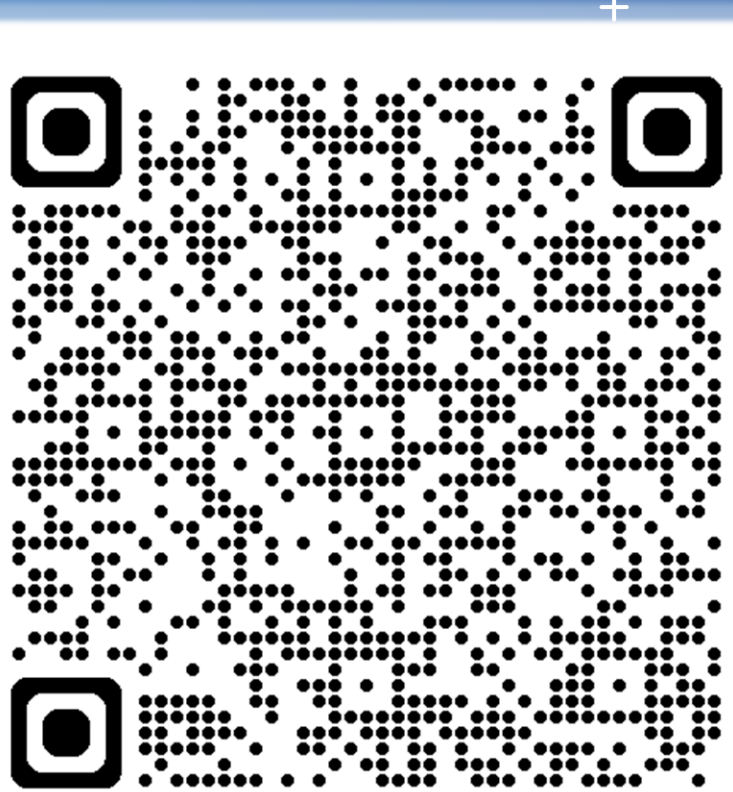
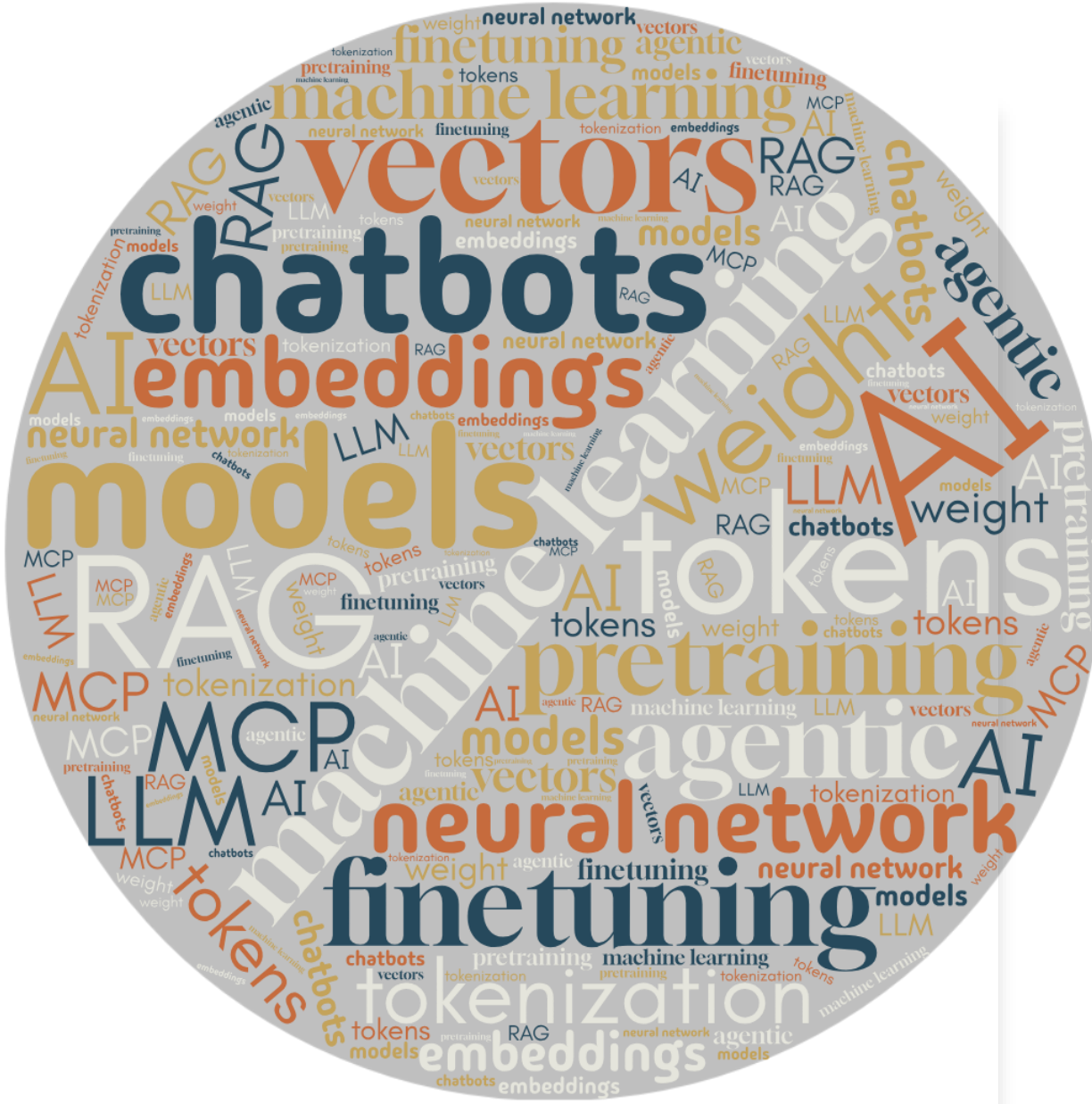


Getting Started with AI for .NET Devs with Semantic Kernel

Daniel Ward

Goals of this talk

- I made this talk for me
- Explaining terms/jargon
- Semantic Kernel
- 50% theory, 50% practice
- Be able to speak about AI



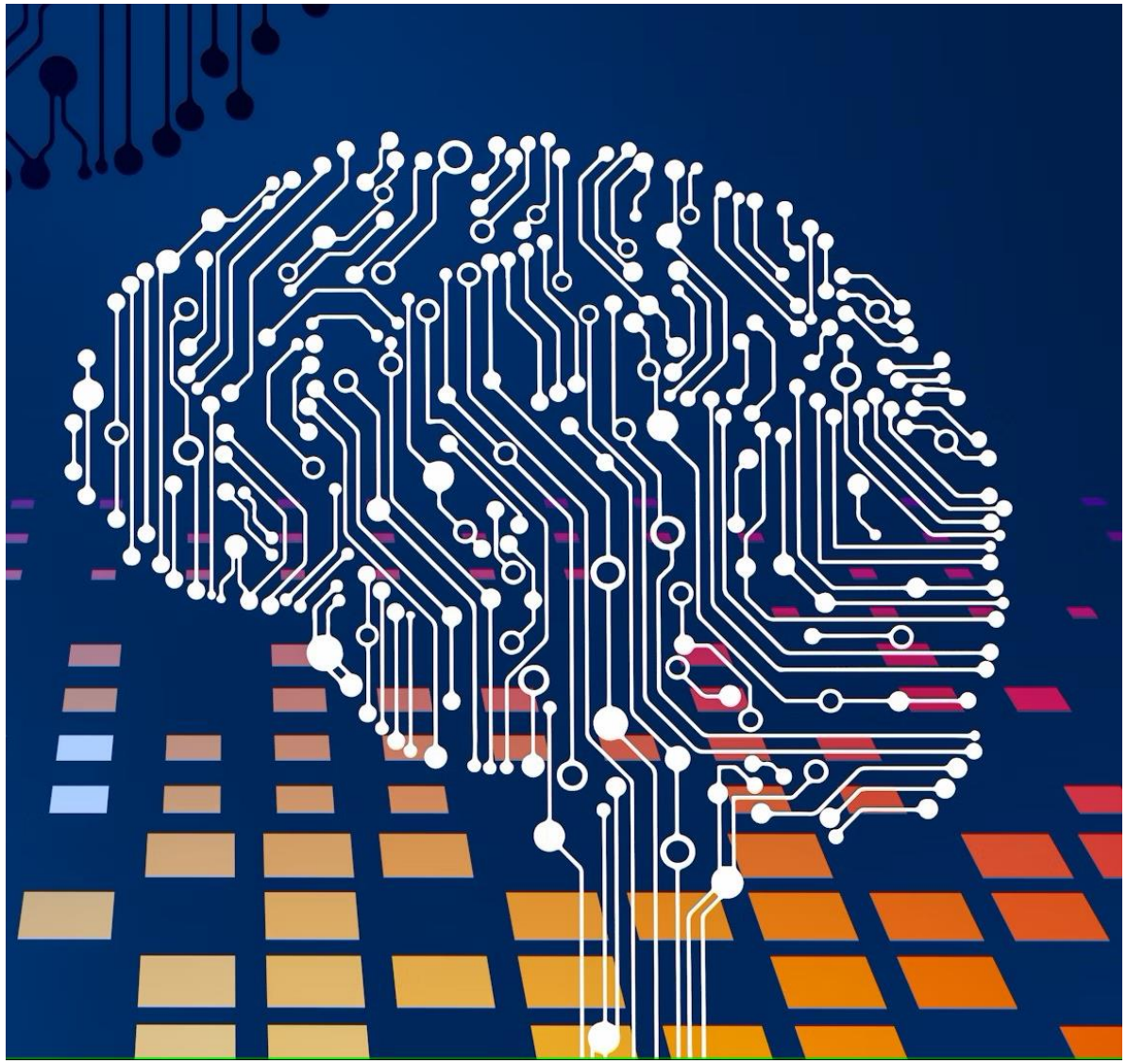
Who am I?

