LLMs

How they actually work,
What they can (and can't) do, and
Why they're an important tool

(for the field of software development)

Disclaimers!

Am I worried about AI displacing humans?

Yes.

Do I think AI is appropriate for all industries?

No.

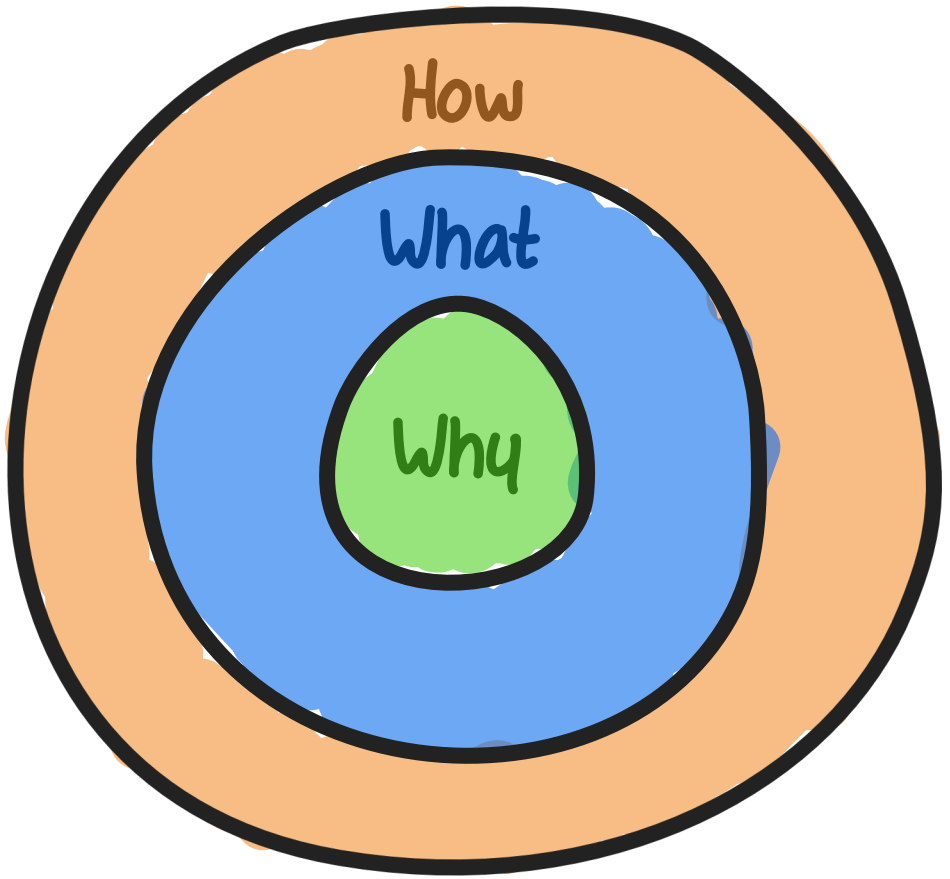
Does AI cause brain rot?

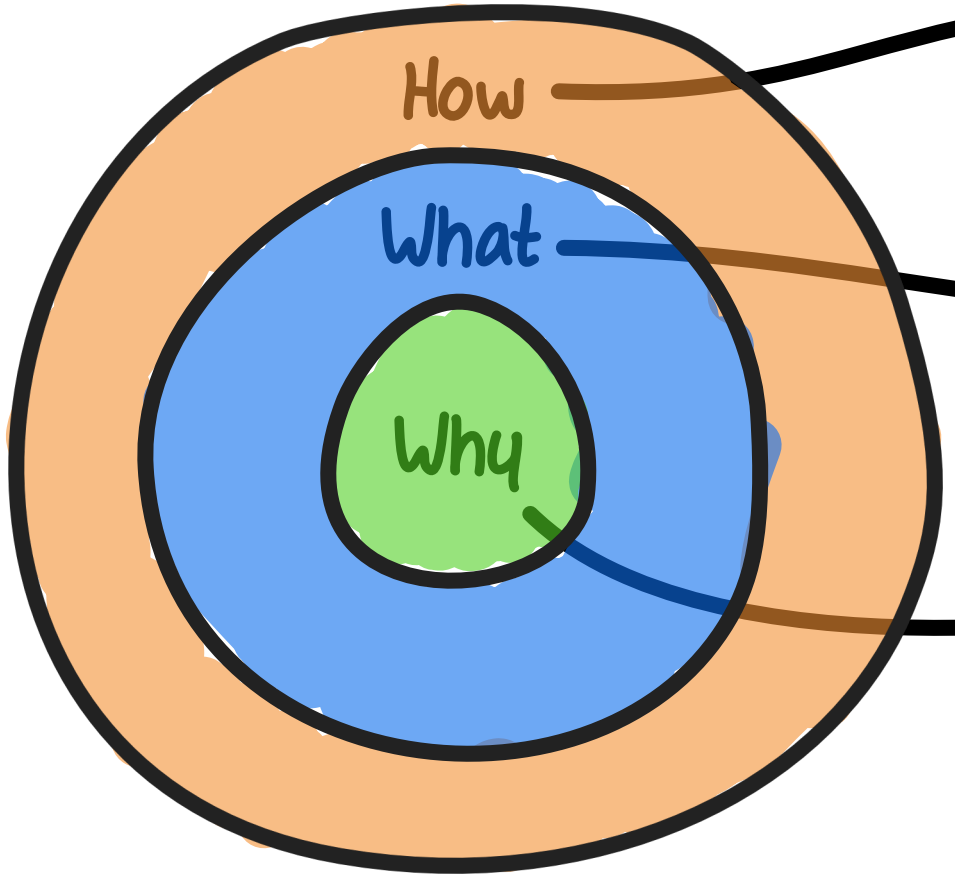
Probably, if you use it wrong.

AI is an important new tool,
and we need to learn how to
evaluate and utilize it effectively.

The Framework

for evaluating a new tool





How do LLMs actually work?

↳ Demystify the internals of LLMs

What are they good at?

↳ A tool we can use, not a silver bullet.

Why are they powerful tools for our work?

↳ and not a replacement for human engineers.

How

LLMs actually work

KEY TAKEAWAYS

1. Understand the path we took to get to the Large Language Model (LLM) architecture
2. Understand that LLMs are math, not magic.

to better understand

what

they're capable of

The Big Question

that kicked off a trillion-dollar industry

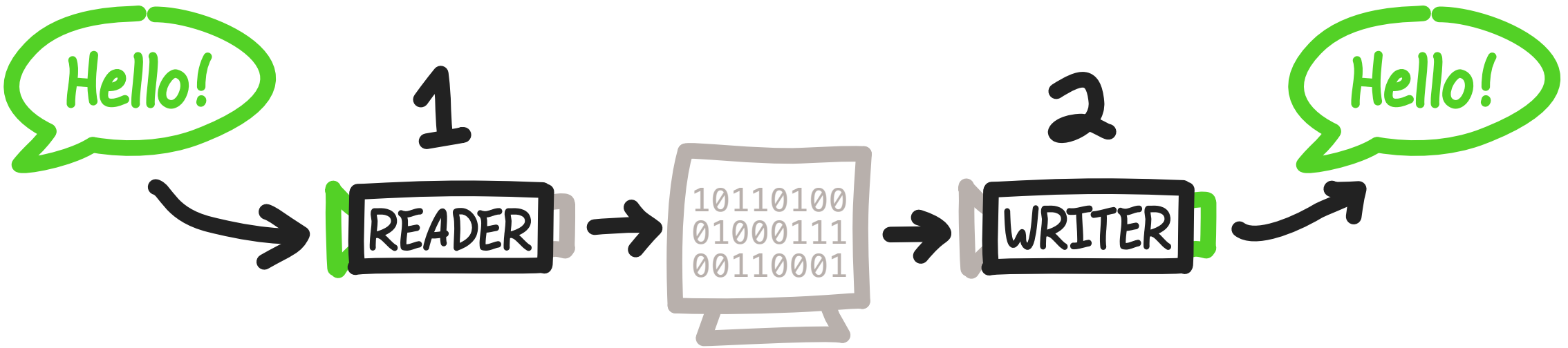
1. You and I communicate with natural language
2. Natural language is flexible and ambiguous
3. Computers are terrible at understanding anything flexible and ambiguous

How would you write a software tool that can read & write natural language?

Architecture of

Natural Language Processing

1. Read natural language into data
2. Write data into natural language



GOAL: Given an input string in natural language, translate it into data that encodes its meaning.

Q's:
1. How can we capture the intent of the input?
2. What data structure stores this meaning in a way the computer can understand?

"How do I write a Java menu to display soup recipes?"

Q's:

1. How can we capture the intent of the input?
2. What data structure stores this meaning in a way the computer can understand?

BRONZE

Keyword Extraction

(~1960s)

SILVER

Artificial Neural Networks

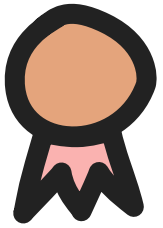
(~1990s)

GOLD

♂ Transformers & LLMs

(~2020s)

The "GPT" in "ChatGPT" stands for "Generative Pre-Trained Transformer"



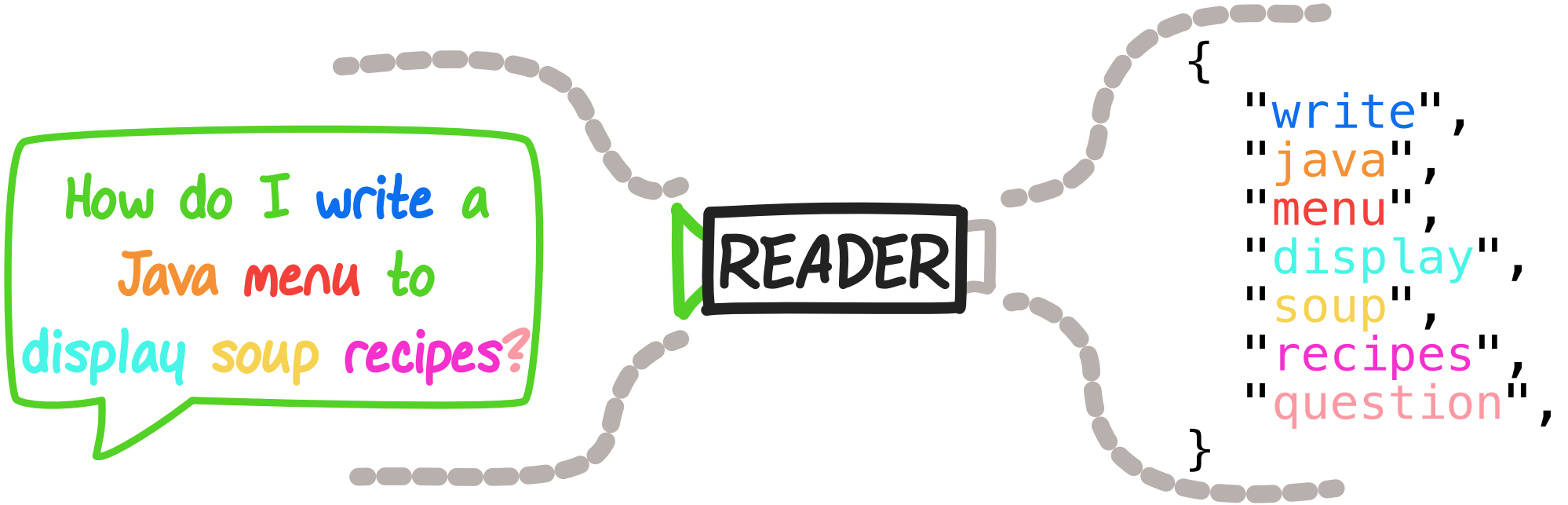
Keyword Extraction

extract the important tokens from text

How do I write a Java menu to display soup recipes?

READER

```
{
  "write",
  "java",
  "menu",
  "display"
  "soup",
  "recipes",
  "question"
}
```



STRENGTHS

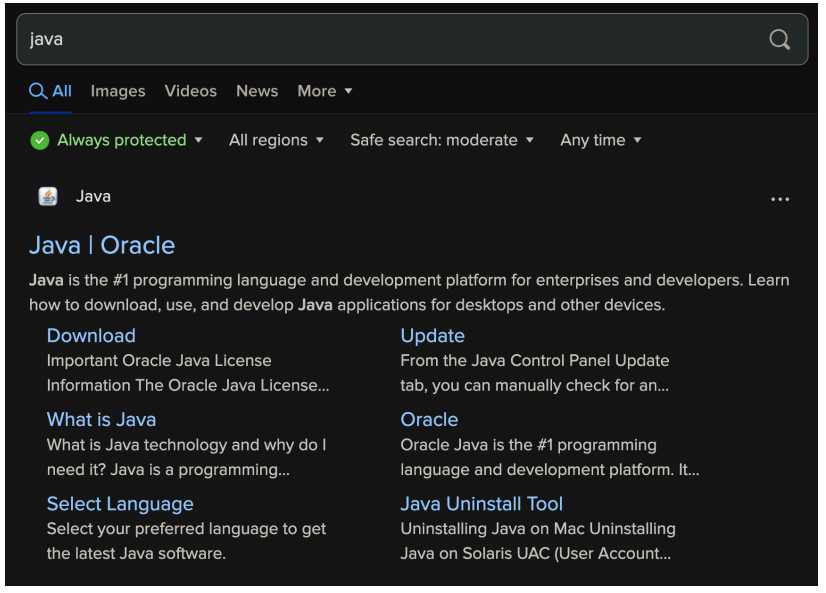
- ✓ Translates the text into a machine-readable structure
- ✓ Stores some explicit topics

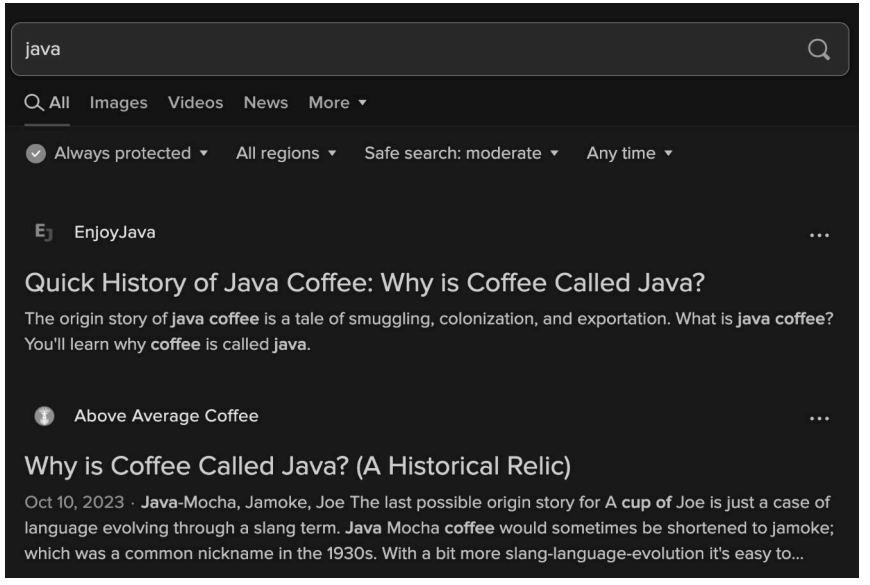
WEAKNESSES

- ✗ Cannot differentiate ambiguous tokens
- ✗ Captures syntax, not symantic meaning

Ambiguity in Text

Can the data differentiate ambiguous meaning?





There is a world of difference between

Java (the programming language) and

Java (a cup of coffee)

Recollection Issues

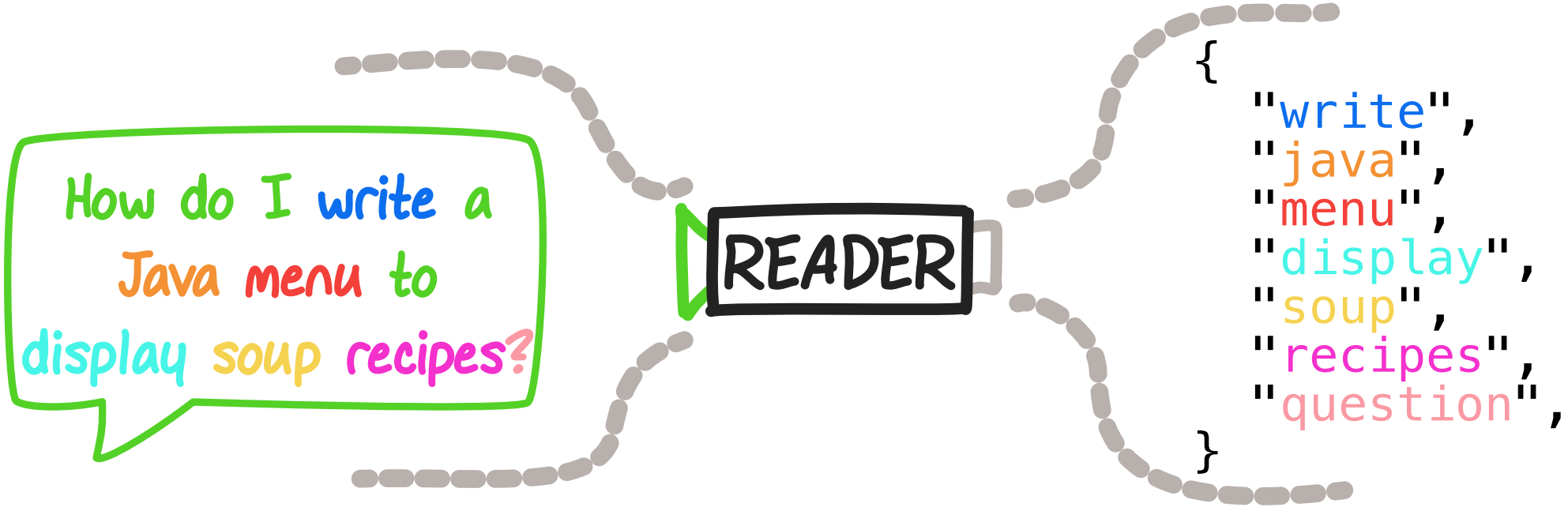
Can the data structure accurately reconstruct the original prompt?

How do I write a Java menu to display soup recipes?

What is a good writing style for the menu display of a Java shop that also serves vegan soup recipes?

```
{
  "write",
  "java",
  "menu",
  "display",
  "soup",
  "recipes",
  "question",
}
```

Keyword extraction can capture syntax, but not necessarily semantic meaning



OPTIMIZATIONS

- How could we represent relationships between tokens to help specify ambiguous language?
- How could we capture semantic meaning by "reading between the lines?"

