>
52
53 </main>
54 </body>
55 </html>
```

```
# styles.css > nav ul li a > .contact-info a: hover
1 * {
2     margin: 0;
3     padding: 0;
4     box-sizing: border-box;
5 }
6
7 body {
8     font-family: 'Arial', sans-serif;
9     line-height: 1.6;
10    color: #333;
11 }
12
13 header {
14     background-color: #ffffff;
15     box-shadow: 0 2px 5px #000000;
16     position: fixed;
17     width: 100%;
18     top: 0;
19     z-index: 1000;
20 }
21
22 nav {
23     display: flex;
24     justify-content: space-between;
25     align-items: center;
26     padding: 1rem 5%;
27     max-width: 1200px;
28     margin: 0 auto;
29 }
30
31 .logo {
32     font-size: 1.5rem;
33     font-weight: bold;
34     color: #2c3e50;
35 }
36
37 nav ul {
38     display: flex;
39     list-style: none;
40 }
41
42 nav ul li {
43     margin-left: 2rem;
44 }
45
46 nav ul li a {
47     text-decoration: none;
48     color: #2c3e50;
49     font-weight: 500;
50     transition: color 0.3s ease;
51 }
```

Welcome to Our Marketing Portal

Discover amazing opportunities and solutions for your business

HIT ME

About Us

We are a leading marketing agency dedicated to helping businesses grow and succeed in the digital age.

Our Services

Digital Marketing

Comprehensive digital marketing solutions for your business.

Content Creation

Engaging content that resonates with your audience.

SEO Optimization

Improve your online visibility and reach.

Welcome to Our Marketing Portal

Discover amazing opportunities and solutions for your business

HIT ME

Have been hit

Fun with web page code

My next prompt:

*now add my email address
xbrazdil@gmail.com to
the contacts with my
name Tomas Brazdil*

About Us

We are a leading marketing agency dedicated to helping businesses grow and succeed in the digital age.

Our Services

Digital Marketing

Comprehensive digital marketing solutions for your business.

Content Creation

Engaging content that resonates with your audience.