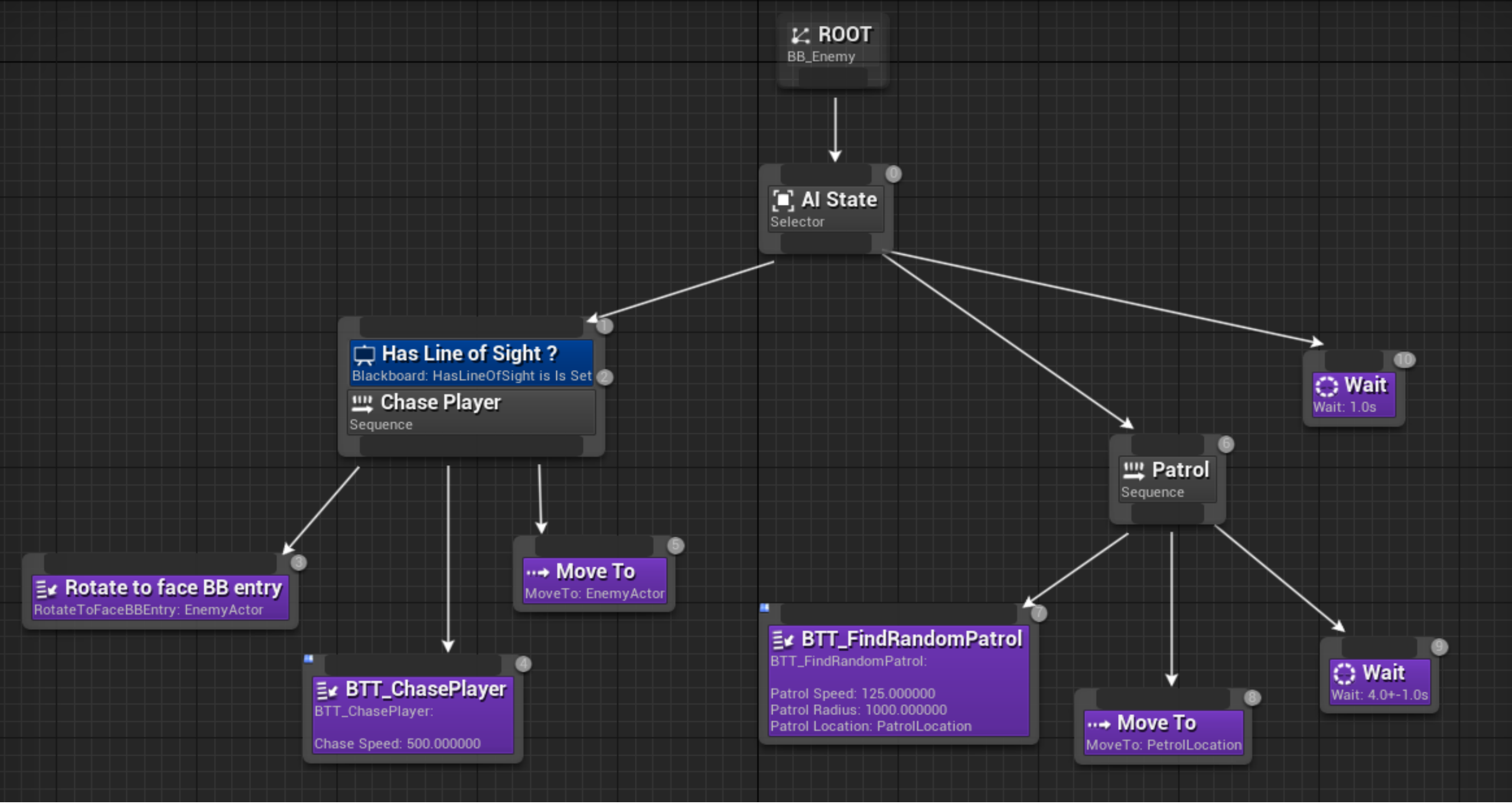
https://leantechniques.com

- Software developer, consultant
- Microsoft .NET MVP
- Co-organizer of the San Antonio/Austin .NET User Group
- daninacan.com
- @danielwarddev
- daniel-ward-dev
- danielwarddev.bsky.social

What is AI, anyway?