Neural Networks

Turning explicit keywords into implicit context

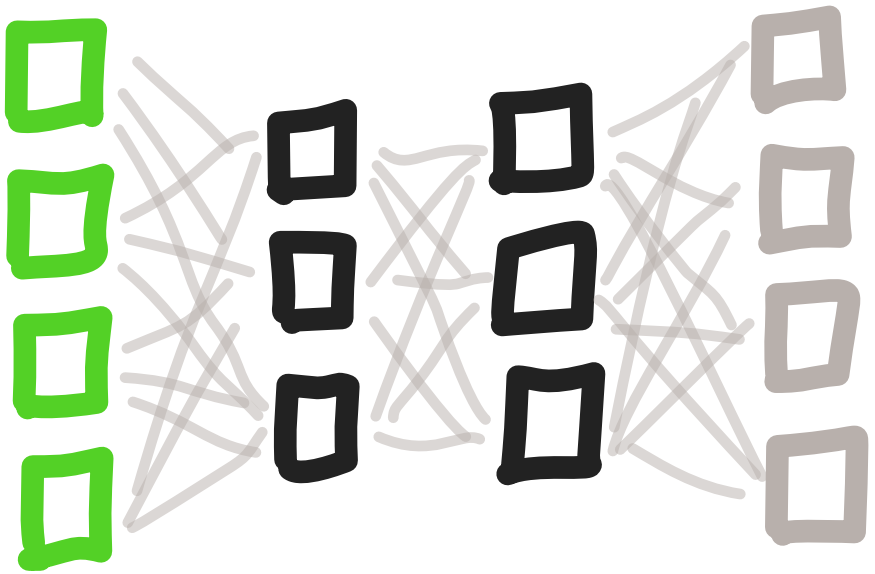
How do I write a Java menu to display soup recipes?

READER

```
{
  "explicit": {...}
  "implicit": {
    "programming": 85%
    "frontend": 77%
    "ingredients": 43%
    ...
  }
}
```

Anatomy of Neural Networks

The input layer takes in tokens from the prompt



The output layer outputs strengths of implicit context

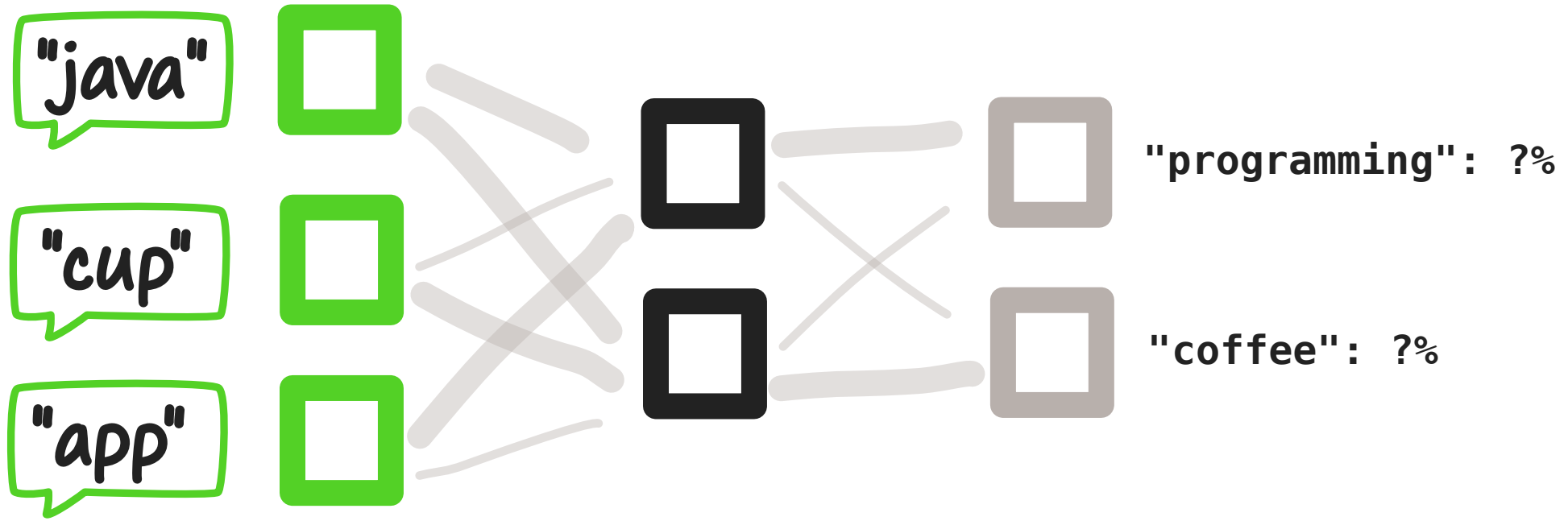
♂ Hidden layers and connection weights can be used to "read between the lines"

3blue1brown

Explanation of Neural Networks

Example:

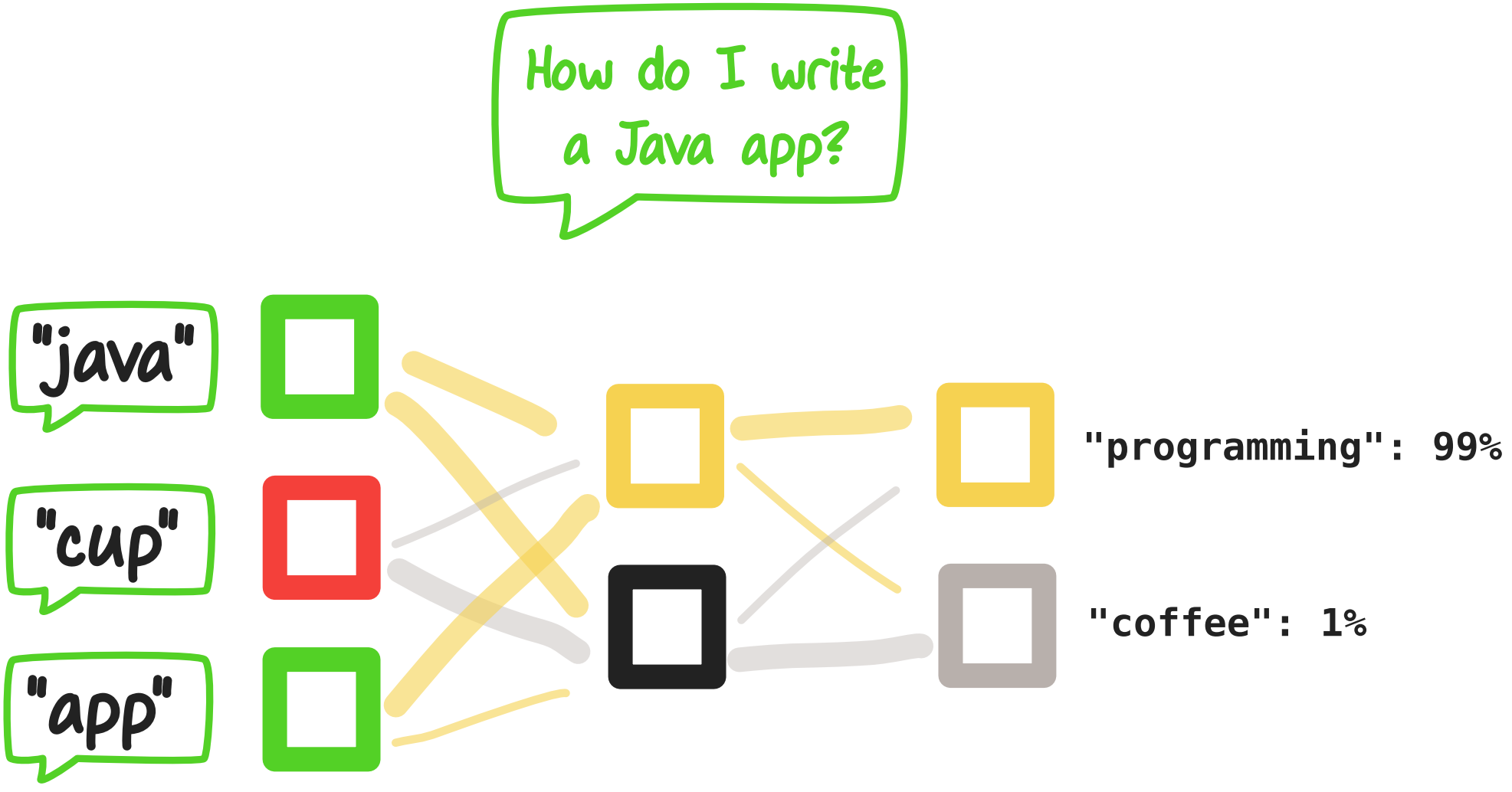
Barista or Engineer?

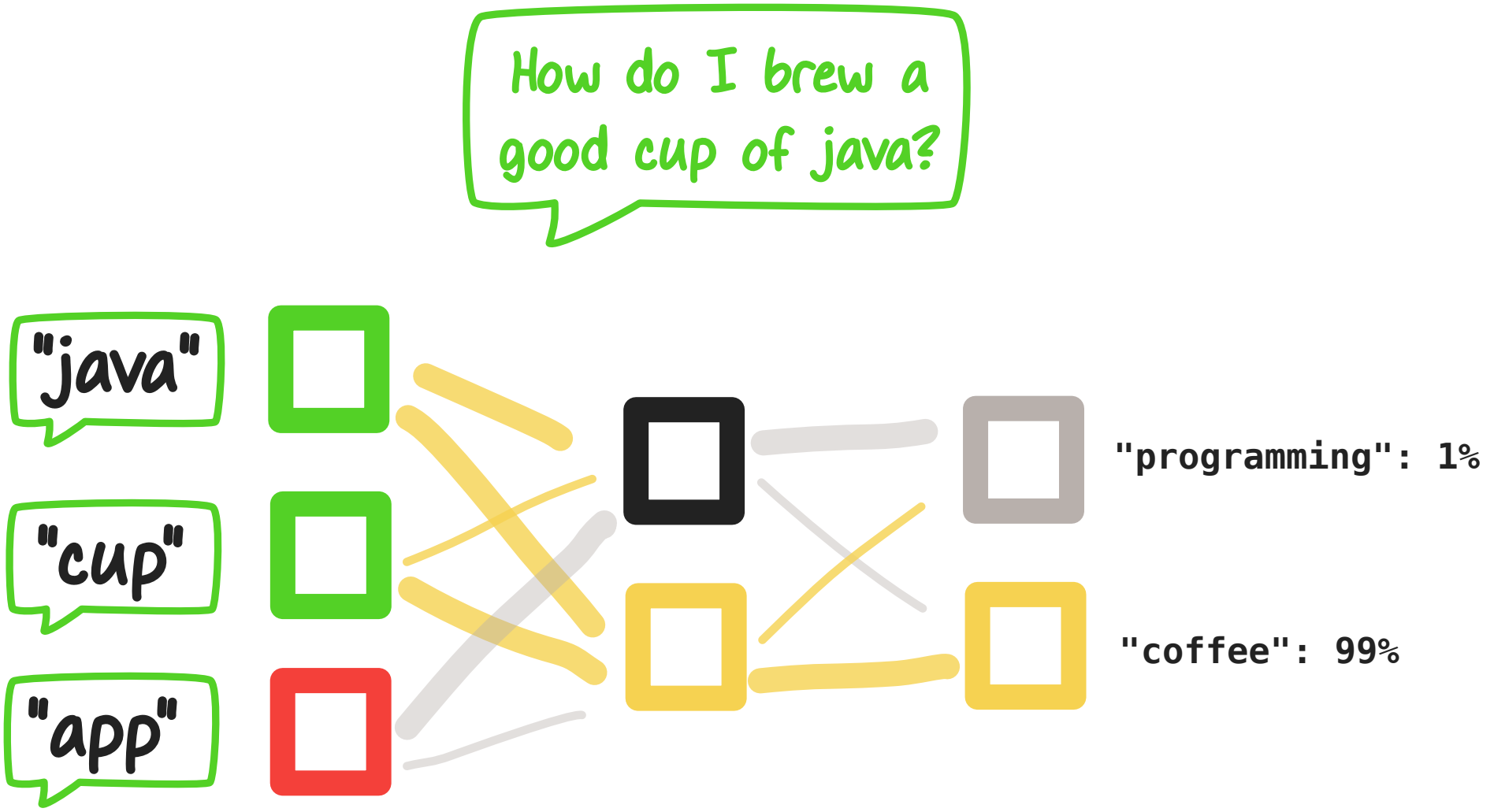


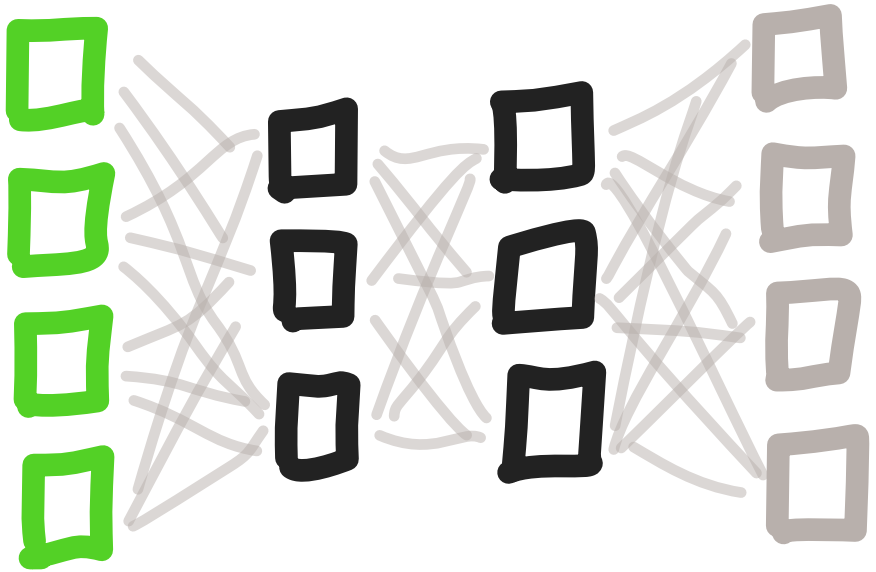
① Network reads the inclusion of certain prompt input tokens

② Weighted connections, learned from training, activate in the network

③ Network outputs connection strength of implicit context


How do I write a Java app?
"java"
"cup"
"app"
"programming": 99%
"coffee": 1%


How do I brew a good cup of java?
"java"
"cup"
"app"
"programming": 1%
"coffee": 99%



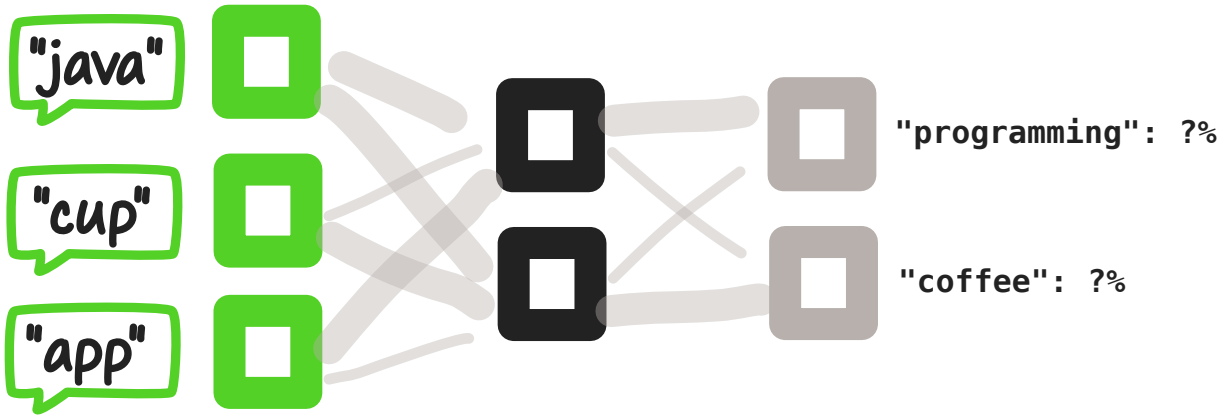
STRENGTHS

- ✓ Ability to link concepts into implicit context
- ✓ Addition of "strength" introduces richer data

WEAKNESSES

- ✗ Network structure is pre-determined
- ✗ Minor changes require full re-training
- ✗ Complexity does not scale well

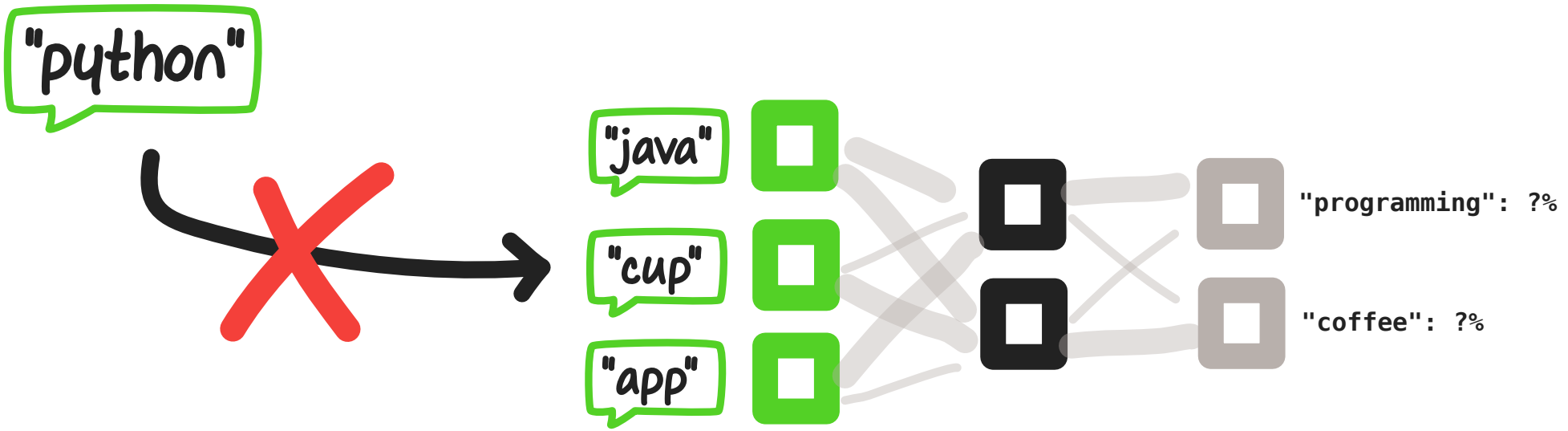
Pre-determined Structure



This network structure is determined by hand, before training, and cannot be modified later.