SEO Optimization

Improve your online visibility and reach.

Contact Us

Get in touch with us to discuss your marketing needs.

Tomas Brazdil

xbrazdil@gmail.com

More complex situation

Programming OAuth library using Claude model, expert comments:

“Initially, I was fairly impressed by the code.” - all code in one file but well structure, not too many comments ... generated tests do not have sufficient coverage but at least some were generated

“A more serious bug is that the code that generates token IDs is not sound: it generates biased output. This is a classic bug when people naively try to generate random strings”

In other words, LLM may repeat “standard” errors made by humans

“The engineers clearly had a good idea of many aspects of the design, and the LLM was tightly controlled and produced decent code. (LLMs are absolutely good at coding in this manner). But it still tried to do some stupid stuff, some of which were caught by the engineers, some were not. I’m sure some are still in there. *Is this worse than if a human had done it? Probably not*”

AI Agents

Some History

- History of **agentic AI** reaches back to Turing's work on machine intelligence and Wiener's work on feedback systems.
- **Agents** equipped with
 - Perception
 - Planning
 - Reasoning
 - Learning
 - Acting

Have been present in AI development for decades

(See e.g. Artificial Intelligence: A Modern Approach by Russell and Norvig, first edition published in 1995)

- All reinforcement learning, game playing and autonomous systems literature has naturally been agentic

What is all the fuss about Agents now?

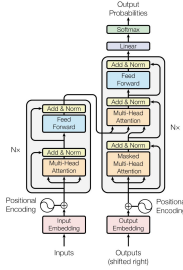
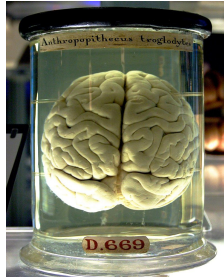
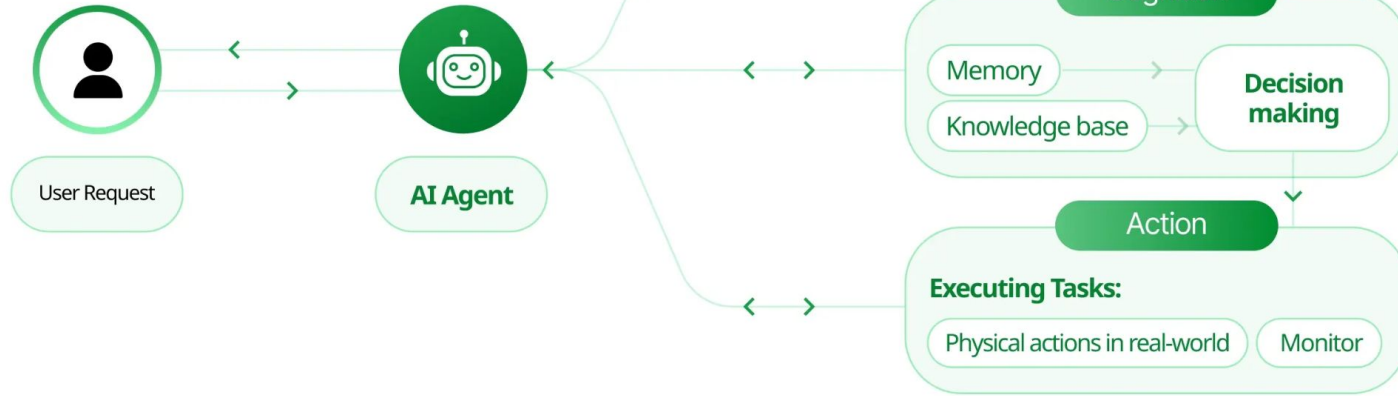


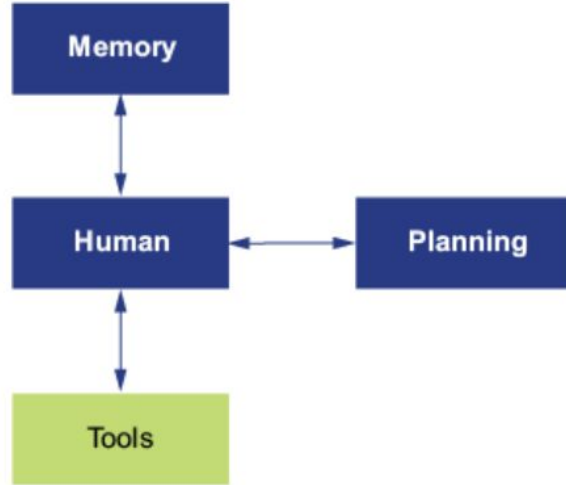
Figure 1: The Transformer - model architecture.



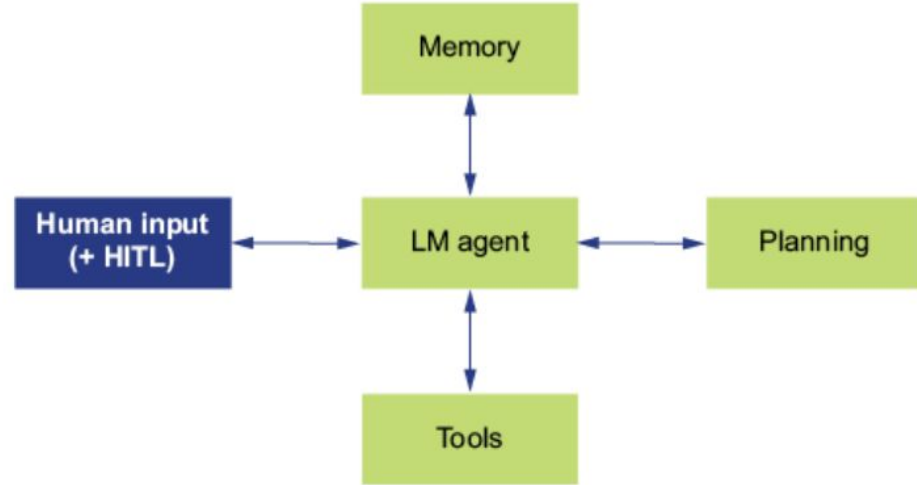
Power the agents with LLM brains.

Human vs LLM Agent Workflow