- Simulating human intelligence
- Not new

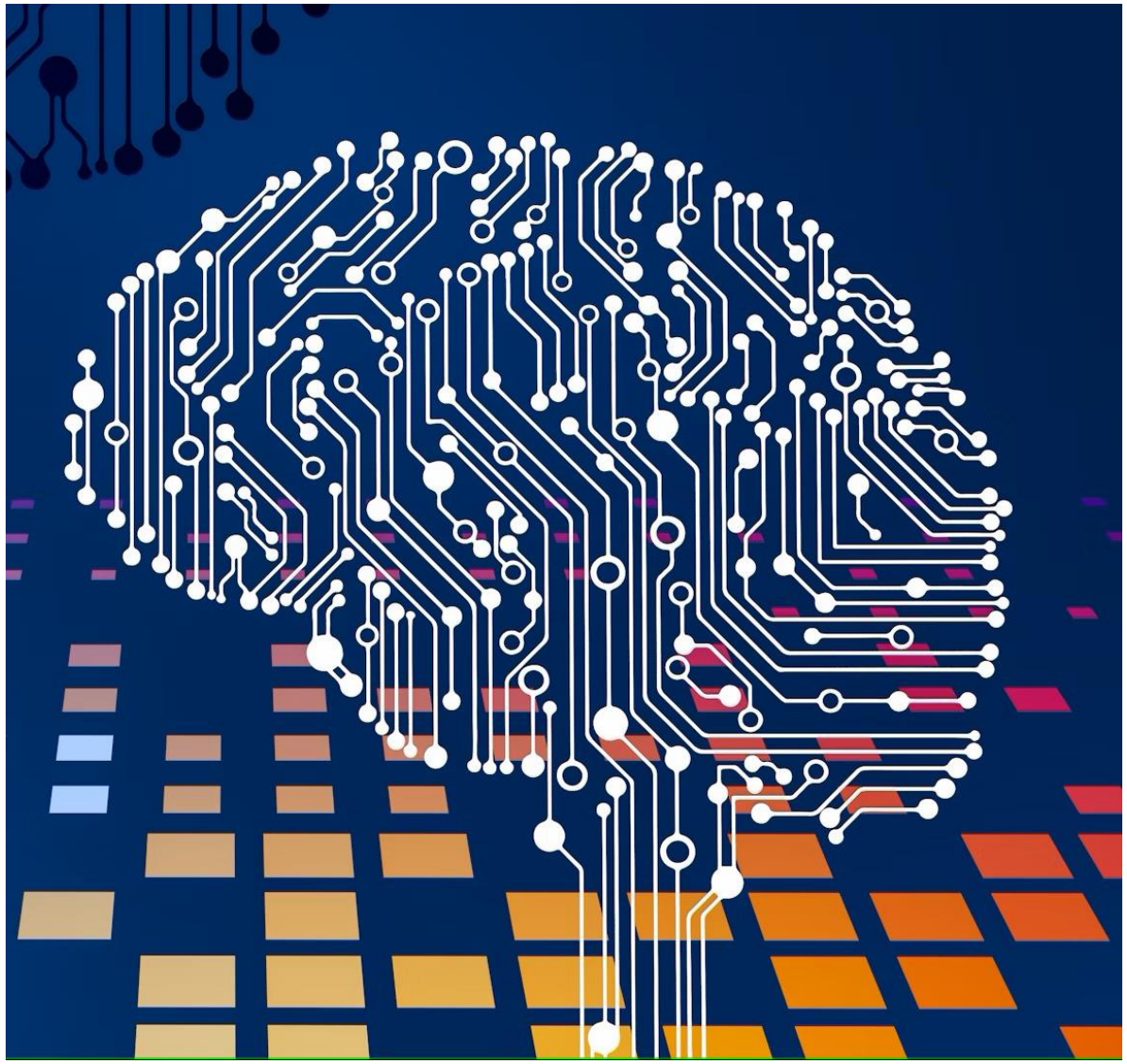




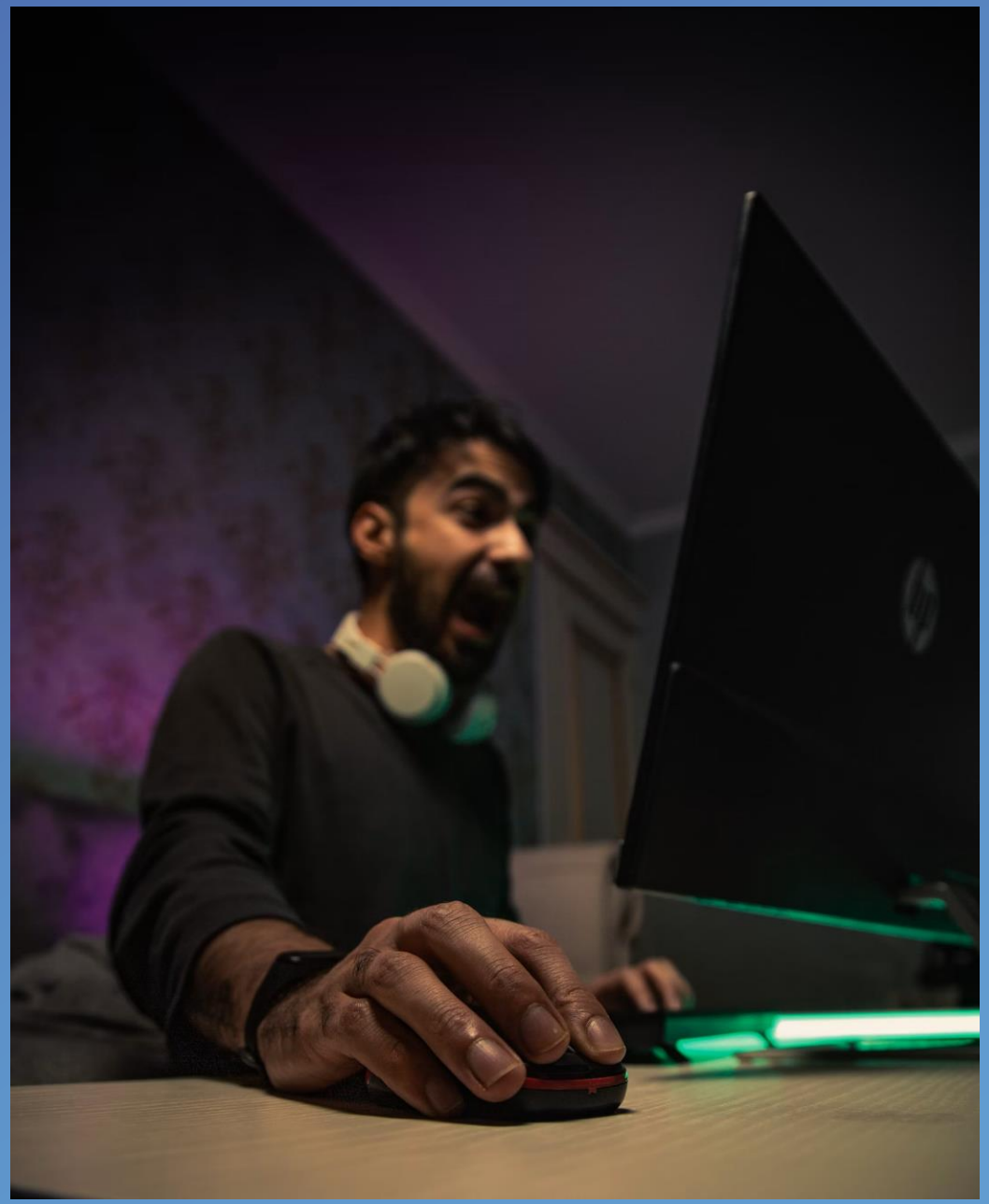
Behavior tree in Unreal Engine. https://dev.epicgames.com/documentation/en-us/unreal-engine/behavior-tree-in-unreal-engine---overview

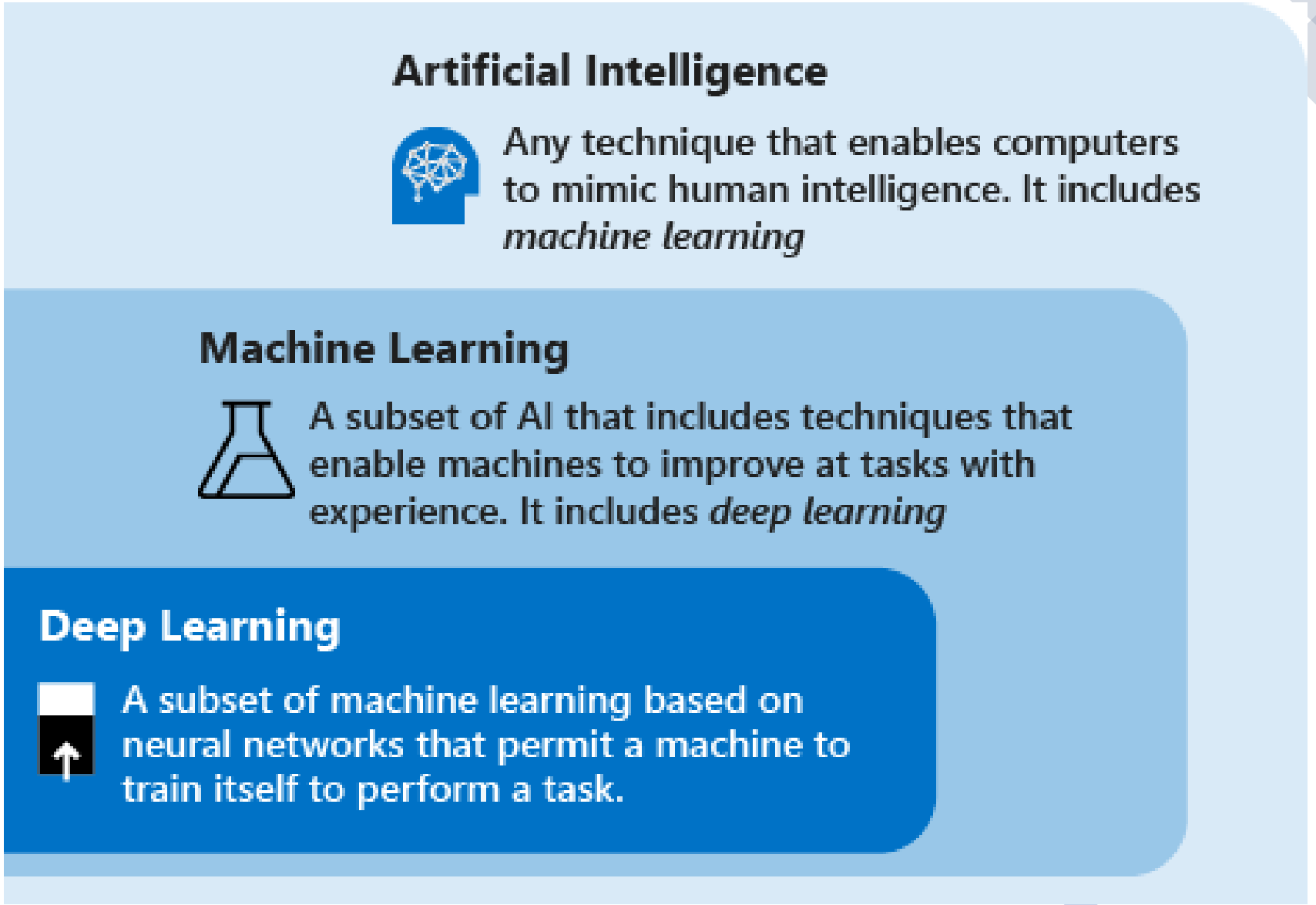
What is AI, anyway?