(this method and training structure is called supervised learning)

Rigid Networks



The network cannot accept unknown tokens, and must be entirely restructured and retrained to add new ones.

(this also applies to output and hidden layers)

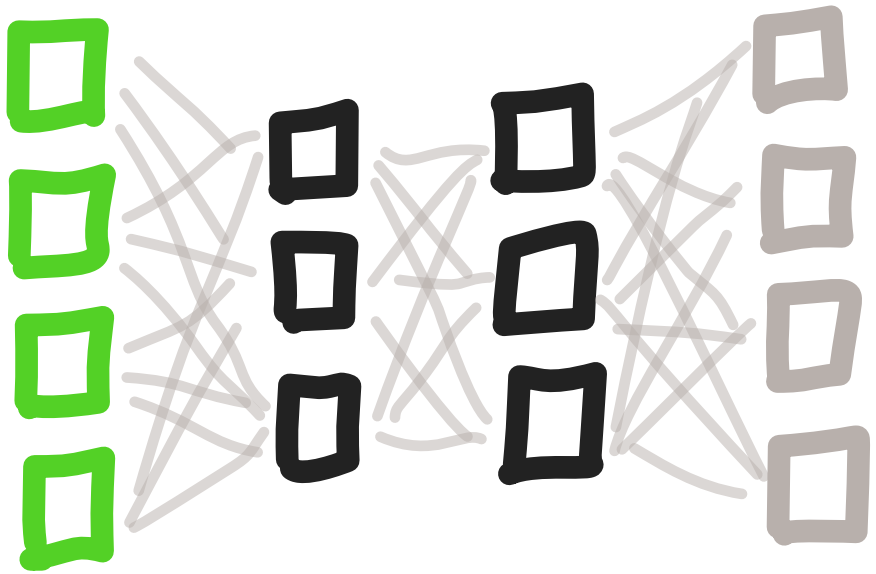
Scaling Issues

10 connections

33 connections

These networks are dense, and scale aggressively as you add more nodes.

(for context, the English Wiktionary contains ~1,500,000 words)

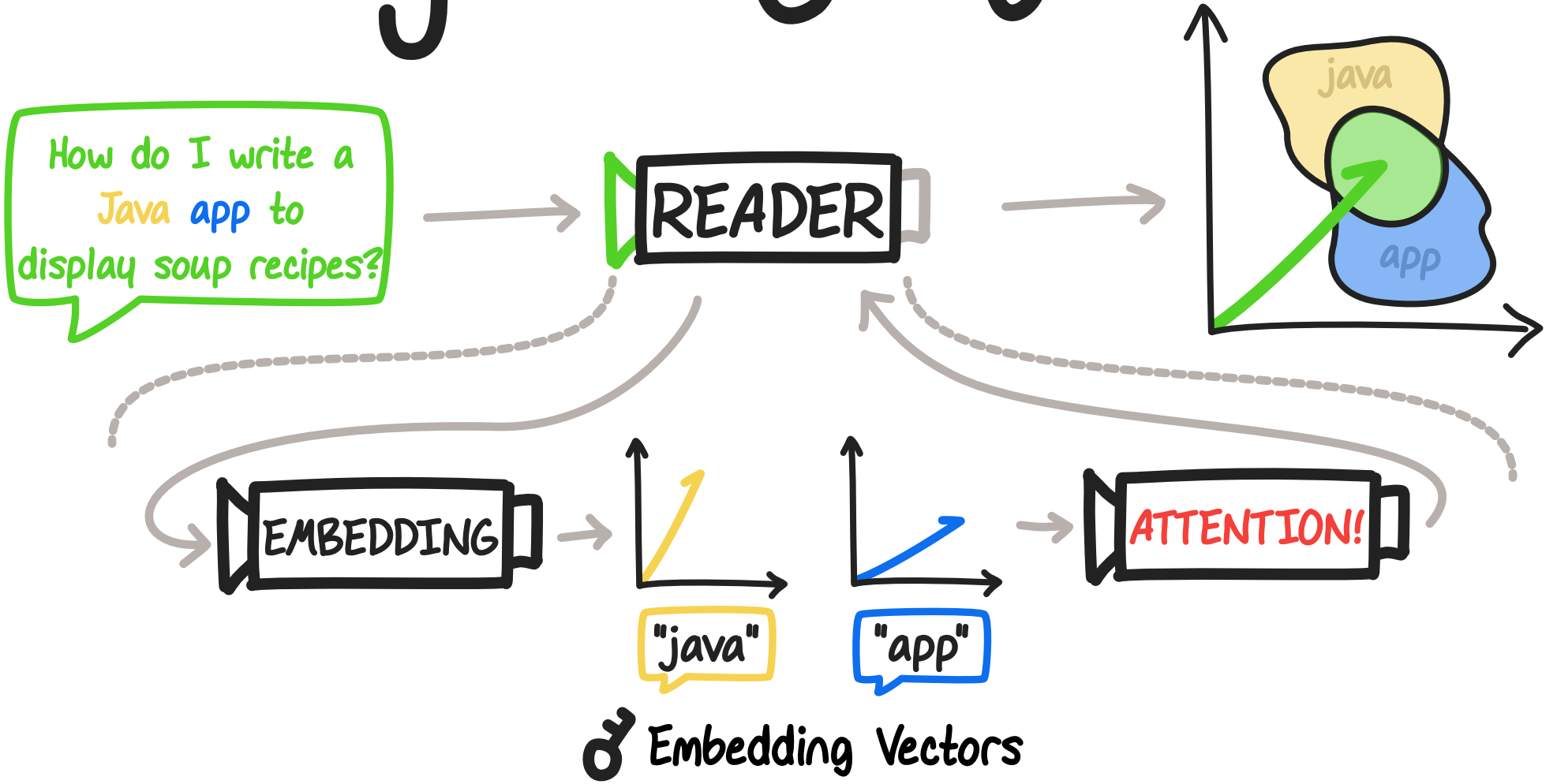


This concept is called "unsupervised learning"

OPTIMIZATIONS

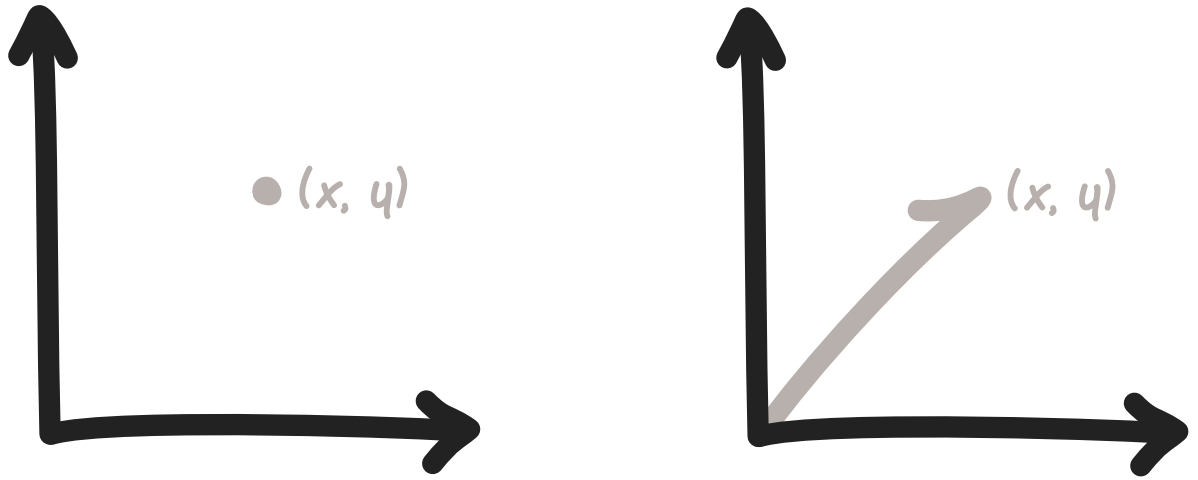
- How could we create a more flexible network that can accept and learn from new tokens?
- How could we create a network that can scale to the size of a whole language?

Large Language Models

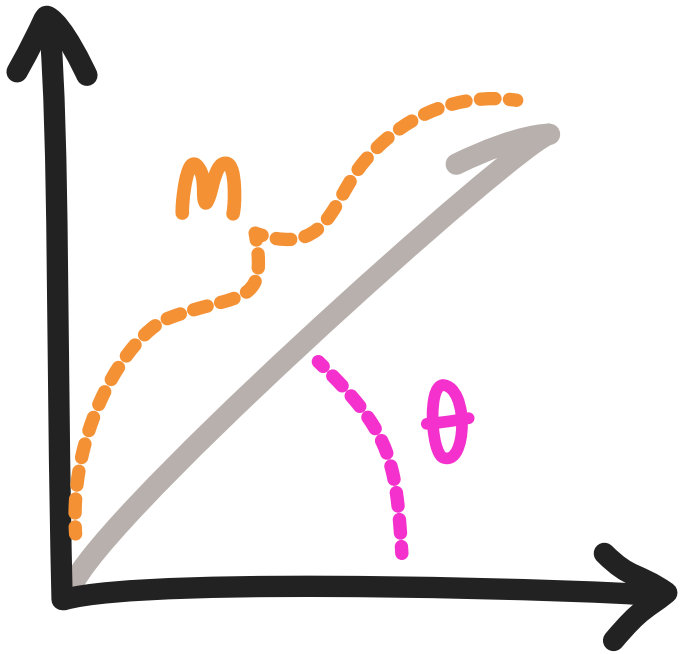


The Power of VECTORS

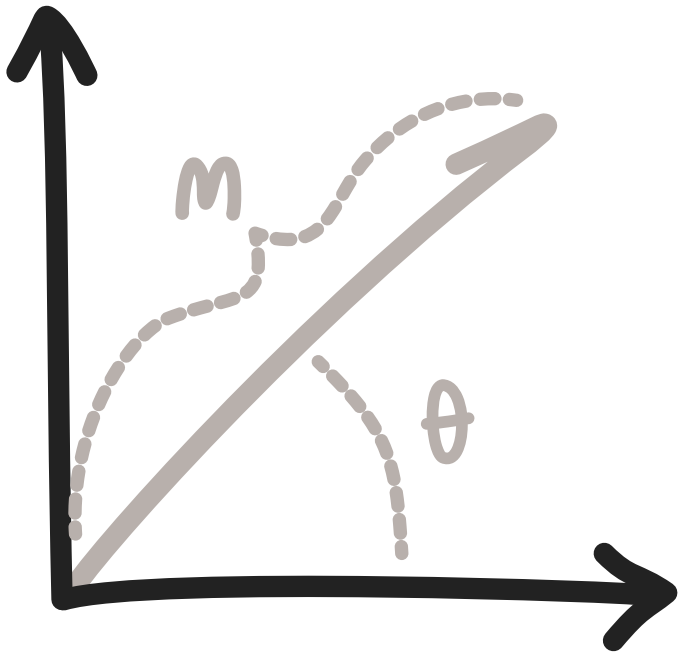
(both direction and magnitude!)



Vectors are simply a way that we can represent some point in a space



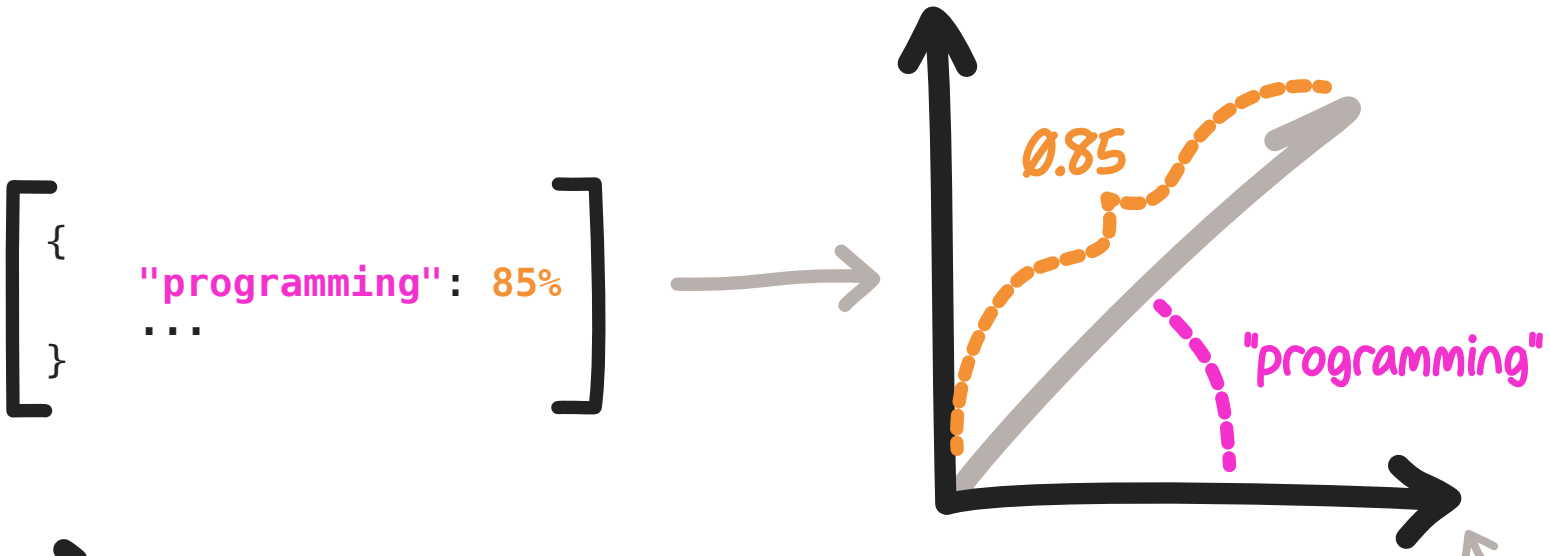
We can define vectors using
a direction, and a magnitude.



θ = undefined
m = undefined

Vectors don't actually mean anything until we assign them a meaning.

Embedding Vectors



I am keeping these axes purposely unlabelled, as their actual meaning is incredibly abstract and not important to the broader concept.

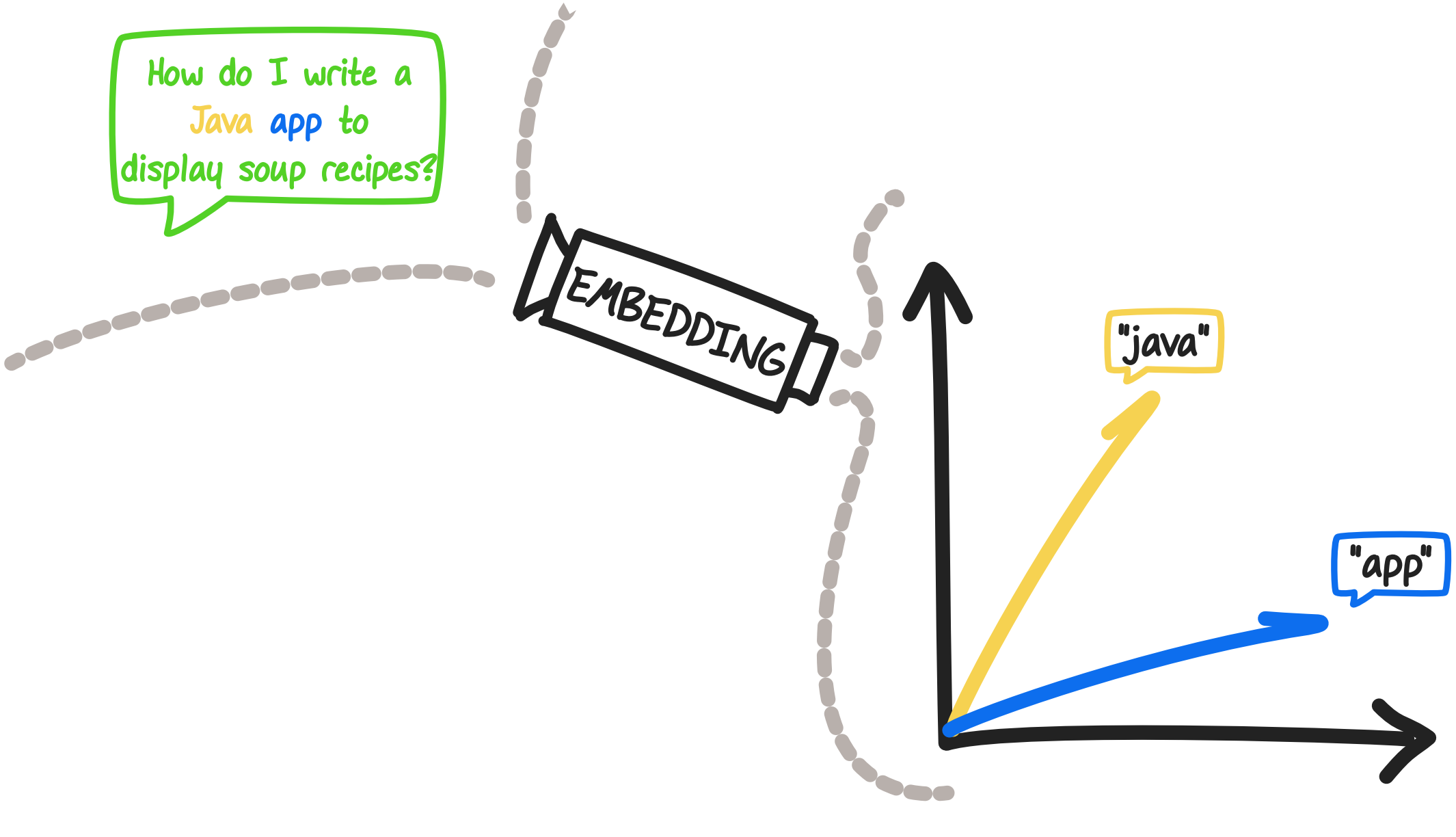
If you are curious, researching "embedding space feature dimensions" is a good place to start.