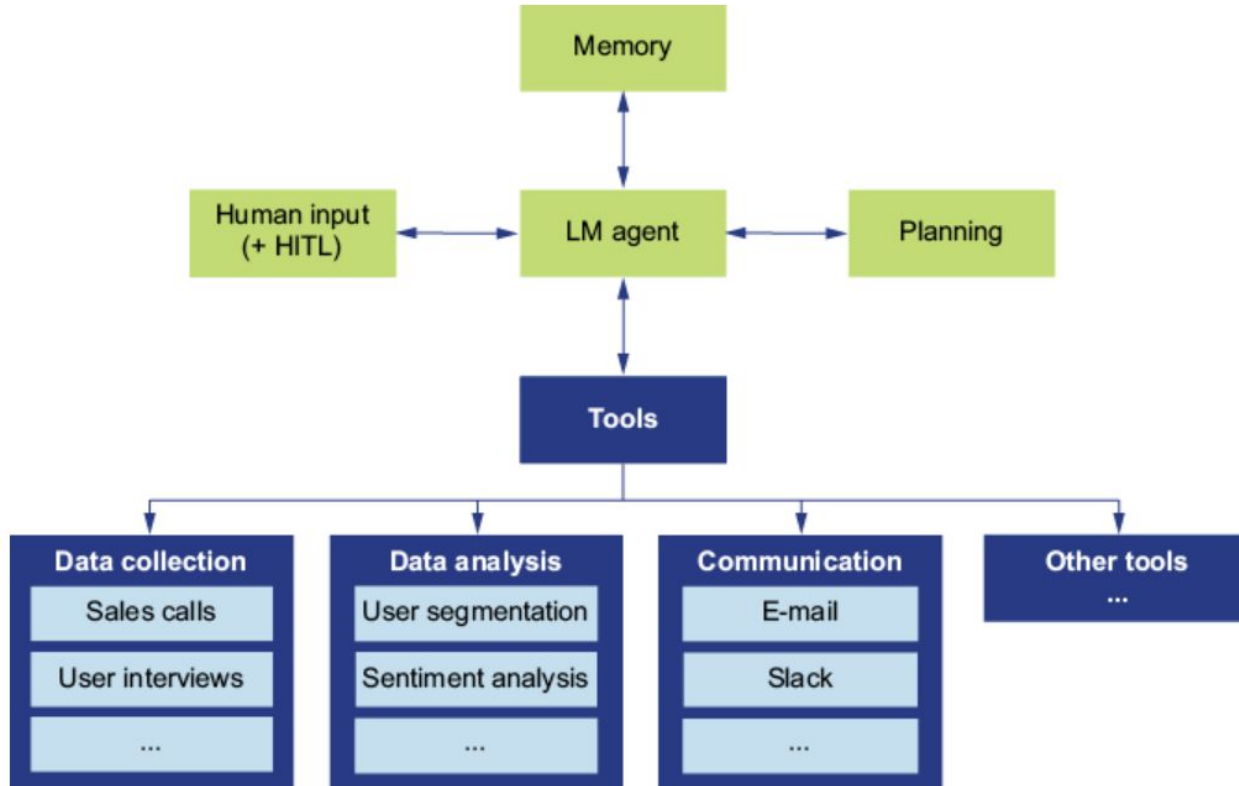
Manual workflow



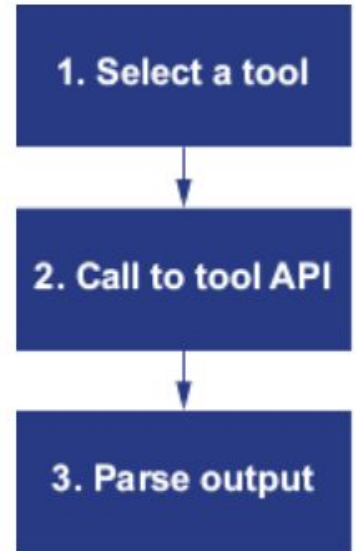
Agentic workflow



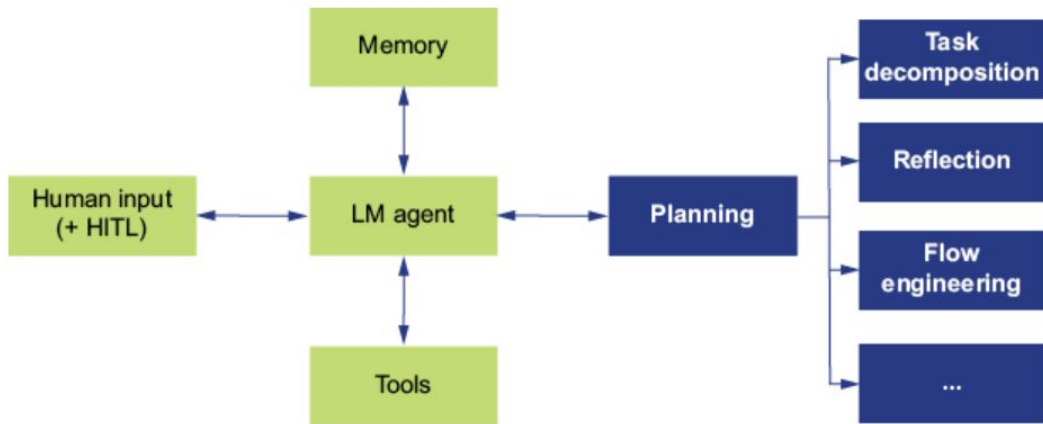
Tools



Using tools:

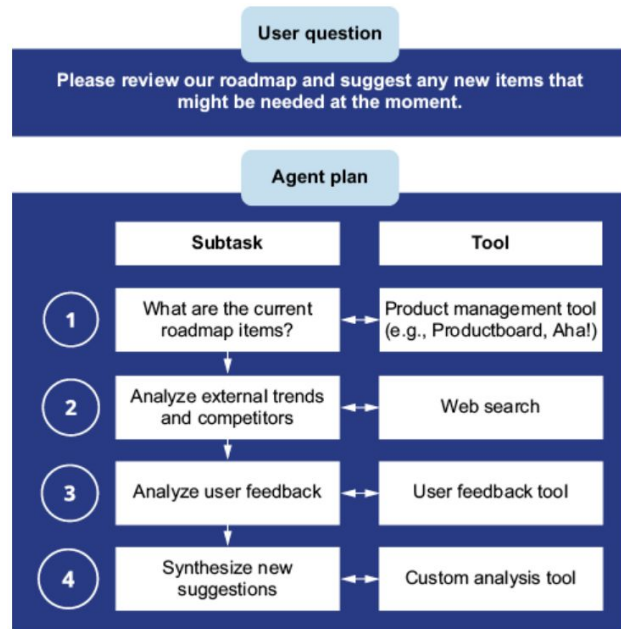


Planning



Agent prompted to decompose into steps:

- Prompt with “Let’s think step by step”
- Or better be more precise and ask for a plan for carrying out the plan



Planning

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?

A: *Let's think step by step.*

LLMs

(a)

First, 20% of the class enrolled in contemporary dance, so that means that 80% of the class is left.

Next, 25% of the remaining 80% enrolled in jazz dance, so that means that 25% of the original class enrolled in jazz dance. That leaves 55% of the original class.

Finally, the rest of the class (55%) enrolled in hip-hop dance. So, 55% of the entire class enrolled in hip-hop dance.



The agent can be instructed to use specific tools in steps, etc.

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?

A: *Let's first understand the problem and **devise a plan** to solve the problem. Then, let's **carry out the plan** and solve the problem step by step.*

LLMs

Plan:

Step 1: Calculate the total number of students who enrolled in contemporary and jazz dance.

Step 2: Calculate the total number of students who enrolled in hip-hop dance.

Step 3: Calculate the percentage of students who enrolled in hip-hop dance.

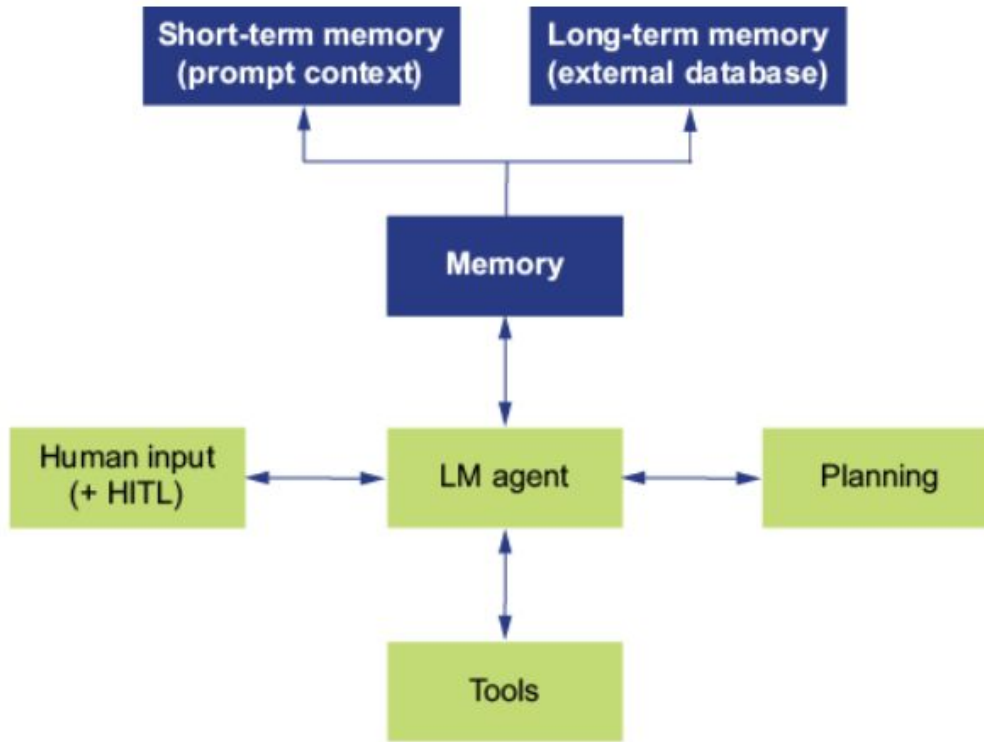
Solution:

Step 1: 20% of 20 students is 4 students. 25% of the remaining 16 students is 4 students. So, a total of 8 students enrolled in contemporary and jazz dance.

Step 2: The remaining 12 students enrolled in hip-hop dance.

Step 3: The percentage of students who enrolled in hip-hop dance is $12/20 = 60\%$.





Short-term : Prompt context, RAG

Long-term : RAG

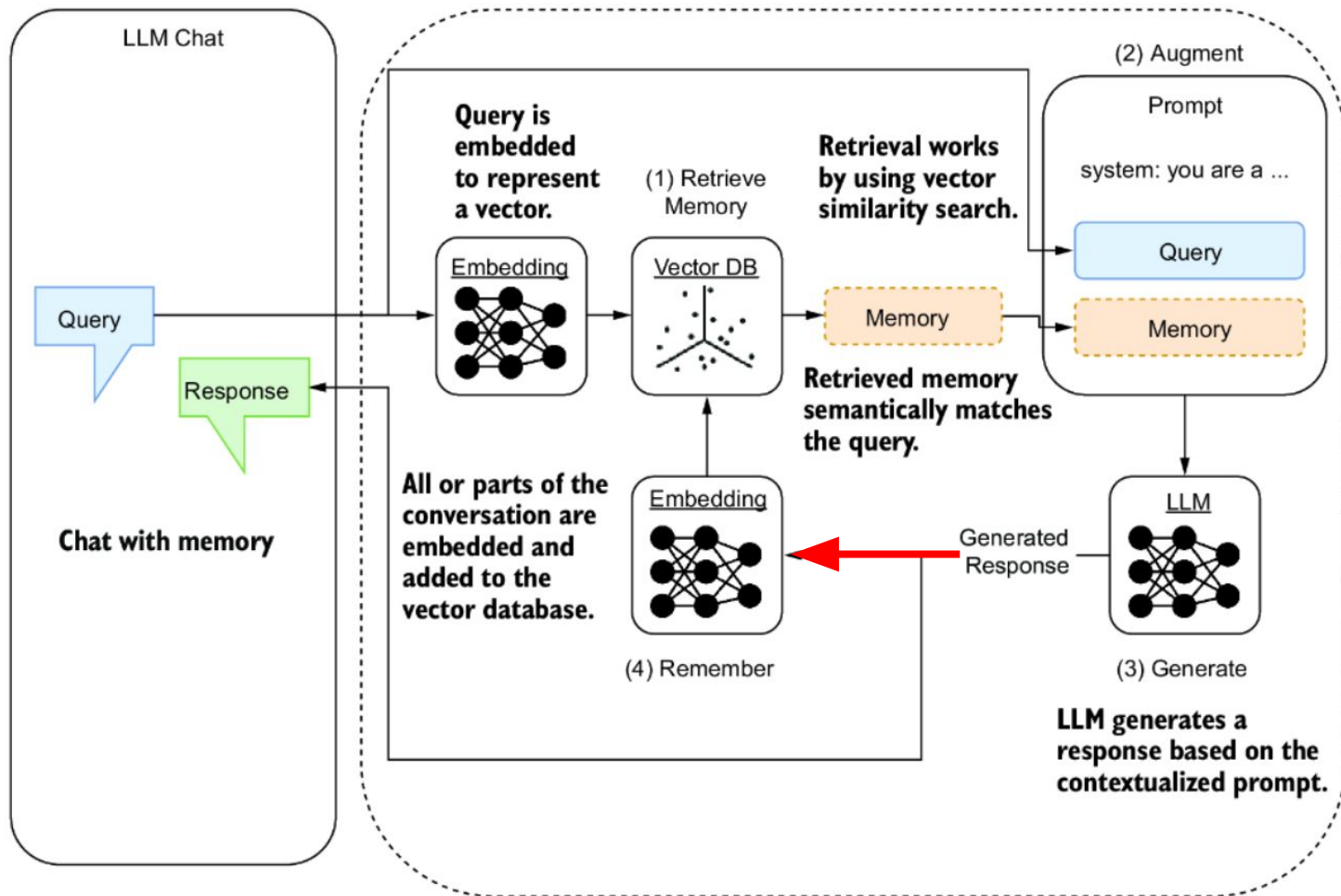
May store (embeddings of) interactions and search