- Simulating human intelligence
- Not new
- ⭐ 1950: Turing Test
- 1960s: learning checkers; algebra word problems; English; neural networks
- 2010s – machine learning (ML), deep learning
- 2010s, 20s – transformers, large language models (LLMs), generative AI
  - Copilot, ChatGPT, etc.



AI vs ML vs deep learning

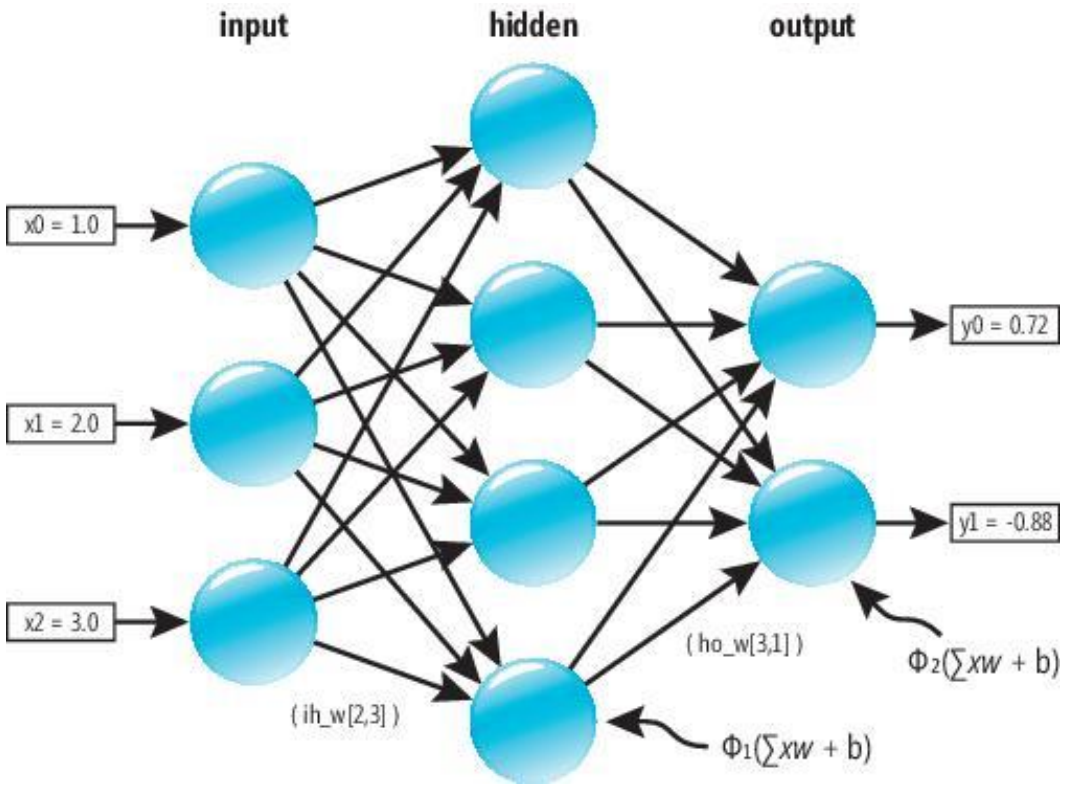




https://learn.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning?view=azureml-api-2

Neural network

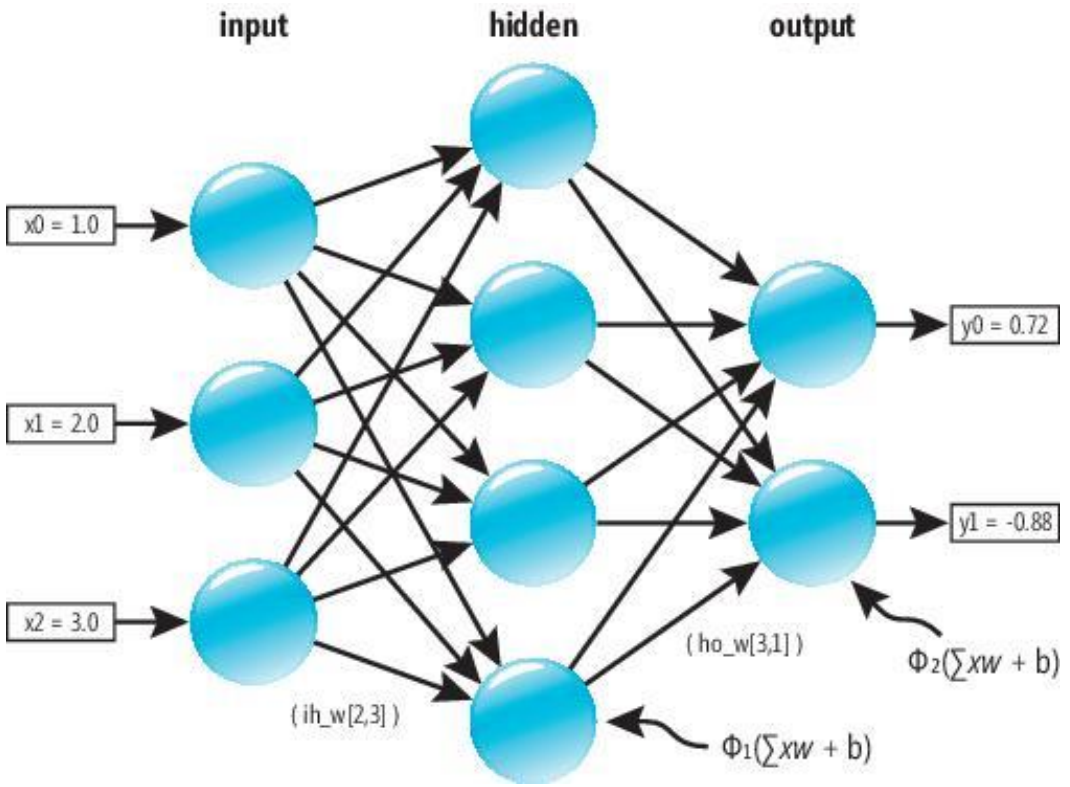
- Loosely based on biological neurons/synapses
- Central part of modern ML
- 3 layers: input, hidden, output
- Each node is a math function
- The more layers, the deeper it is



https://learn.microsoft.com/en-us/archive/msdn-magazine/2012/may/test-run-dive-into-neural-networks

Weights and biases

- output = inputs * weights + bias
- Both weights and biases are changed during training
- **Weight** - strength of a connection between nodes; scales input
- **Bias** - how “sensitive” a node is - helps it activate even with low inputs; flat value
- **Parameters** are the weights + biases



https://learn.microsoft.com/en-us/archive/msdn-magazine/2012/may/test-run-dive-into-neural-networks

How is it possible AI creates new stuff?