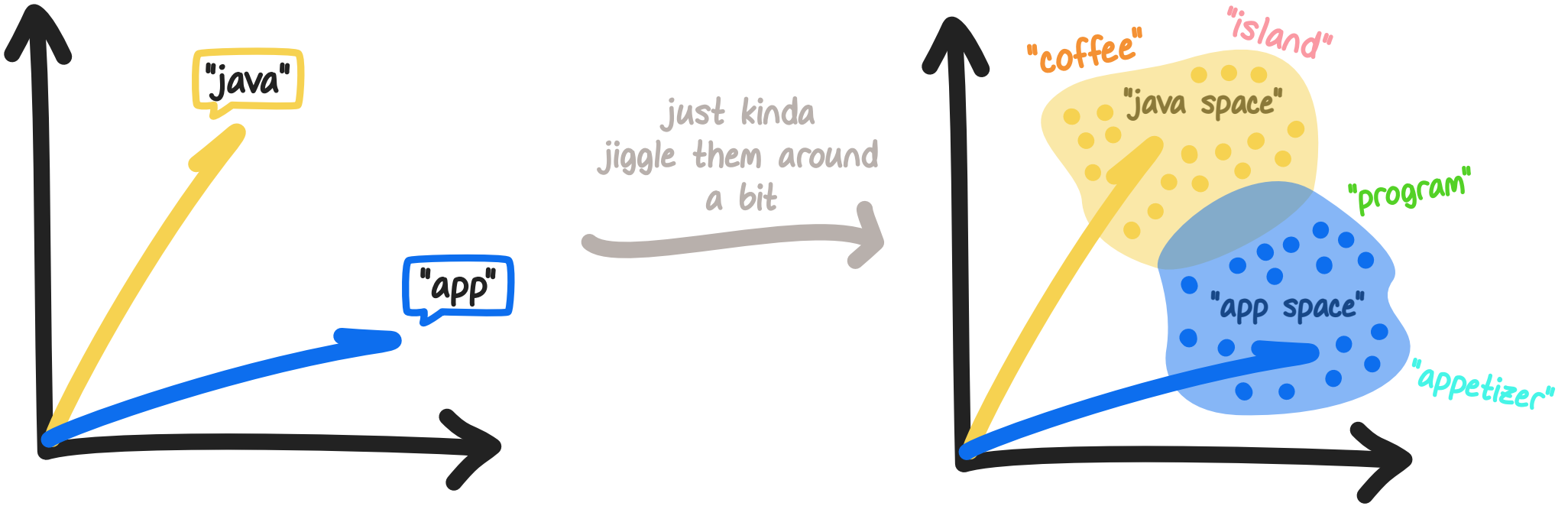
We can assign the token as a direction and strength as magnitude.

This creates a system for translating natural language into embedding vectors

3blue1brown

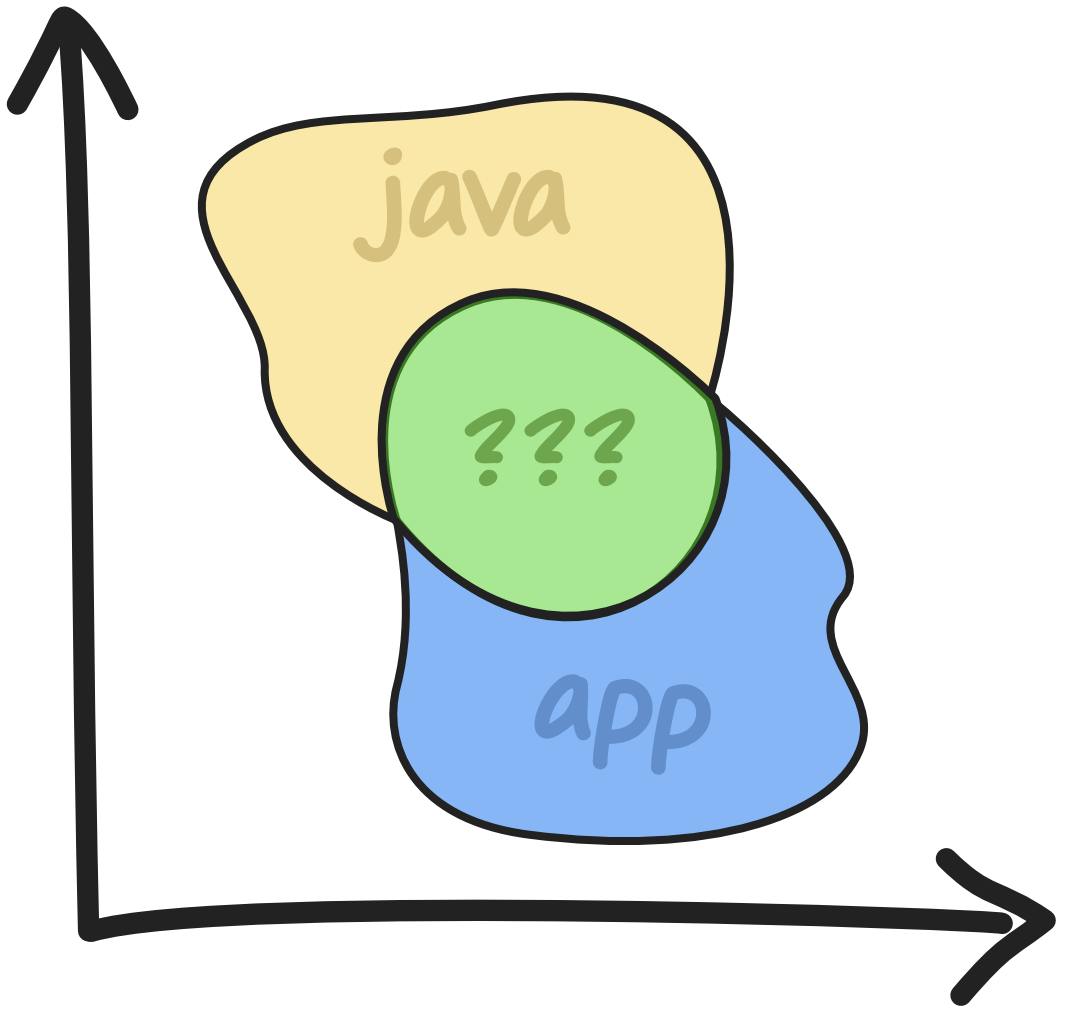
Explanation of Embedding Vectors

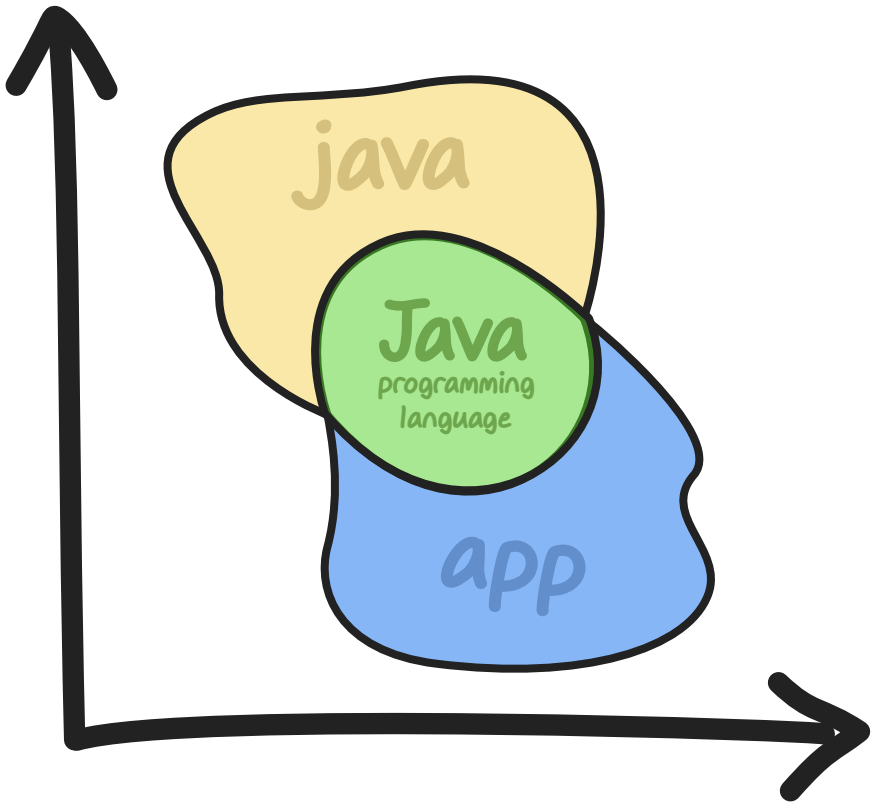

How do I write a Java app to display soup recipes?
EMBEDDING
"java"
"app"



Minor changes to a vector results in minor changes to the concept it embeds.

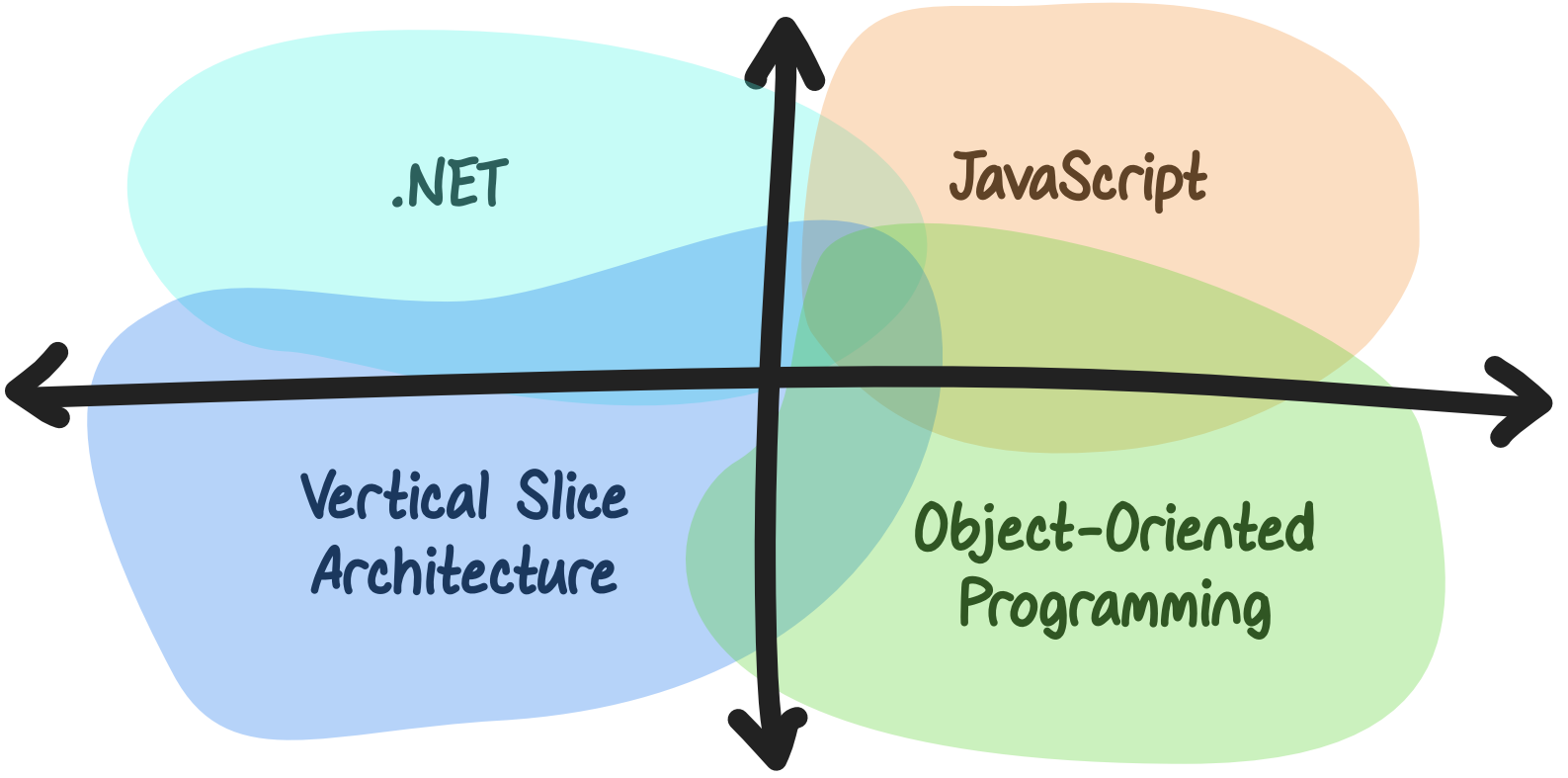
This creates areas in space that can represent different, but related, concepts.


java
???
app

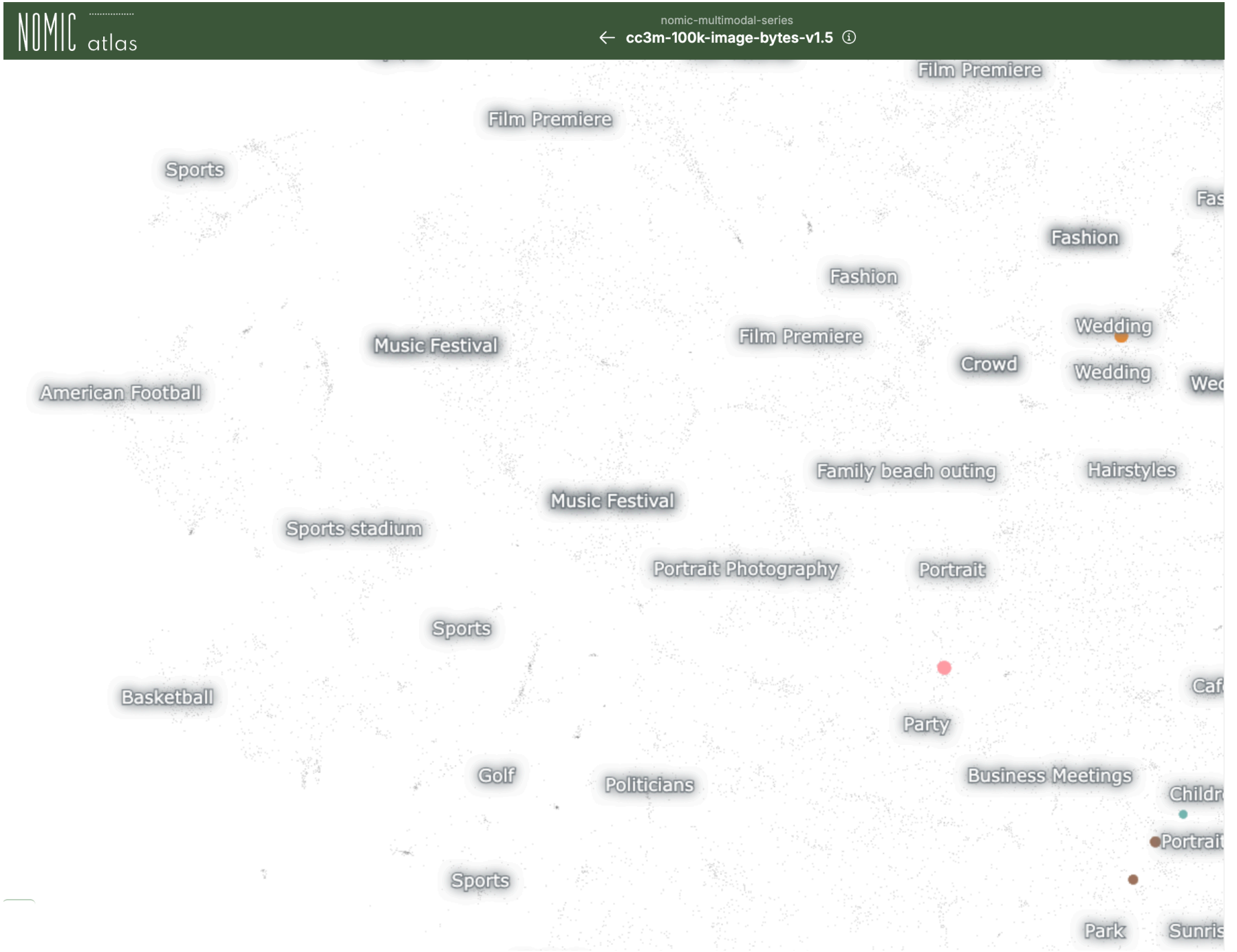


Intersections in these spaces
represent areas where
the concepts overlap.