May store generated summaries, where the agent decides, what to remember (combined with reflection)

Memory

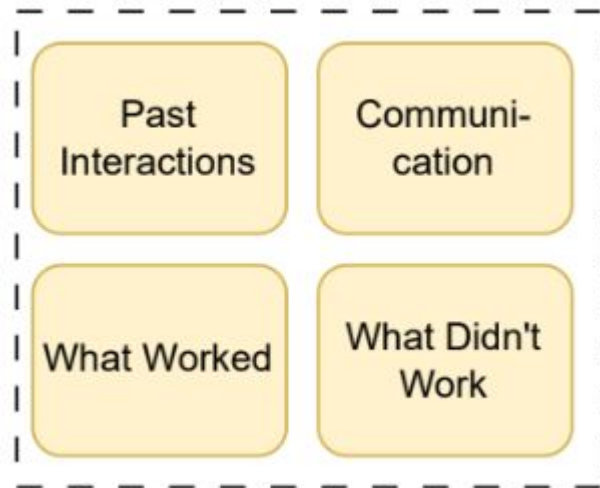
Note the **red** feedback arrow:

By modifying memory the agent implements long-term memory



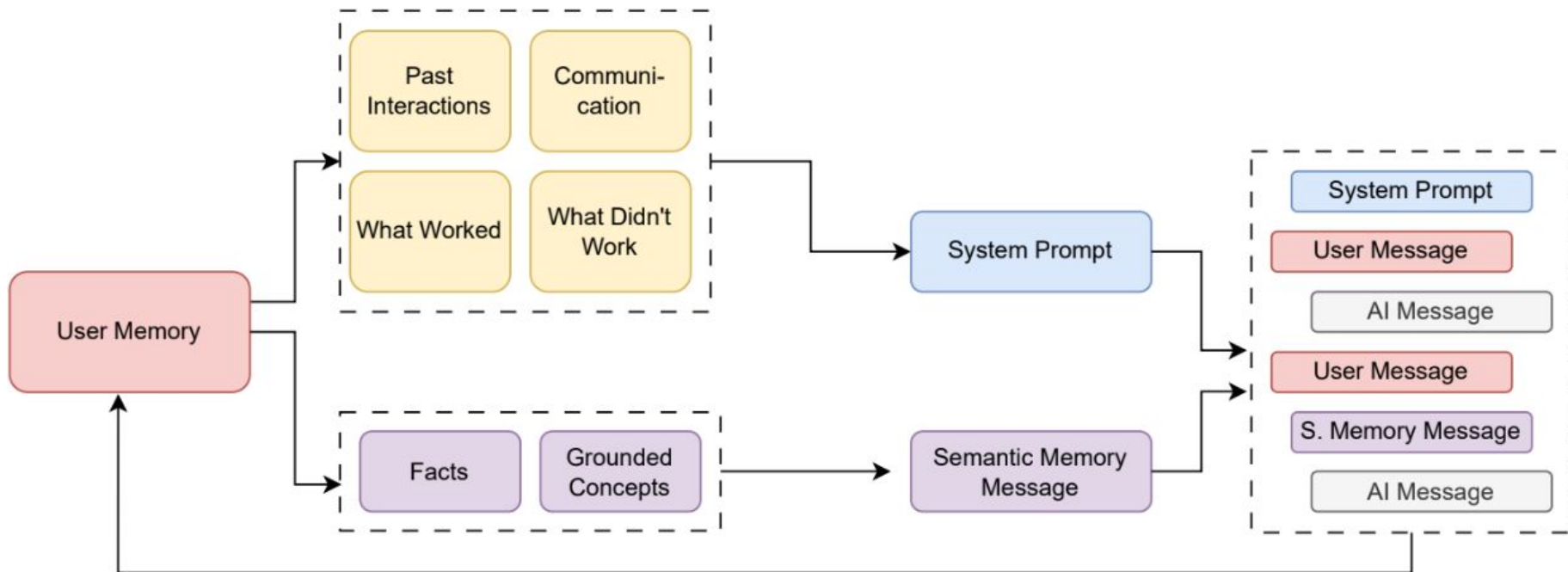
Memory types

- Working memory: The Immediate Context
Implementation: Store the past messages
- Episodic memory: Learning from the past
Implemented using RAG, two ways
 - Store past conversations and then search
 - Feedback-driven: Receive feedback on the performance (possibly by another agent)
-> update the memory



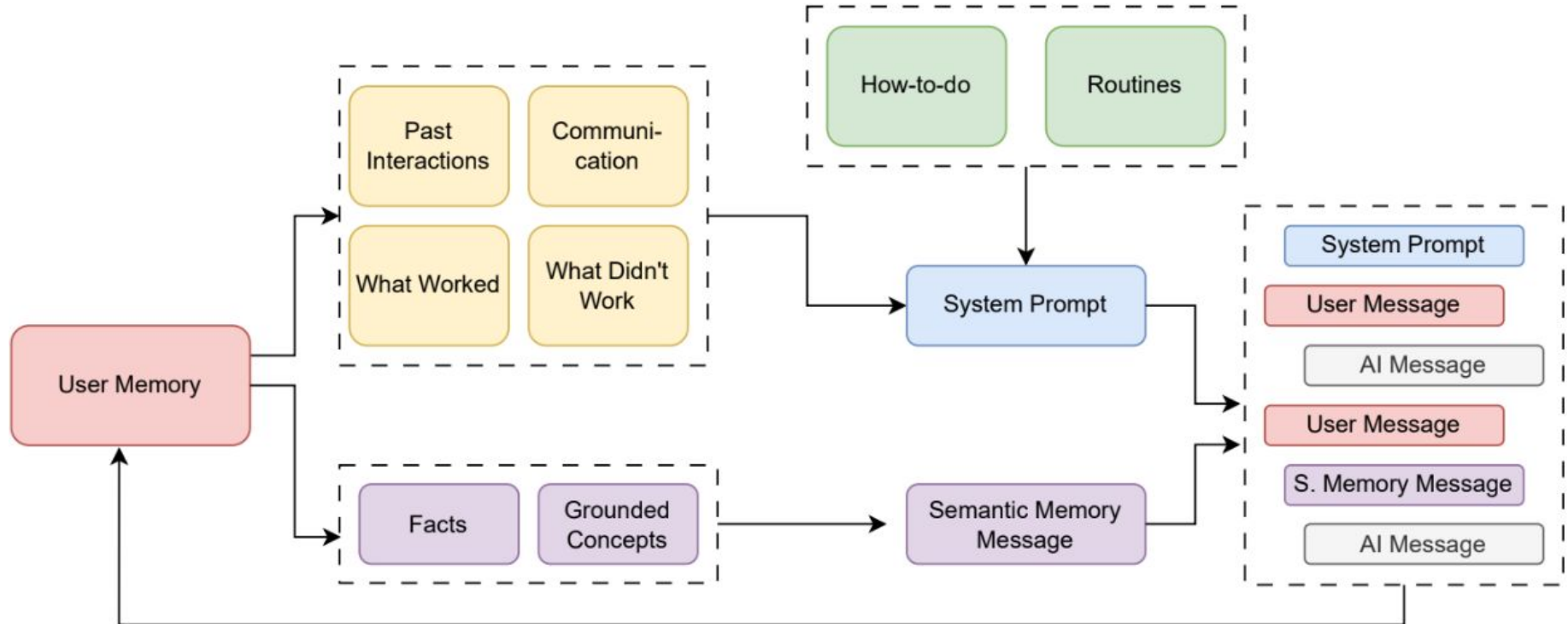
Memory types

- Semantic memory: Store knowledge
Key-value pairs, knowledge graphs



Memory types

- Procedural memory: How-to
Functions or agents represent the skills



Workflow

1. Take a user's message
2. Create a system prompt with relevant Episodic enrichment
3. Insert procedural memory into prompt
4. Create a Semantic memory message with context from the database
5. Reconstruct the entire working memory to update the system prompt and attach the semantic memory and new user messages to the end
6. Generate a response with the LLM

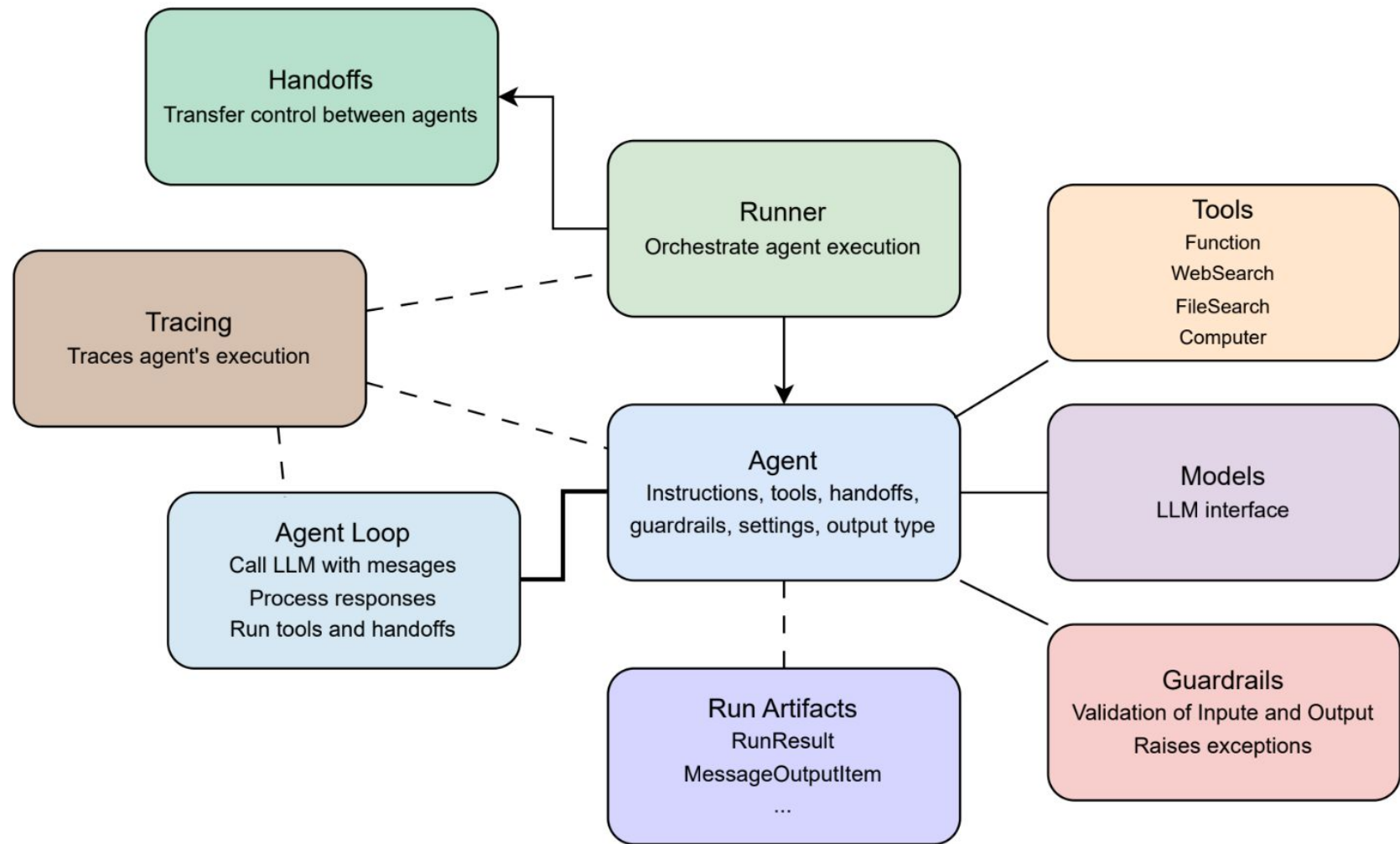
Example

Generated prompt: "Based on my previous experience assisting Dr. Smith with a colonoscopy AI analysis on June 3rd (episodic memory), and using the standardized diagnostic criteria for ulcerative colitis from the ECCO guidelines (semantic memory), I will now pre-process the new biopsy slides using the same segmentation workflow (procedural memory). Currently, I'm holding the patient metadata and the slide IDs in temporary storage (working memory), so I can match them with the existing histological models. Let's initiate inference and flag any slides with a Nancy Index score above 3."

- **Working Memory:** Holds current patient metadata and slide IDs temporarily for immediate use during task execution.