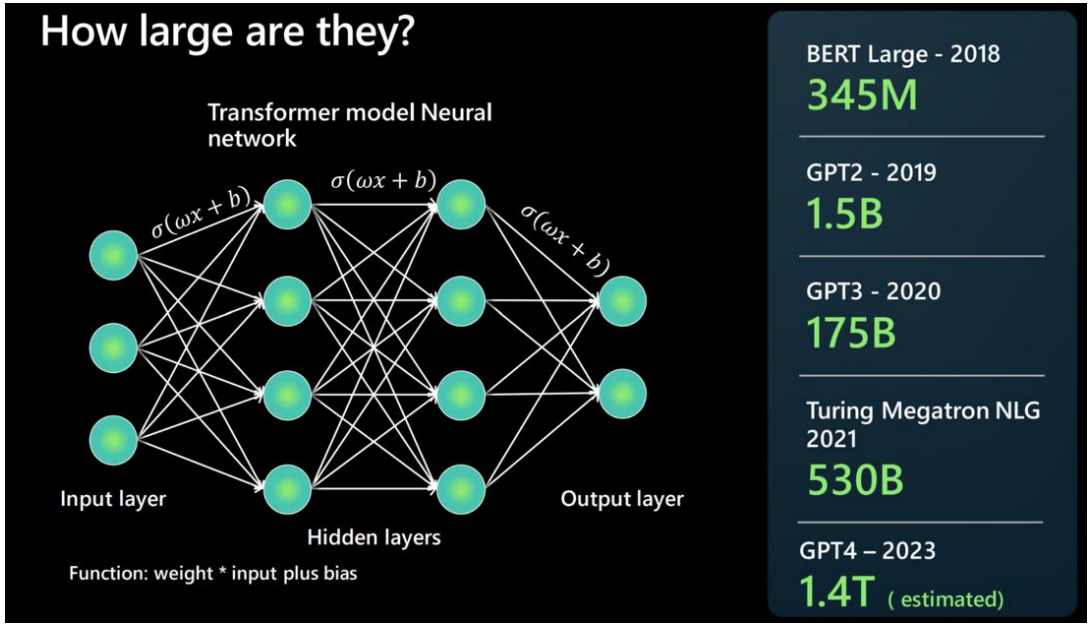
- MIT Technology Review (2023, December 14). *Google DeepMind used a large language model to solve an unsolved math problem*. https://www.technologyreview.com/2023/12/14/1085318/google-deepmind-large-language-model-solve-unsolvable-math-problem-cap-set/
- BBC. (2024, March 20). *NHS AI test spots tiny cancers missed by doctors*. https://www.bbc.com/news/technology-68607059

Generative AI

- Subset of AI
- Uses techniques like deep learning to generate new content
- Depends on a HUGE amount of pre-trained knowledge
- ChatGPT, GitHub Copilot, etc.

Large Language Models (LLMs)

- Type of ML model designed for natural language processing tasks



https://microsoft.github.io/Workshop-Interact-with-OpenAI-models/llms/

Generative Pre-trained Transformer (GPT)

- **Generative** - generates new content
- **Pre-trained** - goes through pretraining
- **Transformer** - specific type of neural network
- A kind of LLM
- Takes n tokens as input and produces one as output

1. How does it understand what we ask it and respond?
2. How does it refer back to its own knowledge?

Tokens

- “Building blocks” of input
- Middle ground between characters and words
- Model keeps a complete list of all possible tokens

Tokens	Characters
9	51

Vectors

- Tokens are embedded into vectors
  - Mathematical representation of data
  - “king” -> [0.12, -0.98, 0.55, …, 1.06]
- Semantic similarity
  - “apple” and “orange” are closer than “apple” and “car”
  - Each dimension is a “feature”
- GPT-3 had 12288 dimensions

Example: ChatGPT

Write me a haiku about C#

```
Write me a haiku about C#
```

Brackets hold the flow,
LINQ dances with sharp logic—
Types bloom, strong and bold.

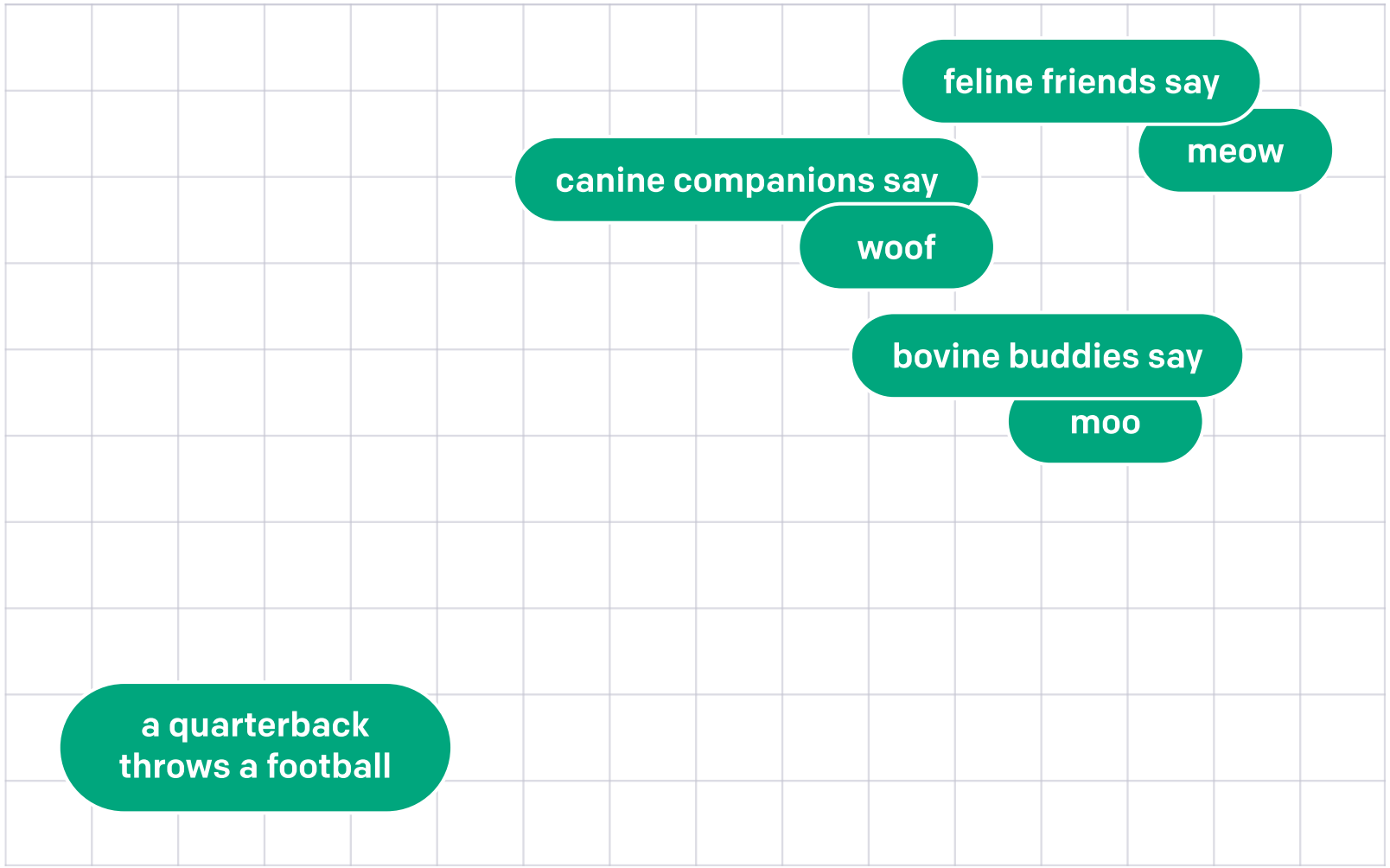
```
Brackets hold the flow,
LINQ dances with sharp logic—
Types bloom, strong and bold.
```