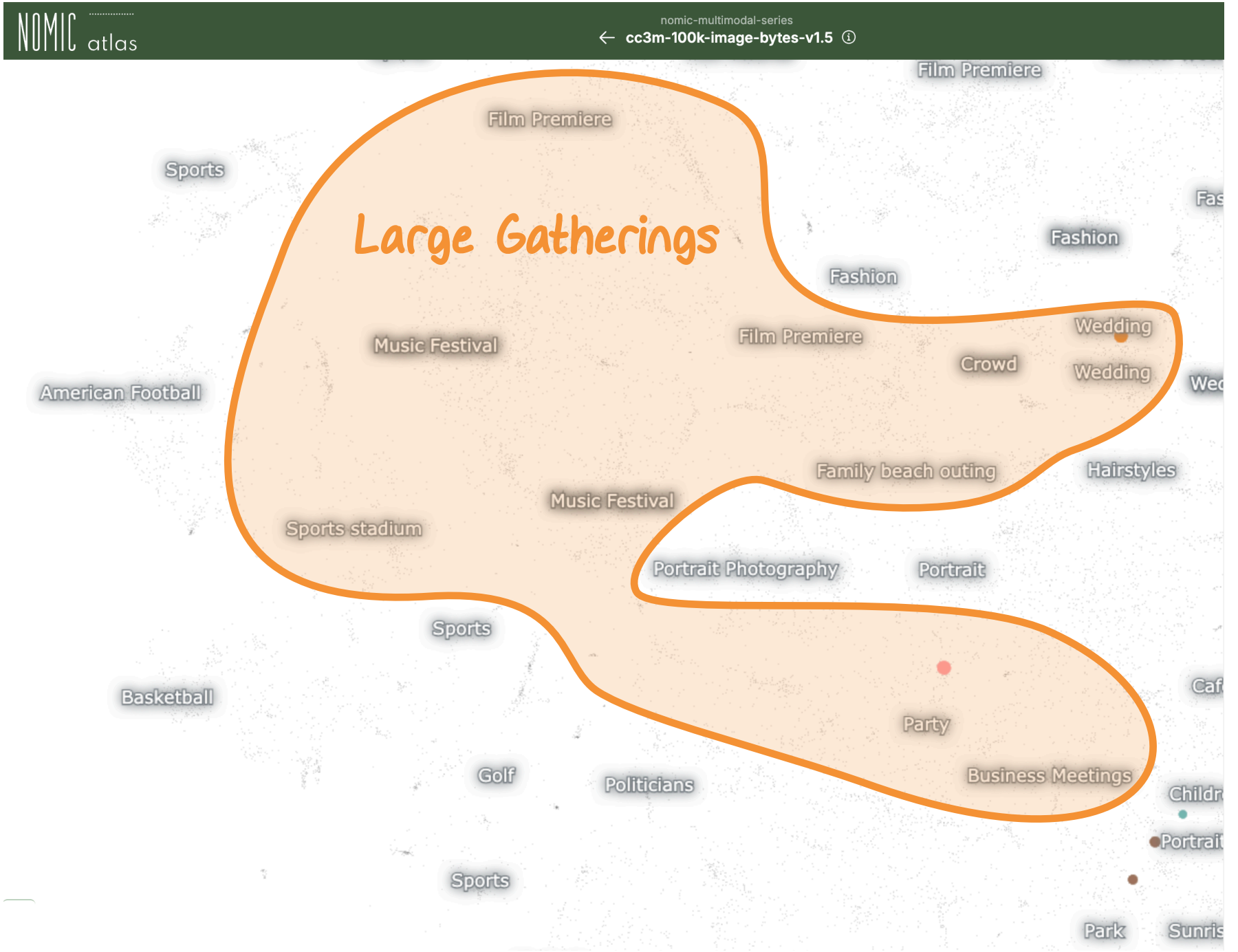
Embedding Space

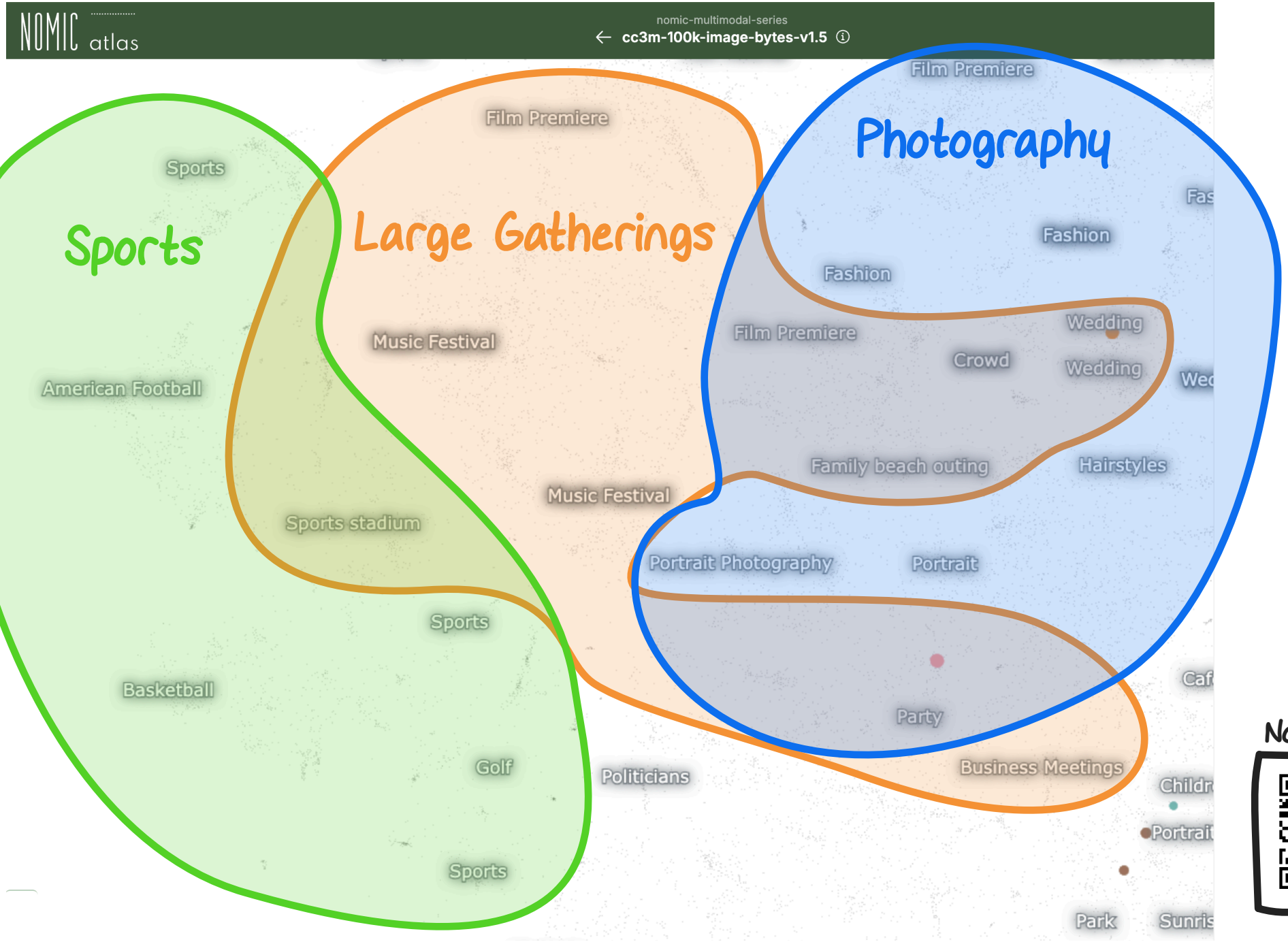


The embedding space
(what tokens exist, and their relationships)
is a large part of what the model "learns" during training.


NOMIC atlas
nomic-multimodal-series
cc3m-100k-image-bytes-v1.5
Film Premiere
Film Premiere
Sports
Fashion
Fashion
Wedding
Film Premiere
Music Festival
Crowd
Wedding
American Football
Family beach outing
Hairstyles
Music Festival
Sports stadium
Portrait Photography
Portrait
Sports
Basketball
Party
Golf
Politicians
Business Meetings
Portrait
Sports
Park

Nomic Atlas


NOMIC atlas
nomic-multimodal-series
cc3m-100k-image-bytes-v1.5
Large Gatherings
Film Premiere
Film Premiere
Sports
Fashion
Fashion
Wedding
Film Premiere
Music Festival
Crowd
Wedding
American Football
Family beach outing
Hairstyles
Music Festival
Sports stadium
Portrait Photography
Portrait
Sports
Basketball
Party
Golf
Politicians
Business Meetings
Sports
Park
Nomic Atlas


NOMIC atlas
nomic-multimodal-series
cc3m-100k-image-bytes-v1.5
Sports
Large Gatherings
Photography
Sports
Film Premiere
Film Premiere
Fashion
Fashion
Music Festival
Film Premiere
Wedding
Crowd
Wedding
American Football
Family beach outing
Hairstyles
Music Festival
Sports stadium
Portrait Photography
Portrait
Sports
Party
Basketball
Golf
Politicians
Business Meetings
Sports
Park
Nomic Atlas

Previous Resources

3blue1brown

Explanation of Neural Networks

3blue1brown

Explanation of Embedding Vectors

Nomic Atlas

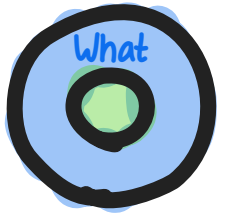
Visualization of embedding spaces

This is a good one to explore during the break!

Next Up

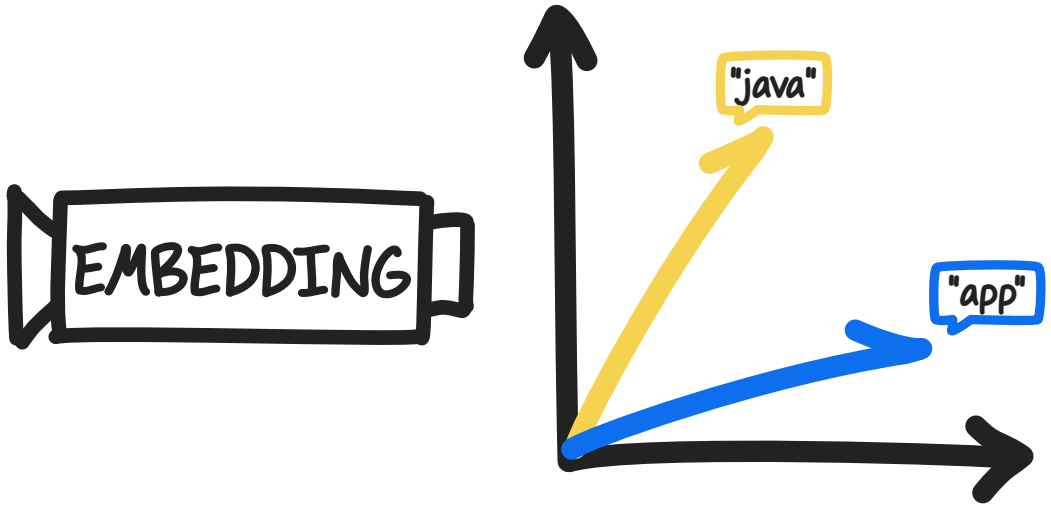
How LLMs transform embeddings into data representations

How LLMS generate token outputs



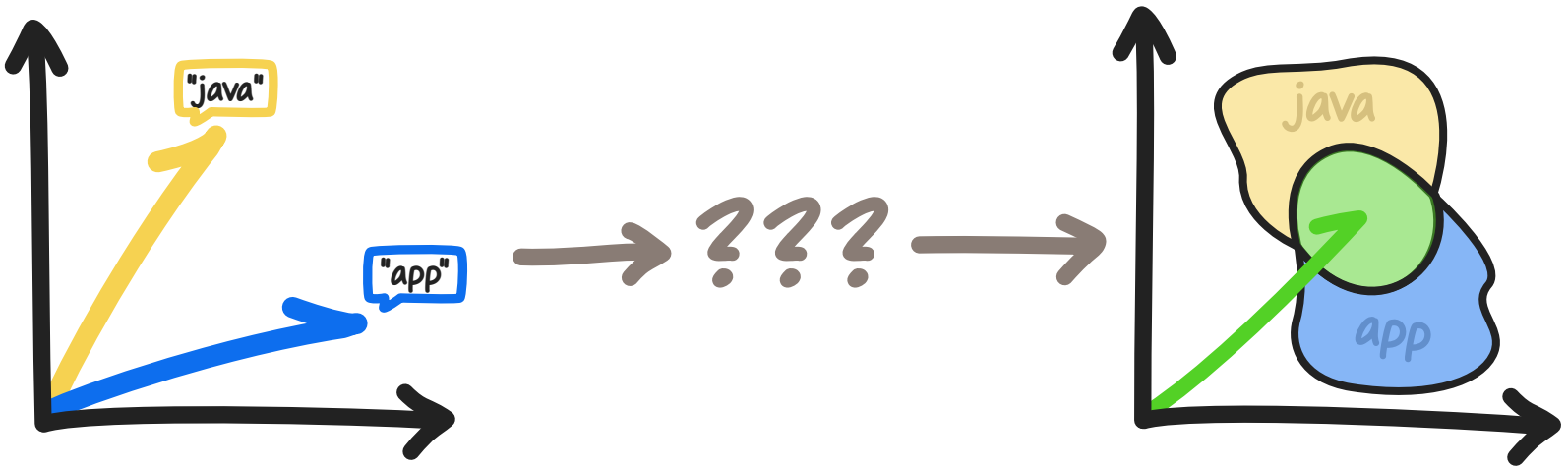
What LLMs can do with these mechanisms (and what they can't)

How do I write a Java app to display soup recipes?



Where are we?

1. The embedding mechanism turns tokens into vectors
2. These vectors are placed in embedding space


"java"
"app"
???
java
app

The attention mechanism performs transformations on embedding vectors to encode the model's input state

This paper from Google Translate is arguably what kicked off the current wave of AI research

Attention is All You Need

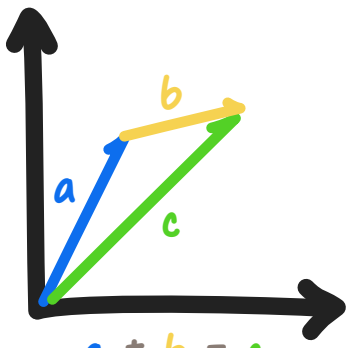
(Google, 2017)

Attention Mechanisms

(3blue1brown)

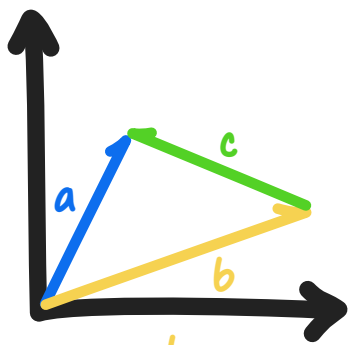
VECTOR OPERATIONS

ADDITION



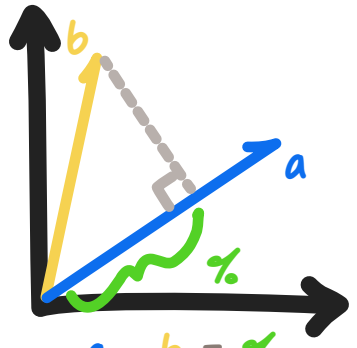
a + b = c

SUBTRACTION



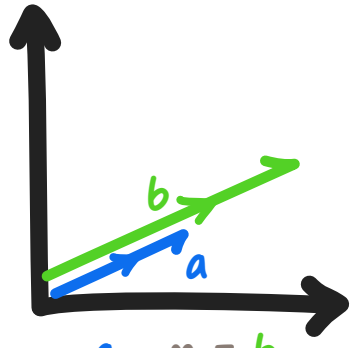
a - b = c

DOT PRODUCT



a · b = %

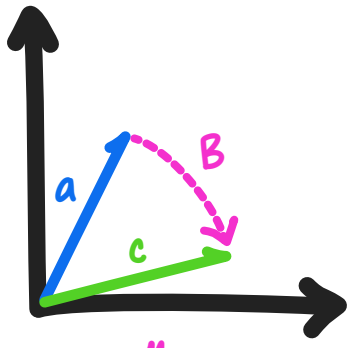
SCALARS



a · m = b

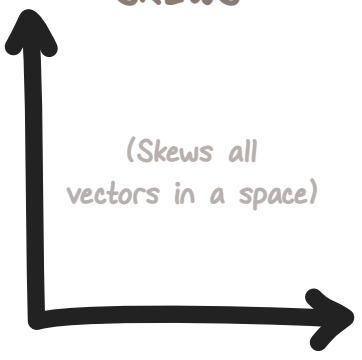
MATRIX TRANSFORMATIONS

ROTATIONS



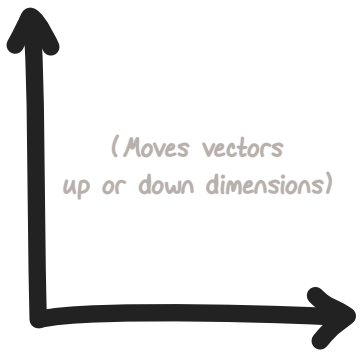
aM = c

SKEWS



(Skews all vectors in a space)

DIMENSION SHIFTS



(Moves vectors up or down dimensions)

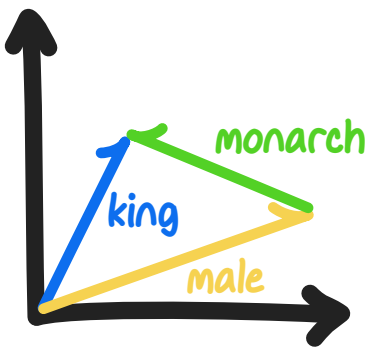
Vectors & Matrices

(3blue1brown)

Example: Subtraction & Addition

Allows us to find differences and composites of embeddings

SUBTRACTION

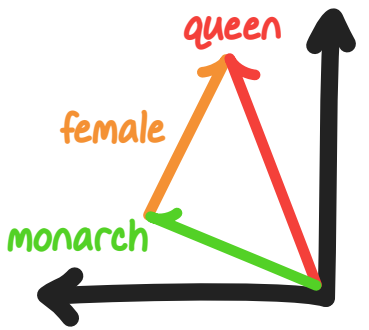


king - male = monarch

1

Subtracting "male" from "king" may result in the general term "monarch"

ADDITION



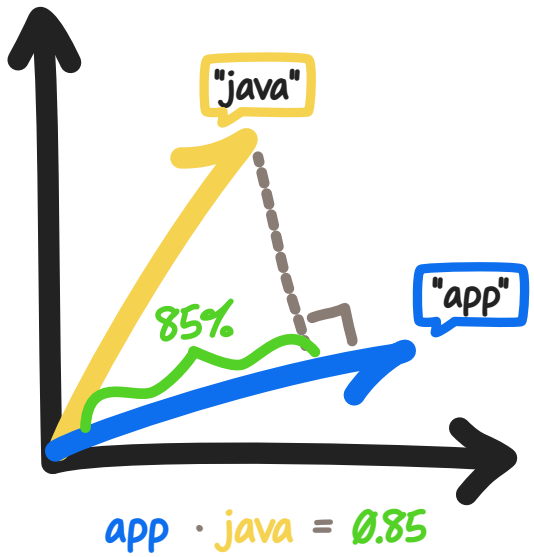
monarch + female = queen

2

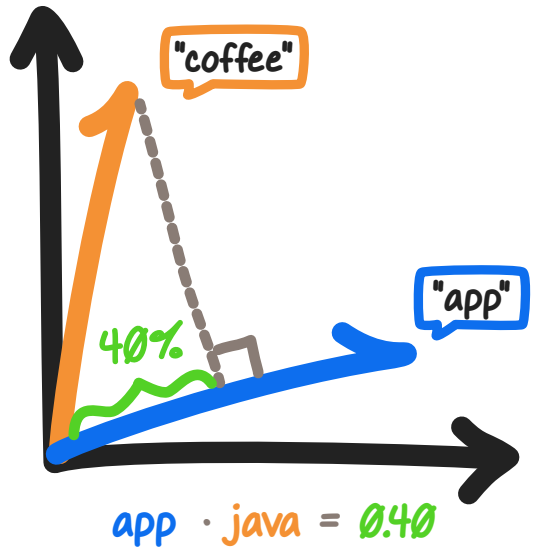
Adding the term "female" to the general term "monarch" may result in "queen"