- **Episodic Memory:** Recalls a specific past experience, assisting Dr. Smith on June 3rd, to inform current judgment or preference.
- **Procedural Memory:** Applies a learned process (e.g., image pre-processing and segmentation workflow) to perform a task.
- **Semantic Memory:** Uses abstracted knowledge or general facts. Here, the ECCO guidelines and Nancy Index definition for ulcerative colitis severity.

OpenAI SDK

DOMAIN	DESCRIPTION	OPENAI PRIMITIVES
Models	Core intelligence capable of reasoning, making decisions, and processing different modalities.	o1, o3-mini, GPT-4.5, GPT-4o, GPT-4o-mini
Tools	Interface to the world, interact with environment, function calling, built-in tools, etc.	Function calling, Web search, File search, Computer use
Knowledge and memory	Augment agents with external and persistent knowledge.	Vector stores, File search, Embeddings
Audio and speech	Create agents that can understand audio and respond back in natural language.	Audio generation, realtime, Audio agents
Guardrails	Prevent irrelevant, harmful, or undesirable behavior.	Moderation, Instruction hierarchy (Python), Instruction hierarchy (TypeScript)
Orchestration	Develop, deploy, monitor, and improve agents.	Python Agents SDK, TypeScript Agents SDK, Tracing, Evaluations, Fine-tuning
Voice agents	Create agents that can understand audio and respond back in natural language.	Realtime API, Voice support in the Python Agents SDK, Voice support in the TypeScript Agents SDK



Some Definitions

- Handoffs: The new agent takes over the conversation, and gets to see the entire previous conversation history.
- Tool call: Agent calls a function.
- Guardrails:
 - Input guardrails:
 - Receive user input for the model
 - If the input satisfies tripwire conditions, raises exception
 - Output guardrails:
 - Receive output of the model
 - If the output satisfies tripwire conditions, raises exception

Illustration: Agent definition