Example: ChatGPT

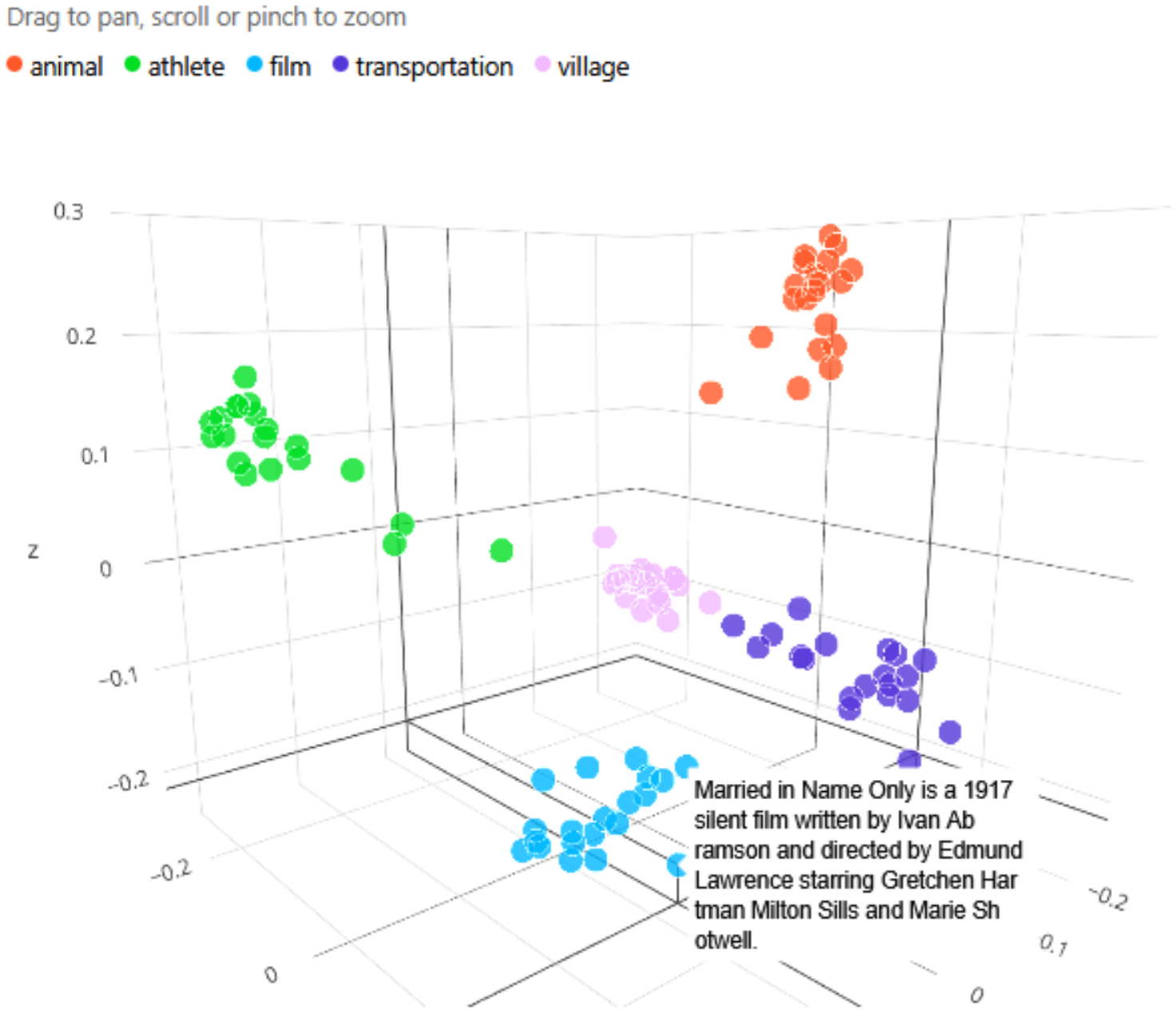
```
Brackets hold the flow,
LINQ dances with sharp logic—
Types bloom, strong and bold.
```

- Br
- Brackets
- Brackets hold
- Brackets hold the
- etc.

flow	0.8
world	0.05
code	0.02
…	…
dog	0



https://openai.com/index/introducing-text-and-code-embeddings/



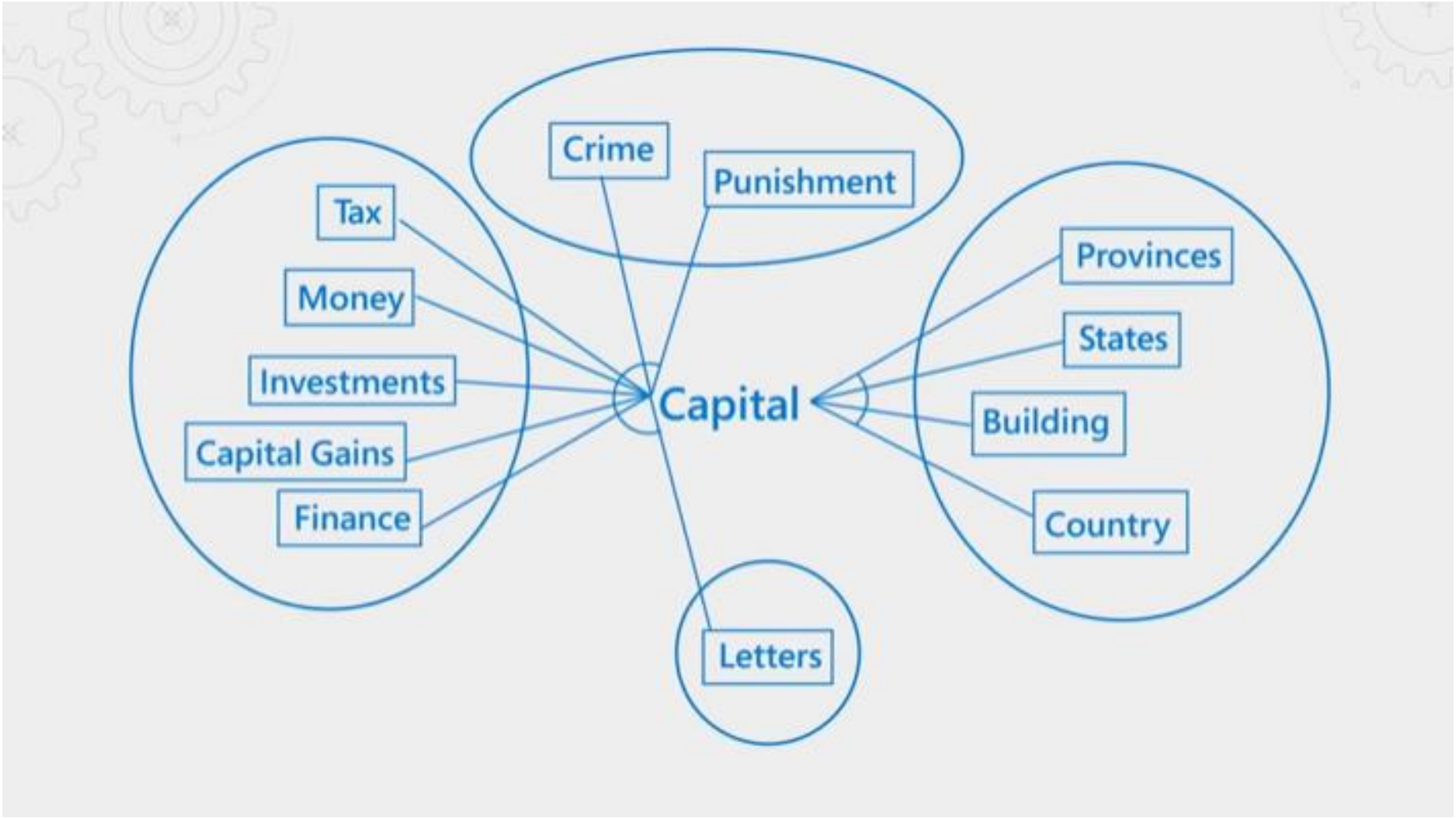
https://openai.com/index/introducing-text-and-code-embeddings/

Attention

- Attention is All You Need, Google, 2017
- “Attention block” – does two primary things

1. Allows vectors to modify each other based on all the other vectors
2. Allows for parallel calculation

“What is the capital of France?”



https://learn.microsoft.com/en-us/azure/search/semantic-search-overview

Context

- The tokens that the LLM keeps in memory to generate new ones
- “Context window”
- Scales quadratically
- 1k tokens - 1000x1000 matrix - **1M vectors**
- 10k tokens - 10000x10000 matrix - **100M vectors**
- 100k tokens - 100000x100000 matrix - **10B vectors**

Training

- Two phases
  - 1. Pre-training. Give it a massive dataset, gets embedded, finds relationship. Get a **base model**
  - 2. Post-training/fine-tuning. Requires at least some manual intervention by humans. Swap out to a different dataset by humans. Get lots of people to write prompts, as well as an ideal response. Get an **assistant model**
- This makes up the **model**

Example: Ollama 2:70b

- **Step 1 – pretraining**

1. Get ~10TB of text from the internet
2. Use ~6,000 GPUs ~$2M, and ~12 days to embed this into vectors in a neural network with rough semantic meaning

- **Step 2 – finetuning**

1. Write labeling instructions for humans
2. Hire lots of humans to write ~100k high quality idea Q&A
3. Use these to “test” the model and refine it

Retrieval Augmented Generation (RAG)

- **Augment** the response of an LLM with info **retrieved** from a data store
- Two main parts

1. Retrieval. User prompts something, system retrieves info from an external knowledge store
2. Generation. Retrieved info is used to augment user's prompt. AI model processes user prompt + retrieved info to produce an enriched response