Example: Dot Products

Computes how well one embedding maps, or relates, to another

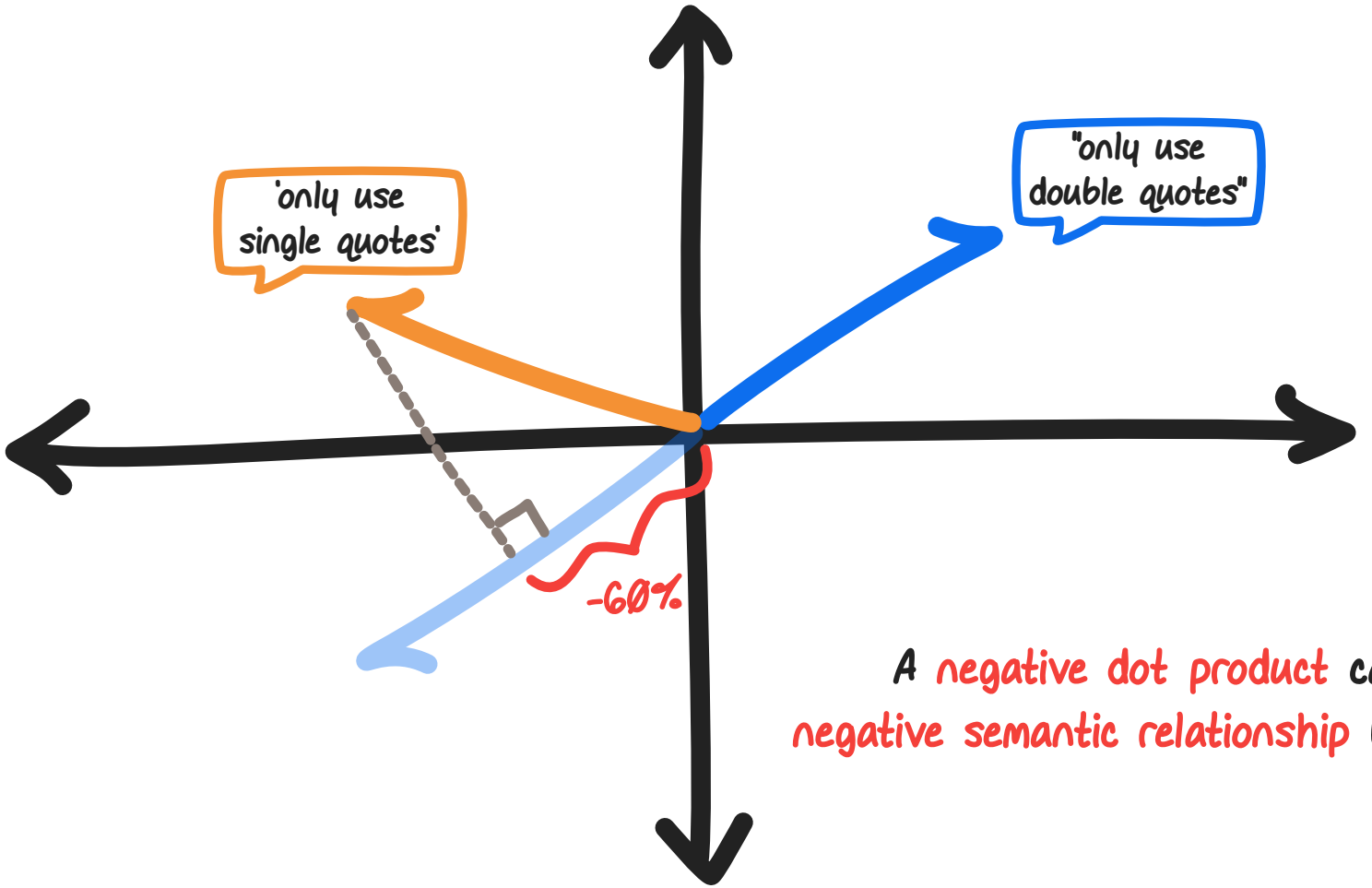


The dot product between two embeddings with a clear semantic connection will be positive



If there is not a clear semantic connection, the dot product will be closer to 0

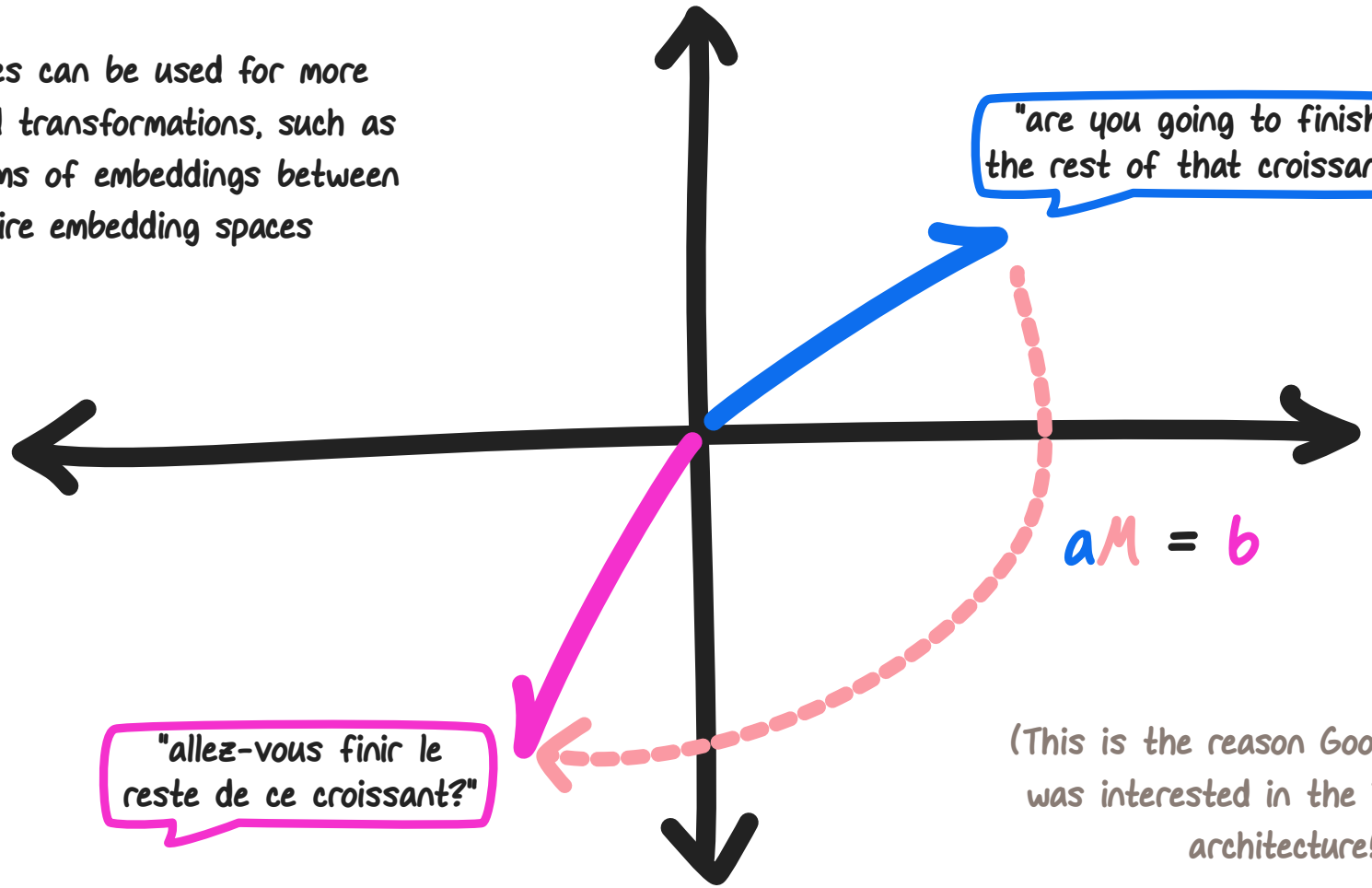
Example: Negative Dot Products



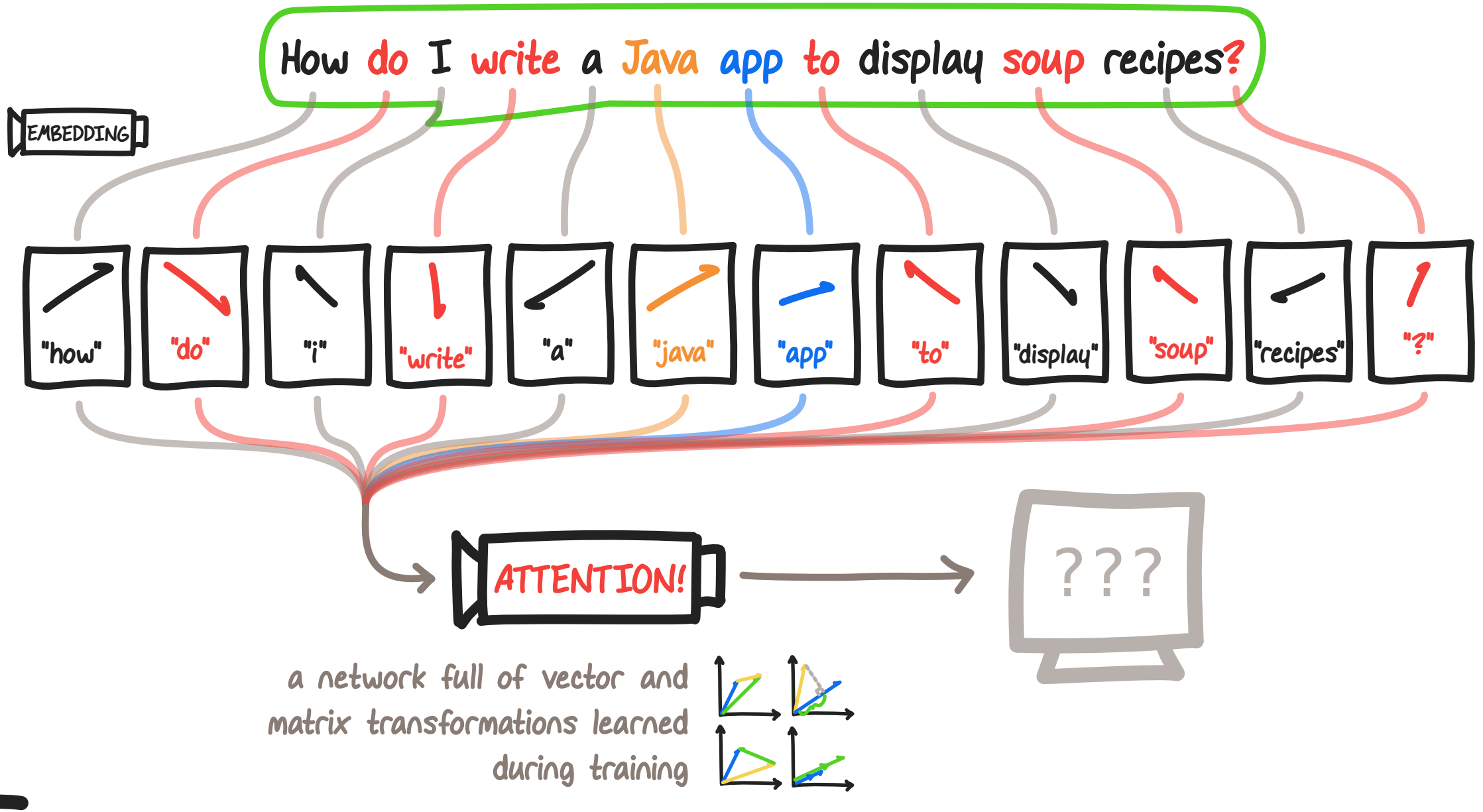
A negative dot product can represent a negative semantic relationship between embeddings

Example: Rotations

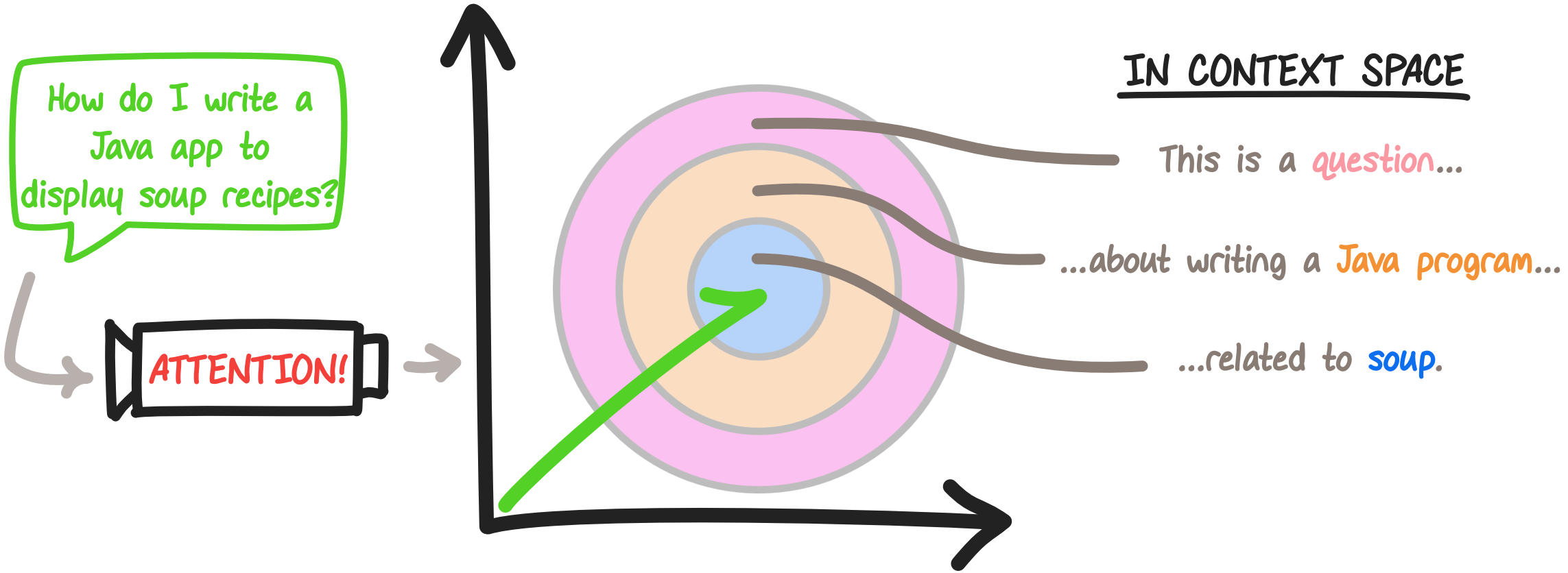
Matrices can be used for more advanced transformations, such as transforms of embeddings between entire embedding spaces



(This is the reason Google Translate was interested in the Transformer architecture!)

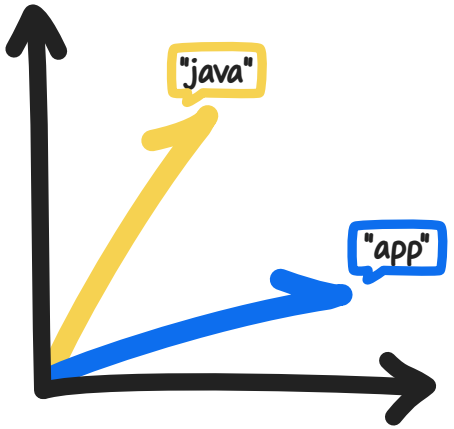

How do I write a Java app to display soup recipes?
EMBEDDING
"how"
"do"
"i"
"write"
"a"
"java"
"app"
"to"
"display"
"soup"
"recipes"
"?"
ATTENTION!
???
a network full of vector and
matrix transformations learned
during training

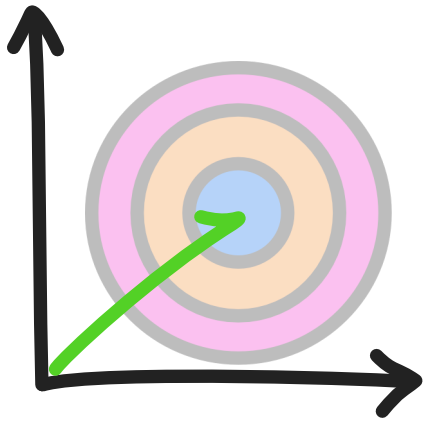
CONTEXT SPACE

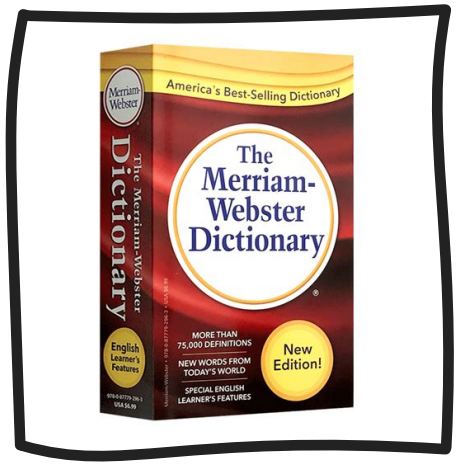


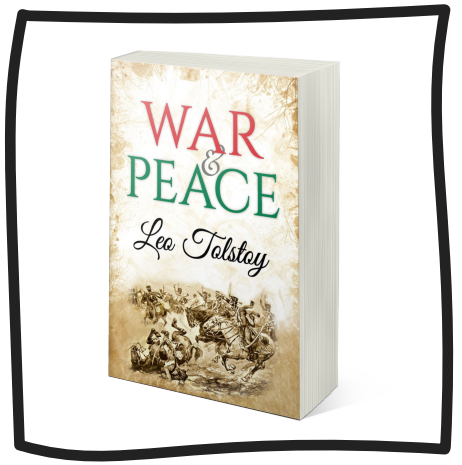
Context Space captures the connections between the input embeddings, allowing for a richer semantic representation of the input data.

Embedding Space is to Context Space...





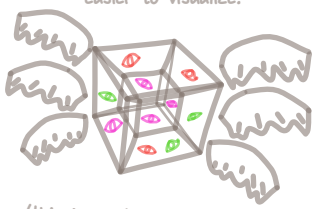




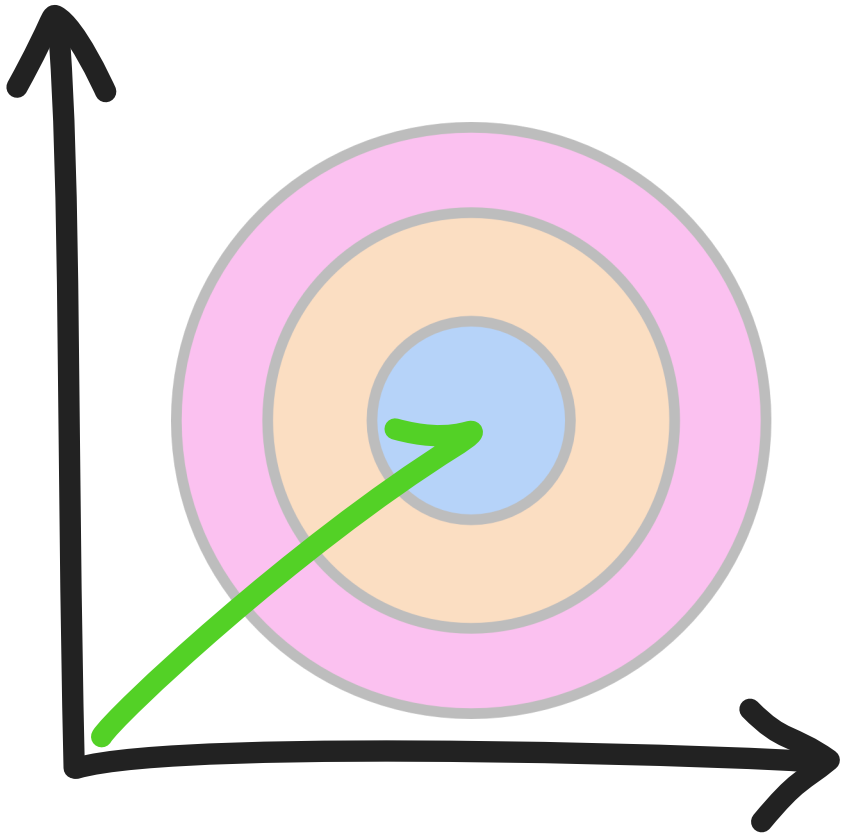
...what a Dictionary is to a Novel

the concrete, technical difference is that context space is represented using tensors - high-dimensional matrices - instead of vectors.

we're going to be continuing to use vectors because they're easier to visualize.