```
INSTRUCTIONS = "You are a research assistant. Given a search term, you search the web for that term and produce a concise summary of the results. The summary must 2-3 paragraphs and less than 300 \ words. Capture the main points. Write succinctly, no need to have complete sentences or good \ grammar. This will be consumed by someone synthesizing a report, so it's vital you capture the \ essence and ignore any fluff. Do not include any additional commentary other than the summary itself."
```

```
search_agent = Agent(  
    name="Search agent",  
    instructions=INSTRUCTIONS,  
    tools=[WebSearchTool(search_context_size="low")],  
    model="gpt-4o-mini",  
    model_settings=ModelSettings(tool_choice="required"),  
)
```

Illustration: Agent definition

```
INSTRUCTIONS = f"You are a helpful research assistant. Given a query, come up with a set of web searches to perform to best answer the query. Output {HOW_MANY_SEARCHES} terms to query for."
```

```
class WebSearchItem(BaseModel):
    reason: str
    "Your reasoning for why this search is important to the query."

    query: str
    "The search term to use for the web search."

class WebSearchPlan(BaseModel):
    searches: list[WebSearchItem]
    """A list of web searches to perform to best answer the query."""
```

```
planner_agent = Agent(
    name="PlannerAgent",
    instructions=INSTRUCTIONS,
    model="gpt-4o-mini",
    output_type=WebSearchPlan,
)
```


Guardrails

```
@input_guardrail
async def math_guardrail(
    ctx: RunContextWrapper[None], agent: Agent, input: str | list[TResponseInput]
) -> GuardrailFunctionOutput:
    result = await Runner.run(guardrail_agent, input, context=ctx.context)

    return GuardrailFunctionOutput(
        output_info=result.final_output,
        tripwire_triggered=result.final_output.is_math_homework,
    )

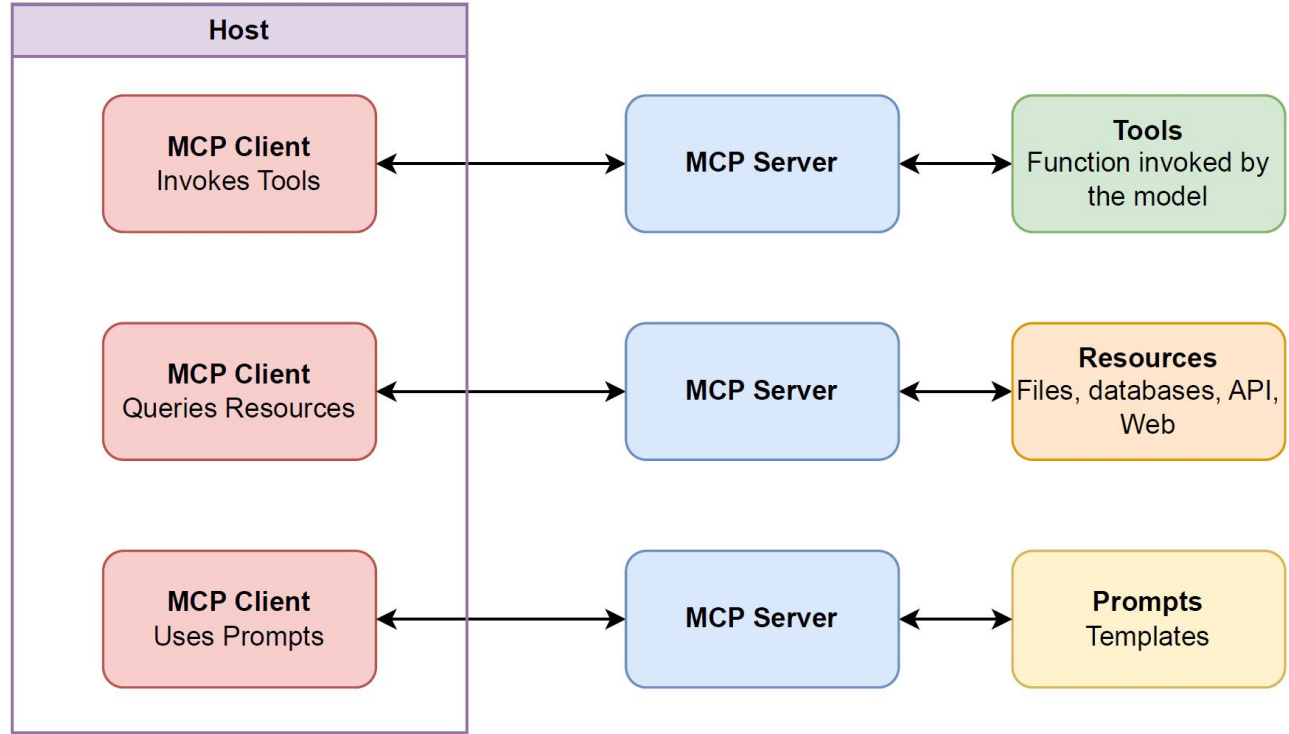
agent = Agent(
    name="Customer support agent",
    instructions="You are a customer support agent. You help customers with the",
    input_guardrails=[math_guardrail],
)
```

Model Context Protocol (USB-C of AI)

MCP = protocol for connecting (remote) tools with AI apps

Host (AI app) contains MCP client(s)

MCP clients connect with MCP servers that provide tools in a unified form



Allows seamless **INTEGRATION** of tools!

OpenAI SDK - MCP