RAG benefits

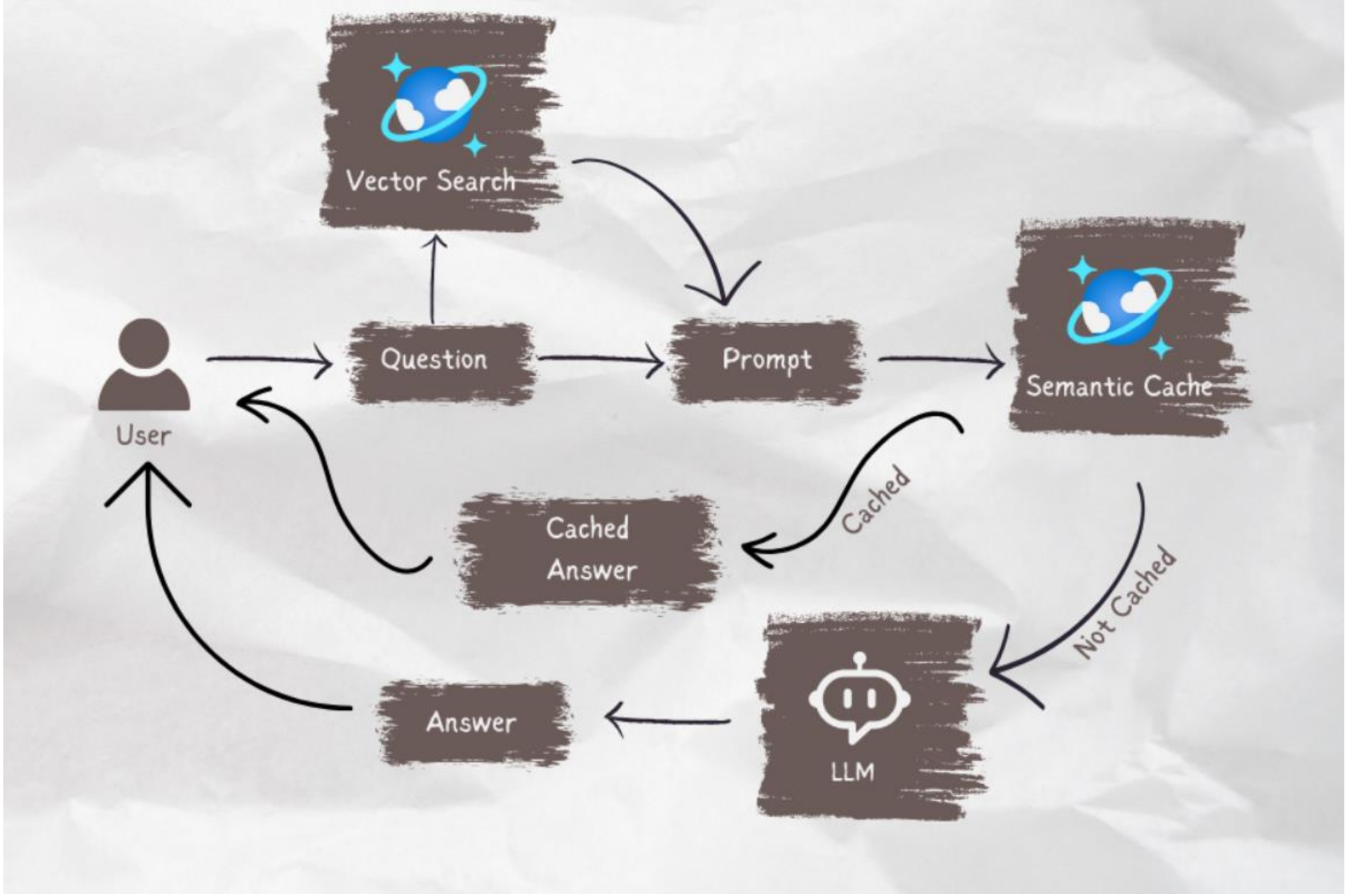
- Improved accuracy
- Reduced hallucinations
- Retrieve up-to-date info
- Get domain-specific knowledge

Agents

- “Agentic”
- 3 parts

1. LLM – model used
2. State – context, helps guide decisions based on past results
3. Tools – like the functions before with the convos, a bridge between the model and external systems

- AI that doesn’t just generate, but takes action on behalf of the user
- Fulfill wide variety of tasks with some/minimal human intervention

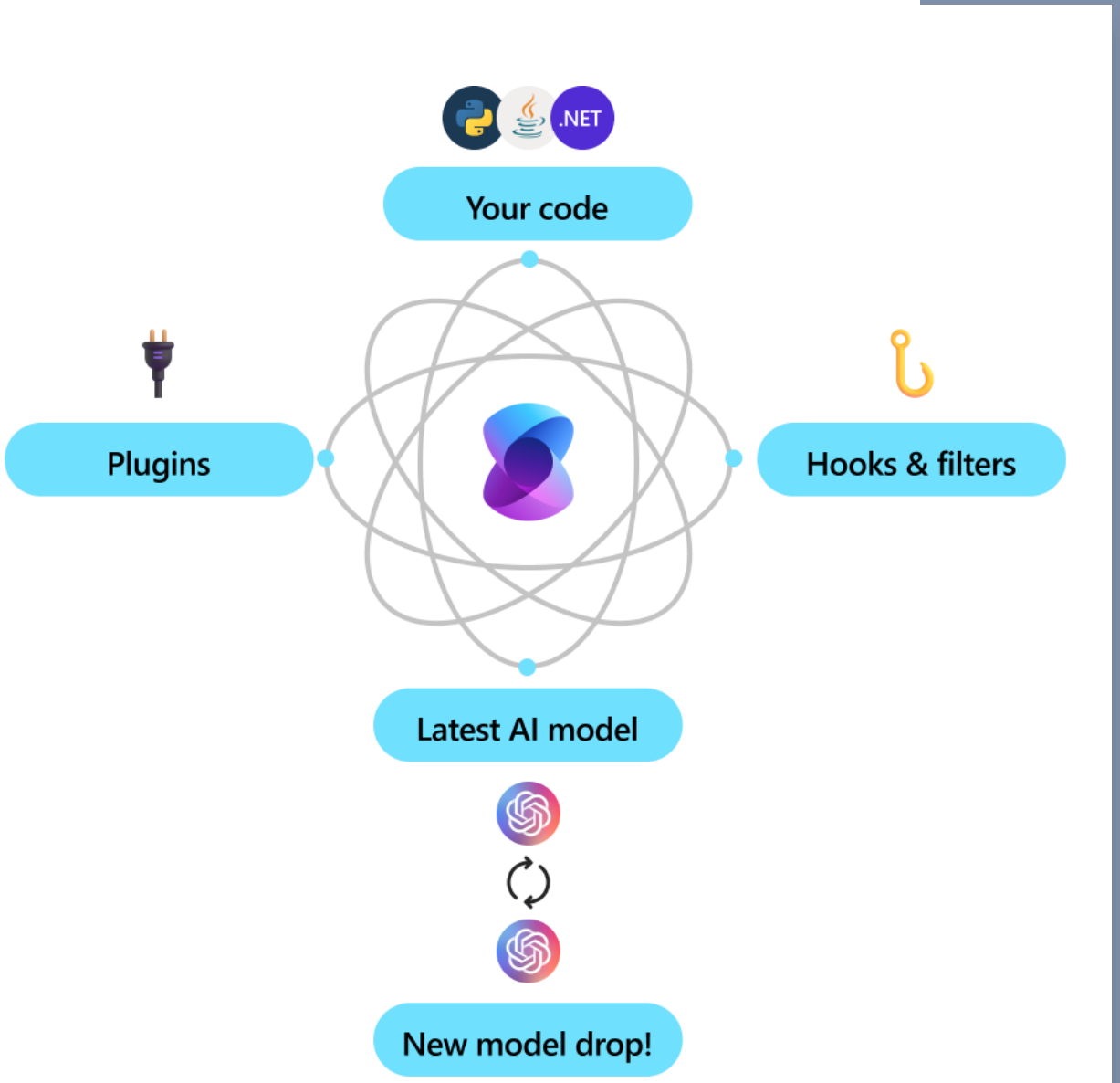


https://learn.microsoft.com/en-us/azure/cosmos-db/ai-agents

Model Context Protocol (MCP)

- **Protocol** for giving **models** more **context**
- Problem: models are generalized + not connected to our tools
- Solution: MCP is a **standardized** way to connect LLMs to tools (APIs, CLIs, databases, agents, etc.)