(this is my best 2D depiction of a tensor.)



For the rest of the presentation,
we will primarily be using context spaces.

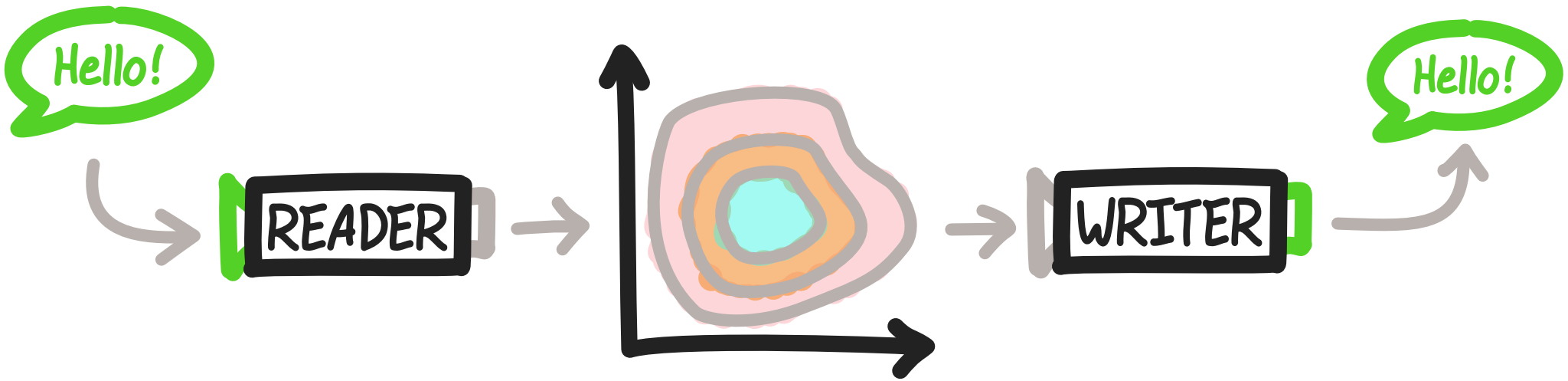
Do you understand:

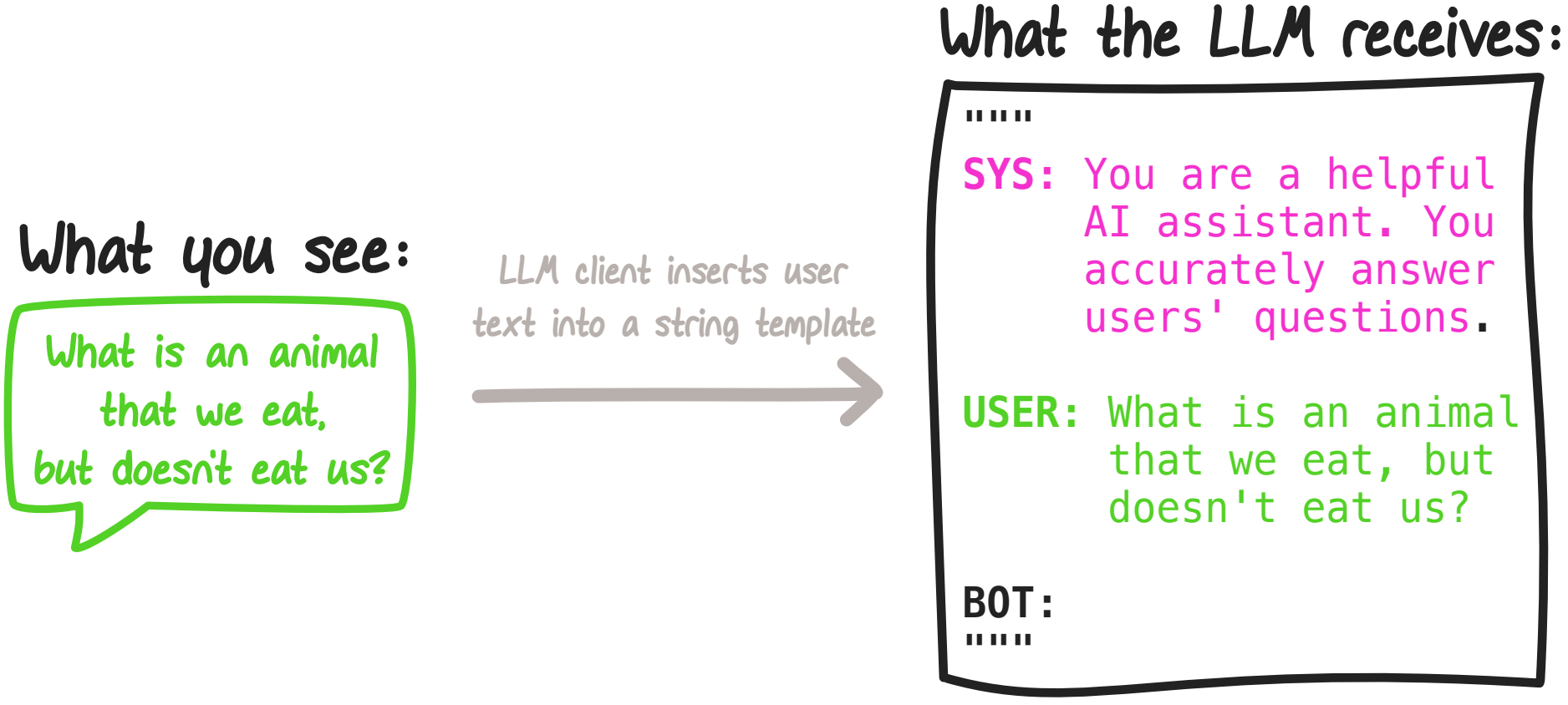
- Vector operations on natural language tokens?
- The difference between embedding and context space?

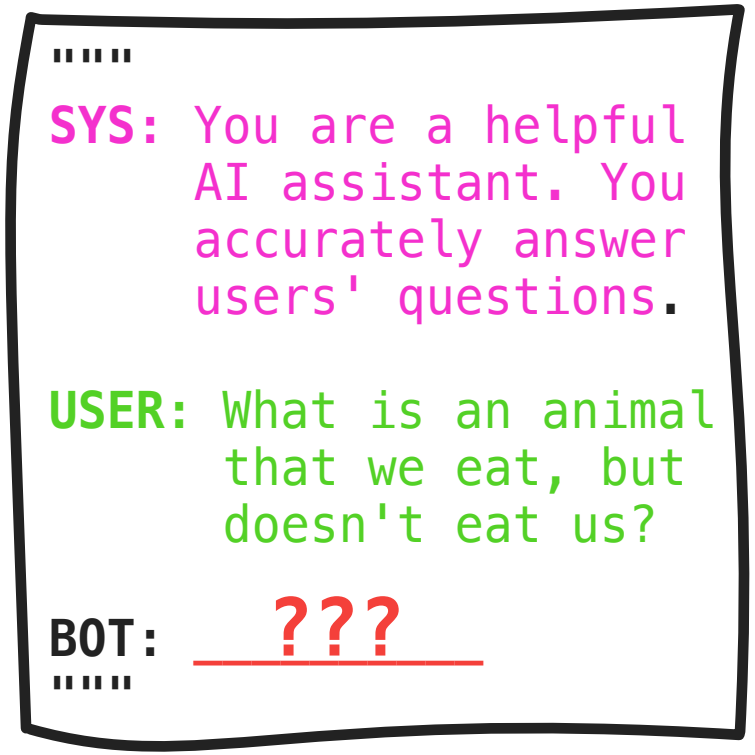
Up Next!

WRITER

Remember me?


Hello!
READER
WRITER
Hello!


What you see:
What is an animal that we eat, but doesn't eat us?
LLM client inserts user text into a string template
What the LLM receives:
"""
SYS: You are a helpful AI assistant. You accurately answer users' questions.
USER: What is an animal that we eat, but doesn't eat us?
BOT:
"""



The WRITER's only goal:

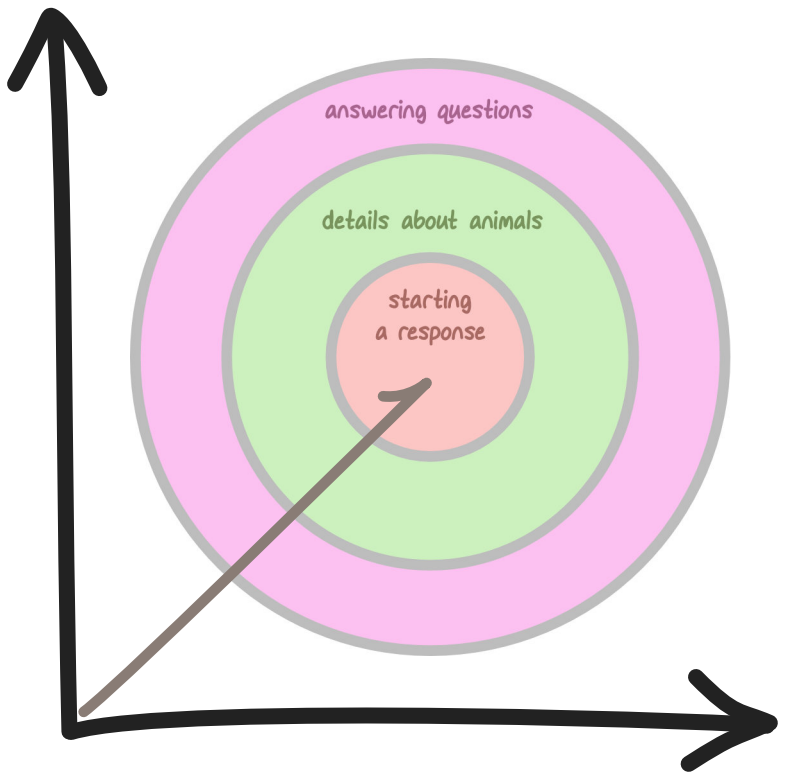
Fill in the blank

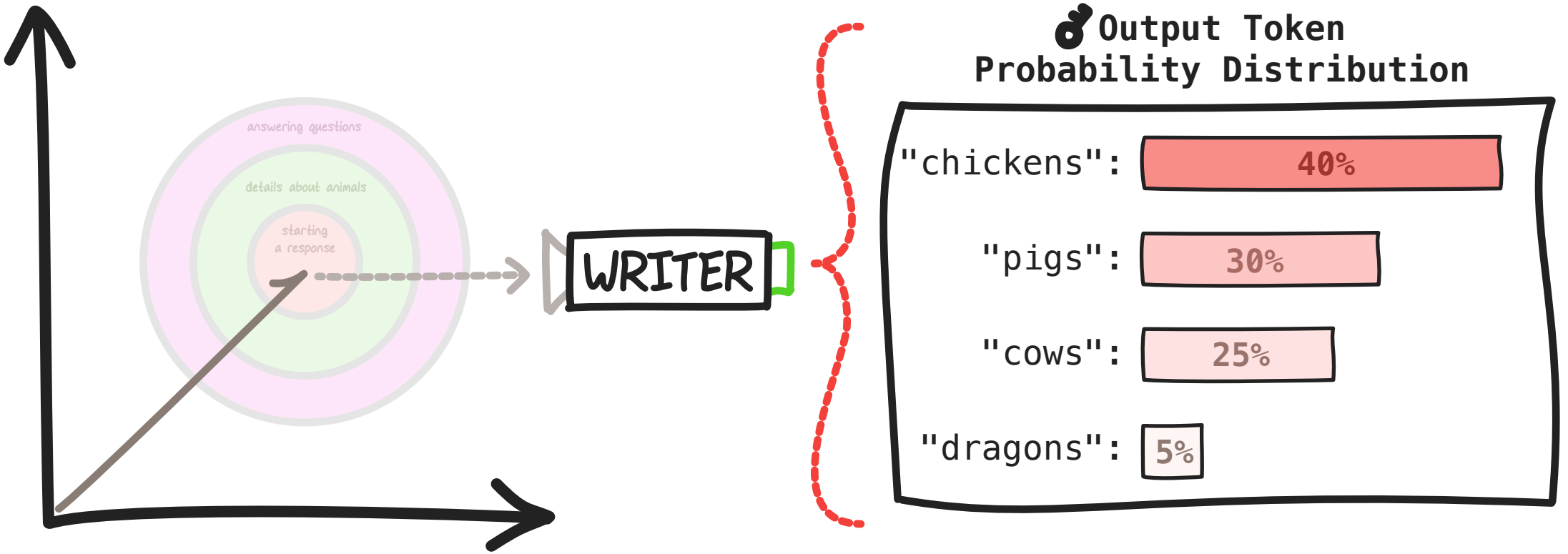
What the LLM receives:

```
"""
SYS: You are a helpful
     AI assistant. You
     accurately answer
     users' questions.

USER: What is an animal
      that we eat, but
      doesn't eat us?

BOT:  ???
"""
```

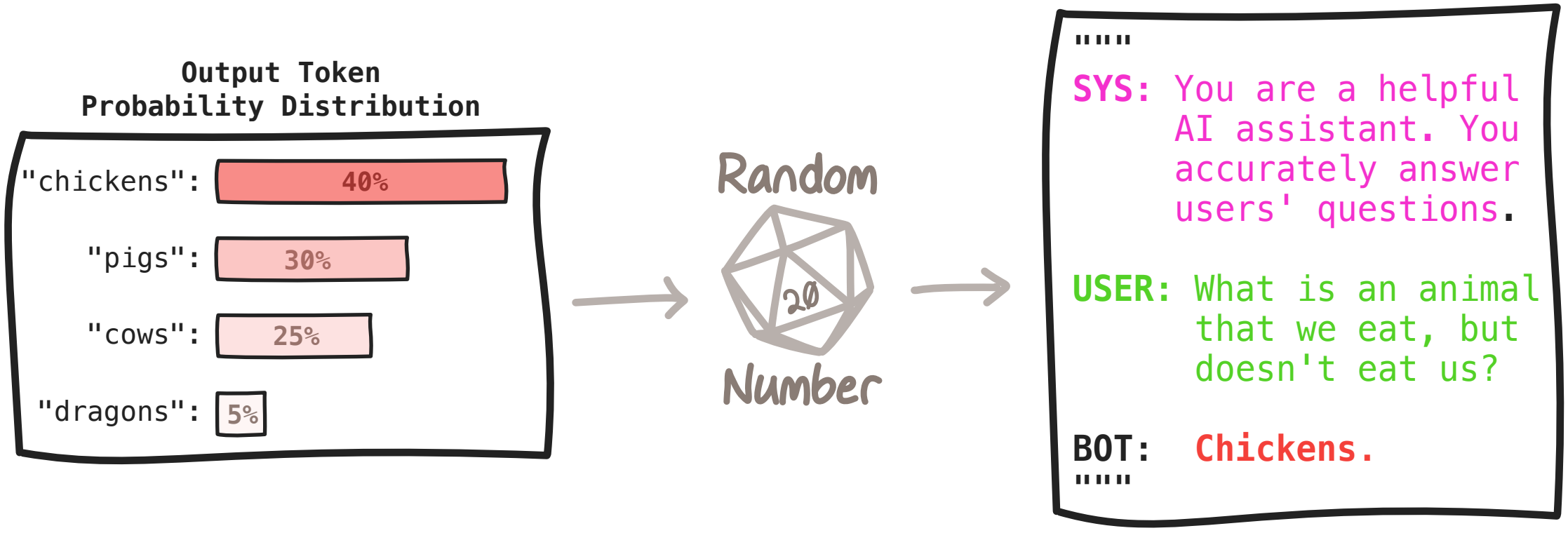
READER

Translation to Context Space:

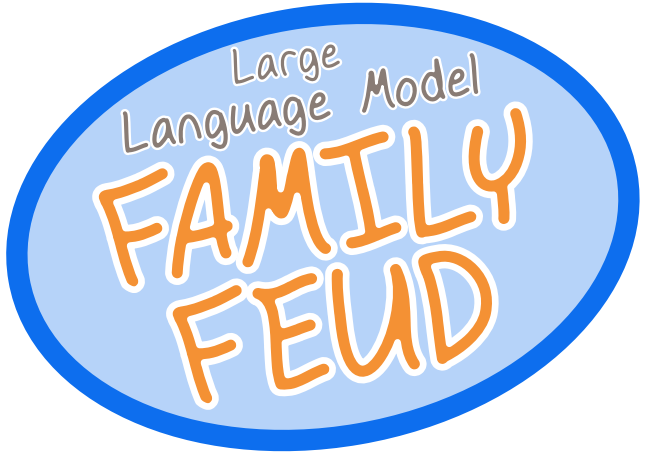


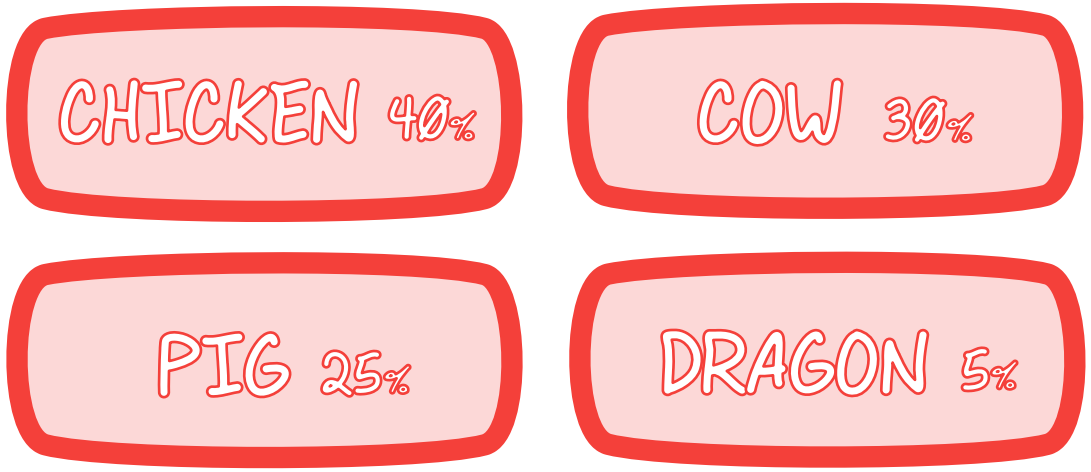


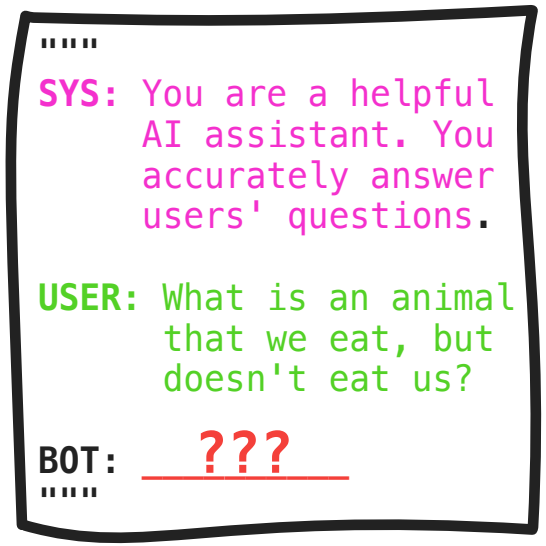
The prompt's position in Context Space maps to a probability distribution of output tokens