```
instructions = """
You are a helpful assistant that can help with tasks concerning the Czech Republic.
You can use the files_tools to get the files and the browser_tools to browse the internet.
"""

prompt = """
Find all restaurants in Prague that serve roast pork with sauerkraut and dumplings, then summarize it in markdown to ZPK.md
"""

files_params = {"command": "npx", "args": ["-y", "@modelcontextprotocol/server-filesystem", sandbox_path]}
puppeteer_params = {"command": "npx", "args": ["-y", "@modelcontextprotocol/server-puppeteer"]}

async with MCPServerStdio(params=files_params, cache_tools_list=True) as mcp_server_files:
    async with MCPServerStdio(params=playwright_params, cache_tools_list=True) as mcp_server_browser:
        agent = Agent(
            name="cuisine-assistant",
            instructions=instructions,
            model="gpt-4o-mini",
            mcp_servers=[mcp_server_files, mcp_server_browser]
        )
        with trace("investigate"):
            result = await Runner.run(agent, prompt)
            print(result.final_output)
```

The two servers expose file system and web browsing tools to the LLM

Manufacturing Maintenance

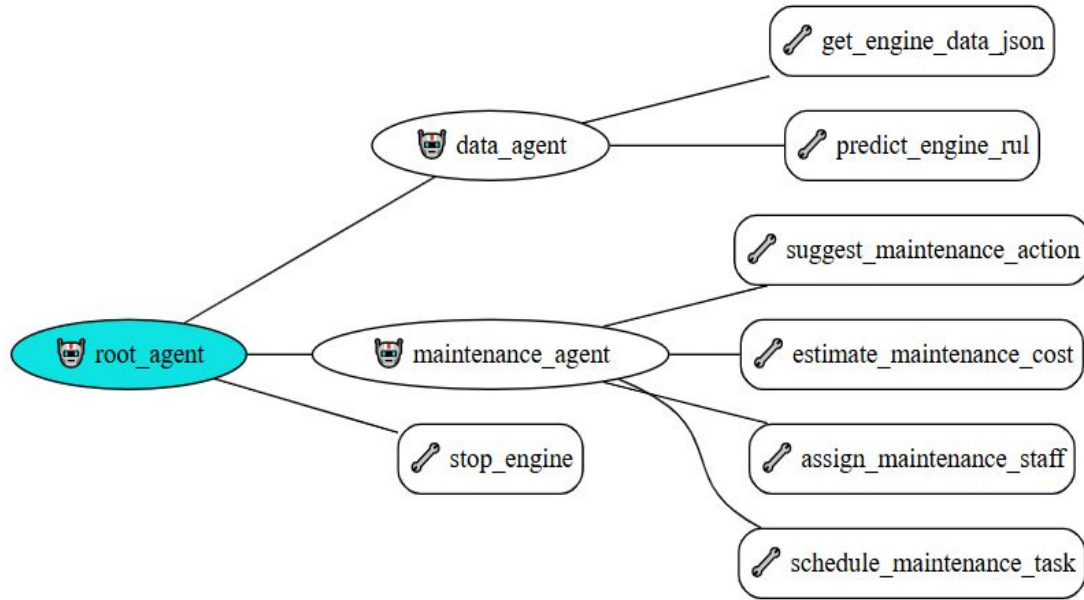
CMAPSS dataset - Aero-Propulsion System Simulation

Prognostics and Health Management (PHM)
models for predictive maintenance

source of industrial data to demonstrate
the system’s ability to interact with a dynamic environment

Attribute	Unit	Type
Engine ID	-	Index
Cycle	-	
Speed	Ma	Operational Setting
Altitude	feet	
Sea level temperature	°F	
Fan inlet temperature	°R	Sensor
LPC outlet temperature	°R	
HPC outlet temperature	°R	
LPT outlet temperature	°R	
Fan inlet pressure	psia	
Bypass-duct pressure	psia	
HPC outlet pressure	psia	
Physical fan speed	rpm	
Physical core speed	rpm	
Engine pressure ratio	-	
HPC outlet static pressure	psia	
Ratio of fuel flow	pps/psia	
Corrected fan speed	rpm	
Corrected core speed	rpm	
Bypass ratio	-	
Burner fuel-air ratio	-	
Bleed enthalpy	-	
Required fan speed	rpm	

Manufacturing Maintenance



root_agent : decomposes user intent into *Expectations, Conditions, Targets, Context* and *Information*

Plans and delegates to other agents

data_agent : retrieves engine telemetry, predicts RUL (in this simulation they predicted from dataset)

maintenance_agent : plans maintenance using tools:

- suggest_maintenance_action
- estimate_maintenance_cost
- assign_maintenance_staff
- schedule_maintenance_task

root_agent receives outputs of the **data_agent** and **autonomously** decides whether to invoke **maintenance_agent** for preventive action or to shut down the engine

"I need to maintain all engines working well according to their predicted RUL, avoiding unexpected stops, please make a consolidated predictive maintenance plan in a table format."

- **Expectation:** Maintain all engines in optimal operating condition to prevent unexpected failures. This was inferred from the phrase "maintain all the engines working well according to their predicted RUL, avoiding unexpected stops," implying continuous, reliable engine performance.
- **Targets:** All 20 engines in the fleet. The request for a consolidated plan covering "all the engines" indicates that the intent applies to the entire fleet under management.
- **Conditions:** RUL must be above a critical threshold, as defined by the `maintenance_agent`. This stems from the clause "according to their predicted RUL," which implies that engines with RULs below the threshold should be prioritized for maintenance.
- **Context:** High-priority, proactive maintenance strategy aimed at minimizing unplanned downtime and maximizing engine lifespan. The phrasing of the request reflects a strategic and preventative approach.
- **Information:** Predicted RUL values and engine sensor data, obtained from the `data_agent`, are required to evaluate each engine's condition and inform decision-making.

Manufacturing Maintenance - Final Plan

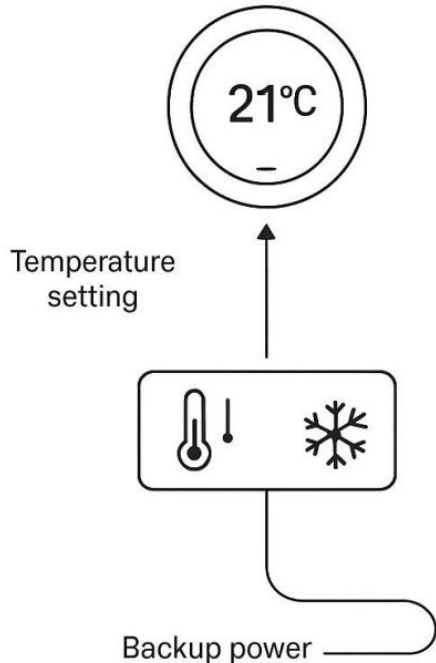
# Engines	RUL Range	Recommended Action	Priority	Cost (USD)	Labor Hours	Assigned Staff	Scheduled Time
15	82-124	MONITOR	low	0	0	[jr_mechanic]	Within 7 days
1	69	MONITOR	low	0	0	[jr_mechanic]	Within 3 days
2	28, 50	REPAIR	high	6000	4	[mechanic, jr_mechanic]	Within 3 days
1	16	STOP	critical	15000	8	[tech_lead, sr_mechanic]	IMMEDIATE

According to the human oversight, the actions are correct

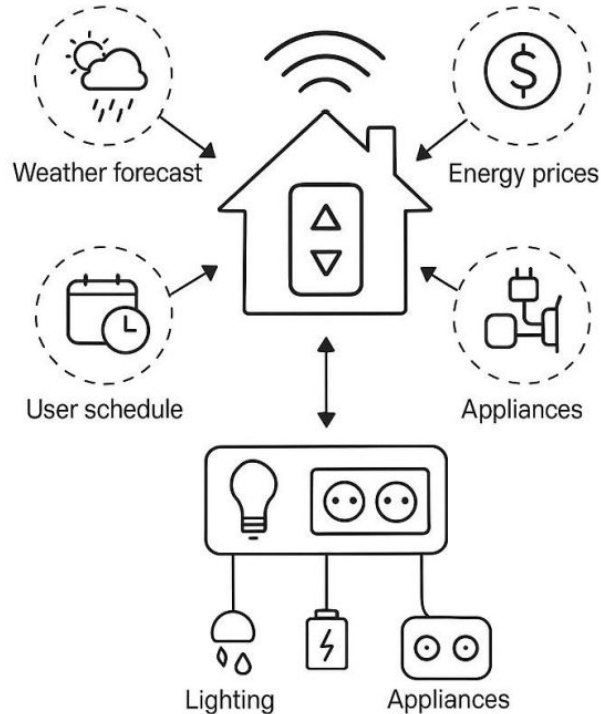
Agentic AI

AI Agents vs Agentic AI

AI Agent



Agentic AI



A single-task AI Agent.

vs

A multi-agent,
collaborative
Agentic AI system

Level	Healthcare Diagnostics Analogy	Agentic AI Analogy	Technology
Level 0 - Manual Operations (Human-Only)	Doctor performs all diagnostics manually using basic tools (e.g., stethoscope, clipboard).	Humans perform all tasks without automation.	Basic digital tools (e.g., Excel, email), manual processing.