Semantic Kernel



https://learn.microsoft.com/en-us/semantic-kernel/overview/

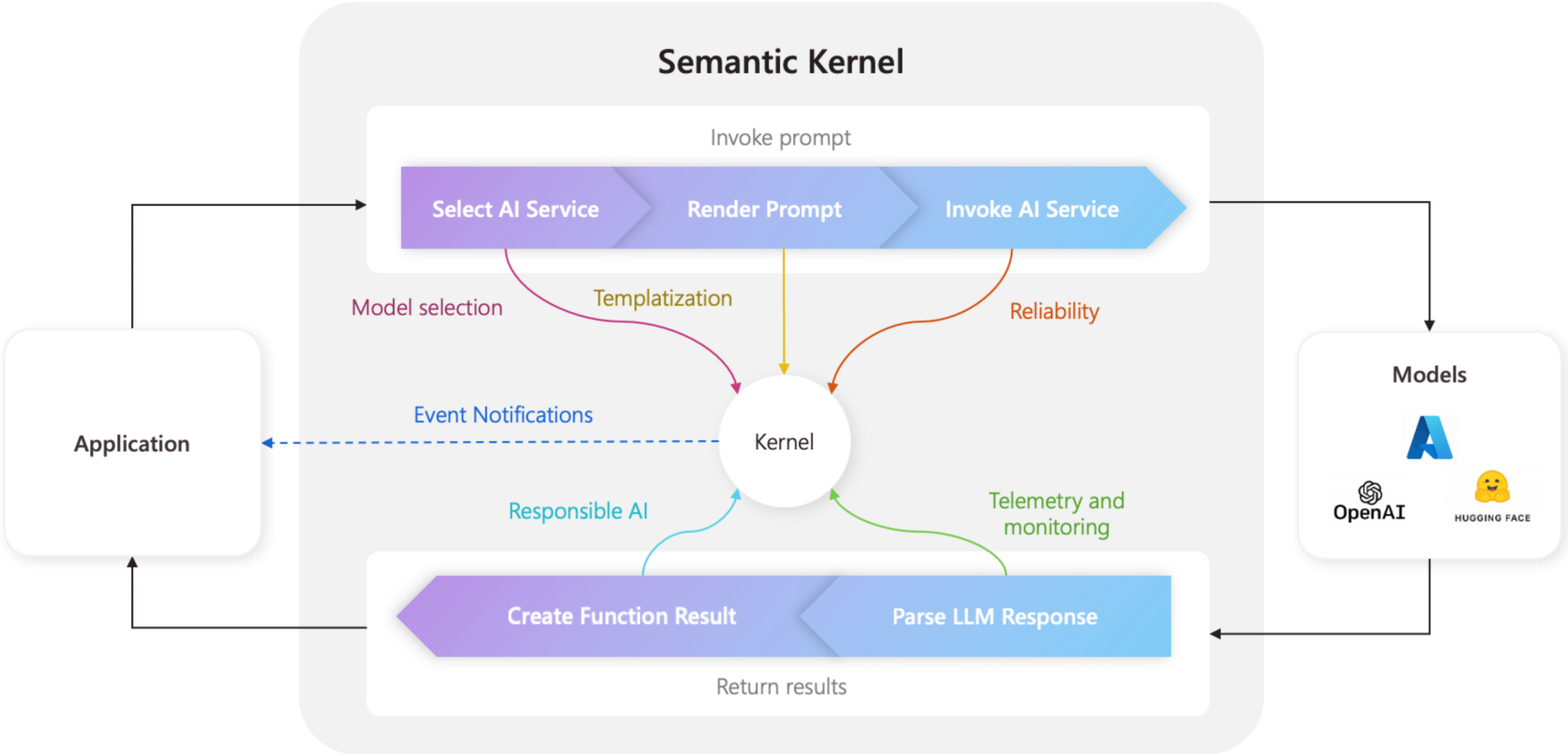
Semantic Kernel

- SDK for building AI agents and integrating AI models into your apps
- Microsoft, open-source
- C#, Java, Python
- Bridge between normal AI usage and your code, which helps simplify process of developing AI-powered apps

What makes up Semantic Kernel?

- Kernel itself is basically a DI container. Central location to configure/monitor agents
- Two types of components

1. Services. AI services like chat completion, but also logging, HTTP clients, etc. Modeled after existing service provider in .NET
2. Plugins. Components used by your AI services to perform work, like retrieving data from a database or calling an external API



https://learn.microsoft.com/en-us/semantic-kernel/concepts/kernel?pivots=programming-language-csharp

Why Semantic Kernel?

- Devs normally have to learn the different APIs/quirks of each service
- SK abstracts and unifies APIs
- Must support function calling
- Bridge between your code and LLM

Coding demo

Get weather | Send email | Search knowledge base

Function calling example with get_weather function — javascript

```javascript
import { OpenAI } from "openai";

const openai = new OpenAI();

const tools = [{
    "type": "function",
    "name": "get_weather",
    "description": "Get current temperature for a given location.",
    "parameters": {
        "type": "object",
        "properties": {
            "location": {
                "type": "string",
                "description": "City and country e.g. Bogotá, Colombia"
            }
        },
        "required": [
            "location"
        ],
        "additionalProperties": false
    }
}];

const response = await openai.responses.create({
    model: "gpt-4.1",
    input: [{ role: "user", content: "What is the weather like in Paris today?" }],
    tools,
});

console.log(response.output);
```

https://platform.openai.com/docs/guides/function-calling?api-mode=responses&lang=javascript

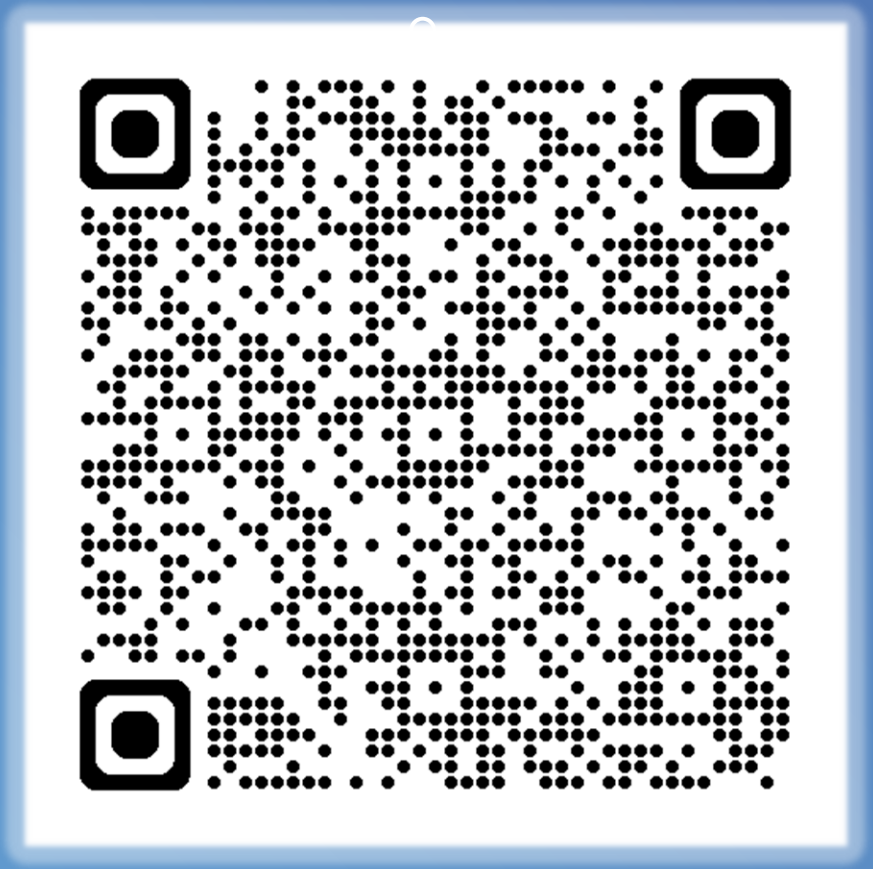
Responsible AI

1. **Fairness**: AI systems should treat all people fairly.
2. **Reliability and safety**: AI systems should perform reliably and safely.
3. **Privacy and security**: AI systems should be secure and respect privacy.
4. **Inclusiveness**: AI systems should empower everyone and engage people.
5. **Transparency**: AI systems should be understandable.
6. **Accountability**: People should be accountable for AI systems.

Resources

- 3Blue1Brown Neural networks playlist
- Andrej Karpathy Intro to Large Language Models
- Microsoft Github samples - Getting Started
- Semantic Kernel Microsoft docs and Microsoft Learn course
- GitHub repo of MCP servers

Thank you!



Daniel Ward