Level 1 - Rule-Based Automation	System flags abnormal lab values based on preset rules; doctor still makes decisions.	Simple automation follows fixed rules (e.g., data entry, RPA systems).	Basic automation tools (RPA, simple scripts, rule engines).
Level 2 - Intelligent Process Automation	System highlights potential diagnoses based on lab and imaging patterns but still requires doctor confirmation.	AI combines automation with cognitive abilities like NLP and machine learning.	AI tools (ML, NLP, CV, RPA, process orchestration).
Level 3 - Agentic Workflows	System generates draft diagnostic reports from multimodal data and recommends tests or referrals; doctor supervises.	Agents plan, reason, and adapt in defined domains.	Foundation models, memory systems, workflow orchestration, reinforcement learning.
Level 4 - Semi-Autonomous Agents	AI conducts full diagnostic workup in narrow specialties, including decision and follow-up plan.	Agents work autonomously within defined expertise.	Advanced reasoning, real-time planning, causal inference.
Level 5 - Fully Autonomous Agents	AI acts as primary diagnostician across all specialties with no human input.	AI systems handle any task, adapt and reason independently.	Advanced learning mechanisms, full autonomy infrastructure.

Feature	AI Agents	Agentic AI
<i>Definition</i>	Autonomous software programs that perform specific tasks.	Systems of multiple AI agents collaborating to achieve complex goals.
<i>Autonomy Level</i>	High autonomy within specific tasks.	Broad level of autonomy with the ability to manage multi-step, complex tasks and systems.
<i>Task Complexity</i>	Typically handle single, specific tasks.	Handle complex, multi-step tasks requiring coordination.
<i>Collaboration</i>	Operate independently.	Involve multi-agent information sharing, collaboration and cooperation.
<i>Learning and Adaptation</i>	Learn and adapt within their specific domain.	Learn and adapt across a wider range of tasks and environments.
<i>Applications</i>	Customer service chatbots, virtual assistants, automated workflows.	Supply chain management, business process optimization, virtual project managers.

Agentic AI is a new area where people (sometimes) marvel about agent's capabilities

Agentic manufacturing optimization

Quoting from Bornet et al. Agentic Artificial Intelligence: Harnessing AI Agents to Reinvent Business, Work and Life. Irreplaceable Publishing, 2025

1. Initial Task: The AI agent monitors factory machinery to predict potential breakdowns and minimize downtime. For example, it analyzes data from sensors tracking vibration, temperature, and wear.

2. Action Generation: Based on the retrieved insights, the AI generates actionable recommendations:

“The vibration pattern suggests bearing wear in Machine X. Schedule a bearing replacement within the next 72 hours to prevent failure.”

3. Automated Feedback Through Revenue Metrics:

- The system tracks the financial outcomes of its actions using predefined indicators, such as reduced downtime, lower repair costs, or increased output.
- If the maintenance intervention prevents a breakdown, it records this as a positive outcome and links it to the specific recommendation and retrieved data.

Agentic manufacturing optimization

Still quoting from Bornet et al. Agentic Artificial Intelligence: Harnessing AI Agents to Reinvent Business, Work and Life. Irreplaceable Publishing, 2025

4. Positive Reinforcement Learning:

- The AI reinforces the association between vibration patterns and bearing wear in its predictive model.
- It flags the retrieved data as highly relevant for similar issues, improving its retrieval accuracy for future anomalies.

5. Updating the memory:

- Maintenance logs and outcomes from this event are added to the database, creating new knowledge the system can draw from in the future.
- The system also incorporates cost-benefit analysis, associating specific actions with the revenue saved or generated.

6. Adaptive Behavior: Over time, the AI becomes better at identifying subtle signs of failure earlier, optimizing its recommendations to reduce costly downtime. It may also learn to prioritize actions based on financial impact, ensuring the most critical interventions are addressed first.

Digital labor is
REVOLUTIONIZING
how businesses operate

salesforce



Conclusions

We went rapidly through the whole evolution from predictive to agentic AI

- Predictive models trained in supervised manner
- Agents trained using reinforcement learning
- LLM trained using self-supervised, supervised, and reinforcement learning
- LLM Agents training on themselves using prompting and RAG
- Agentic AI training emergent behaviors from collaboration of agents

Keep in mind that the current results are mostly experimental

The main problems: Deployment and monetization!