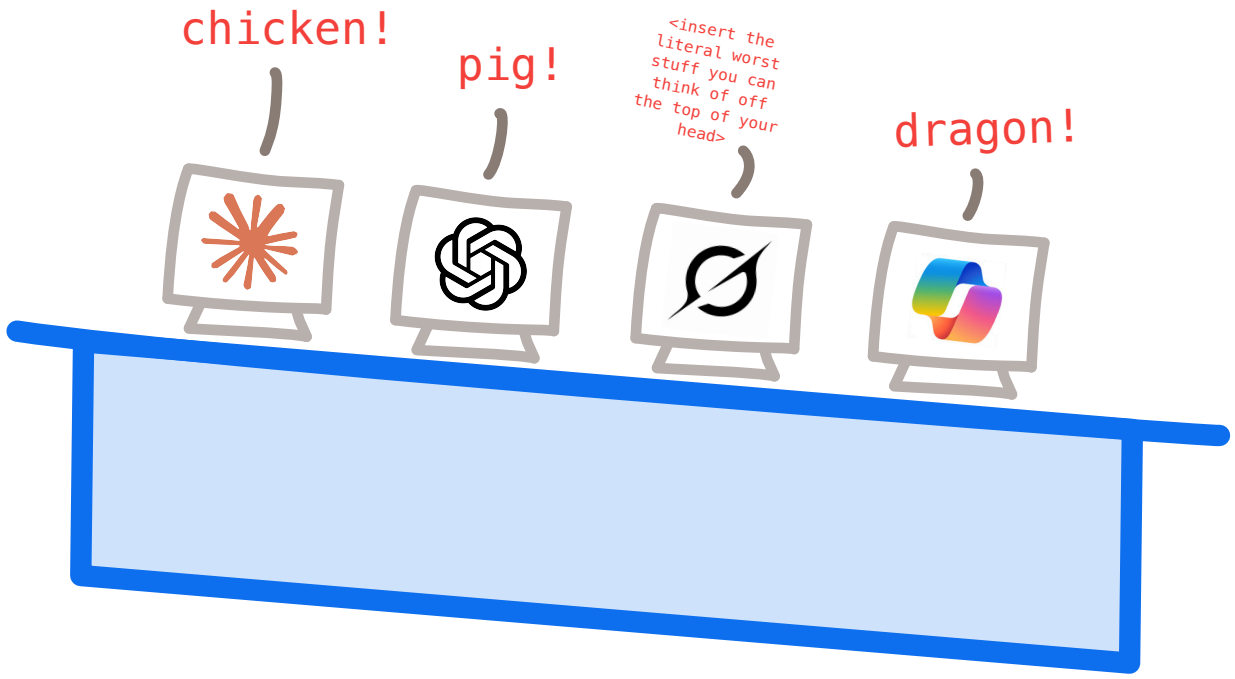


The Writer selects a random token, based on the distribution, and appends it to the end of the string.


Large
Language Model
FAMILY
FEUD

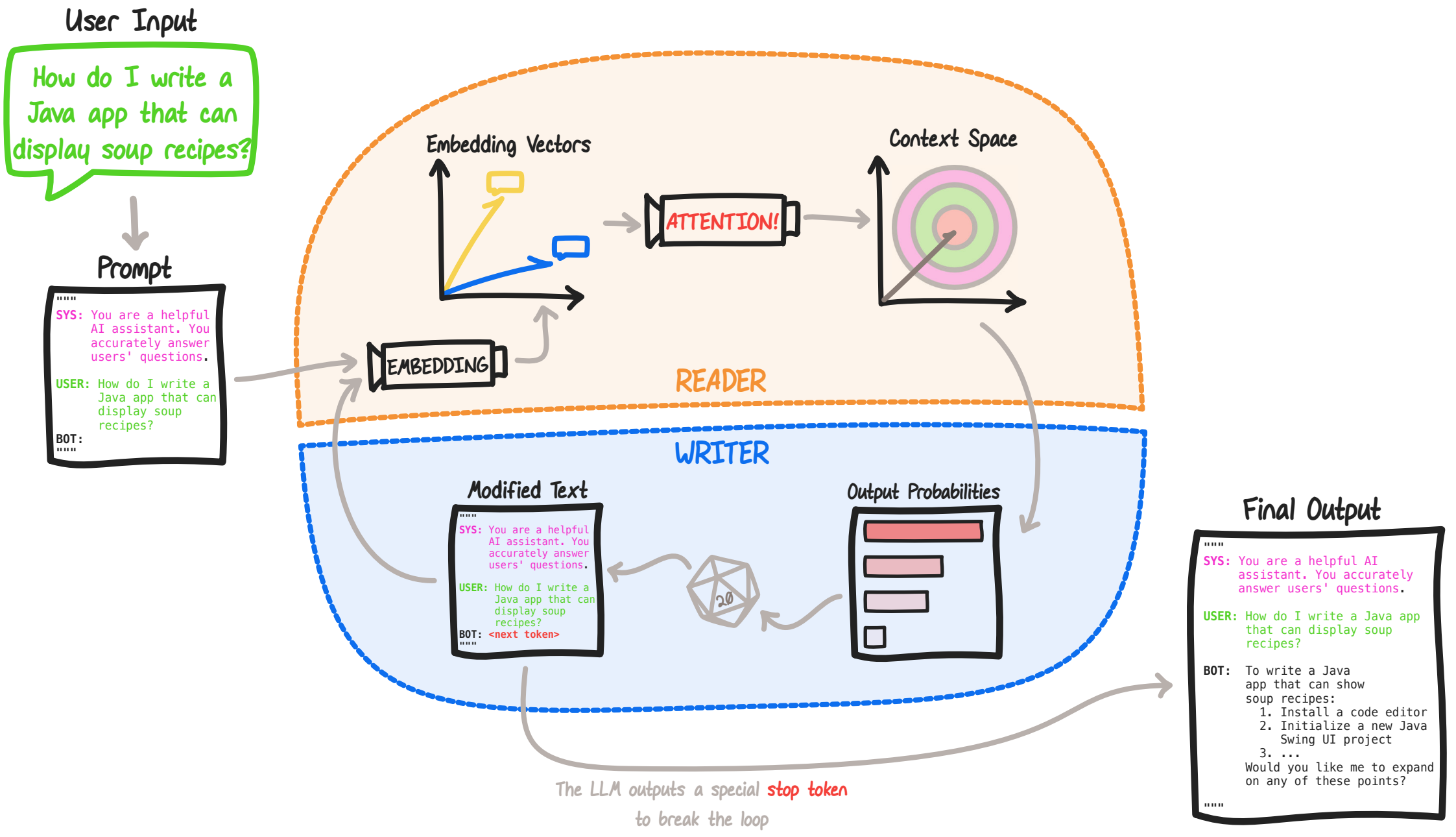

CHICKEN 40%
COW 30%
PIG 25%
DRAGON 5%

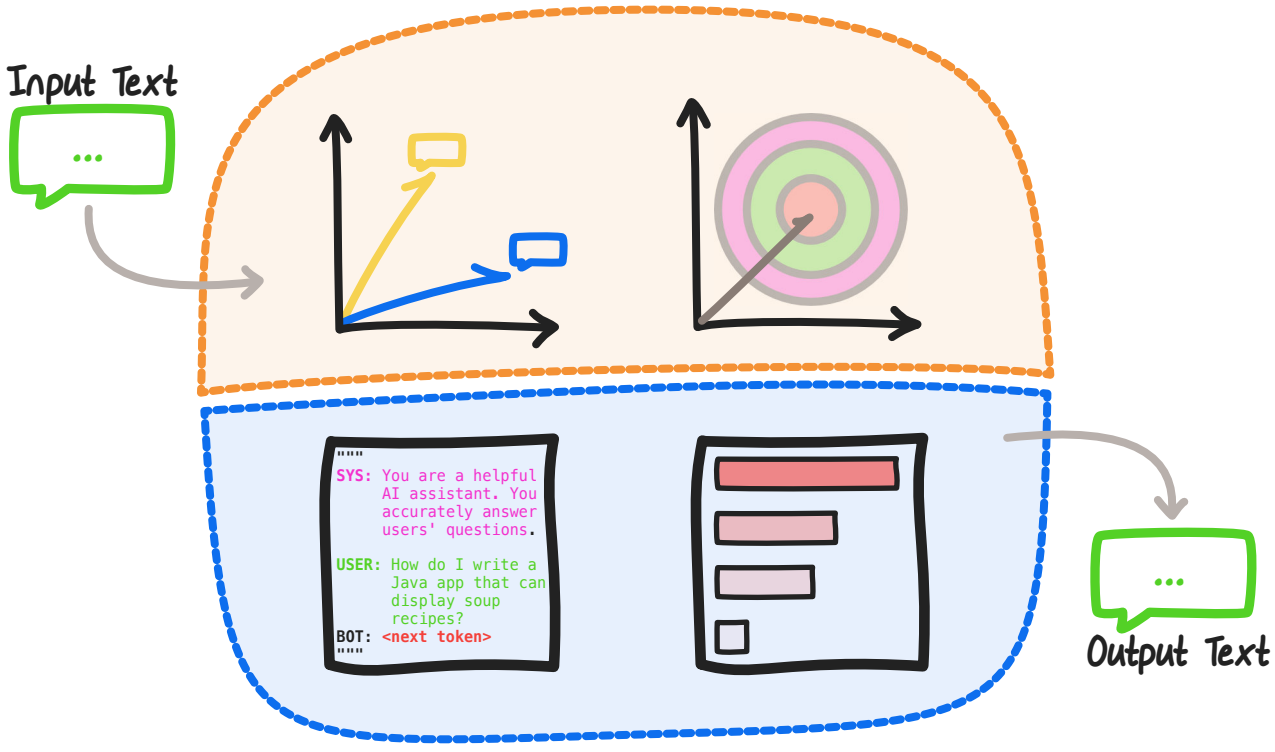

"""
SYS: You are a helpful AI assistant. You accurately answer users' questions.
USER: What is an animal that we eat, but doesn't eat us?
BOT: ???
"""


chicken!
pig!
<insert the literal worst stuff you can think of off the top of your head>
dragon!

Let's bring this all together, now.

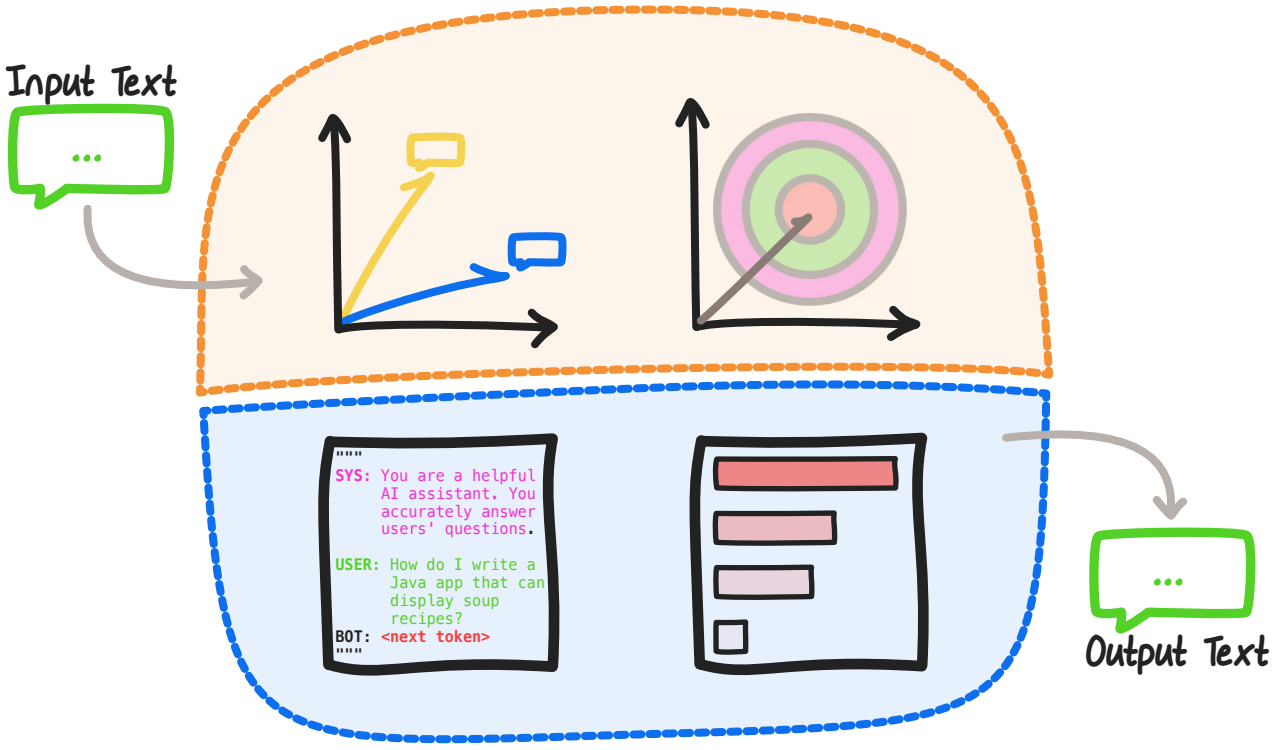
(it's flowchart time, folks!)


User Input
How do I write a Java app that can display soup recipes?
Prompt
"""
SYS: You are a helpful AI assistant. You accurately answer users' questions.
USER: How do I write a Java app that can display soup recipes?
BOT:
"""
Embedding Vectors
EMBEDDING
ATTENTION!
Context Space
READER
WRITER
Modified Text
"""
SYS: You are a helpful AI assistant. You accurately answer users' questions.
USER: How do I write a Java app that can display soup recipes?
BOT: <next token>
"""
Output Probabilities
Final Output
"""
SYS: You are a helpful AI assistant. You accurately answer users' questions.
USER: How do I write a Java app that can display soup recipes?
BOT: To write a Java app that can show soup recipes:
1. Install a code editor
2. Initialize a new Java Swing UI project
3. ...
Would you like me to expand on any of these points?
"""
The LLM outputs a special stop token to break the loop



- This loop is the only thing LLMs can do.

- It is the only thing LLMs will ever do.

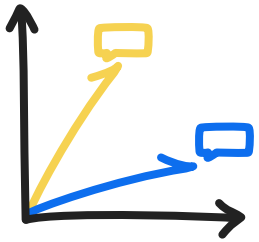


simple ≠ useless

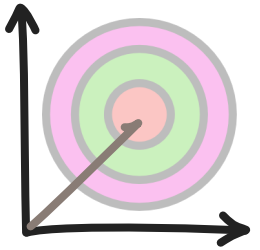
KEY TAKEAWAYS

about how LLMs work

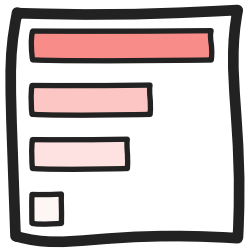
Embeddings



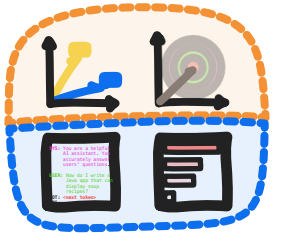
Attention & Context



Output Probabilities



Generative Loop



You need to understand how LLMs work before diving into what they're really capable of

What

are LLMs actually capable of?

KEY TAKEAWAYS

- Recognize shortcomings of the LLM architecture itself
- Explore solutions to these shortcomings and how to use them

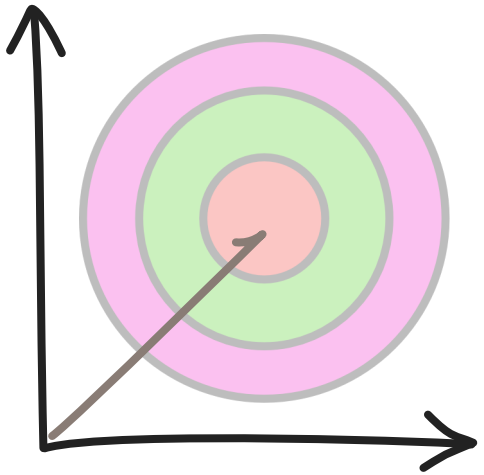
to better understand

why

they affect our industry

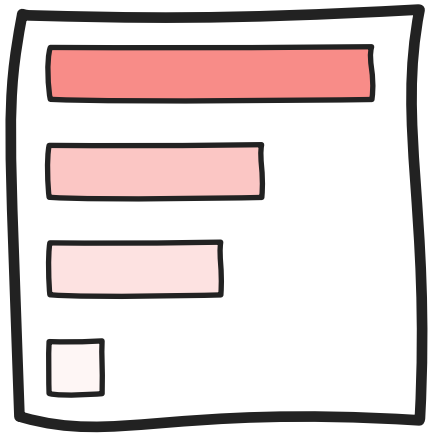
Strengths
of the LLM Architecture

Rich Context



The attention mechanism and context space is an elegant way to explore concepts and trivializes bulk text summarization

Probability Quality



Provided with a specific and intentional position in context space, these output probability distributions can be incredibly accurate

Loop Scale

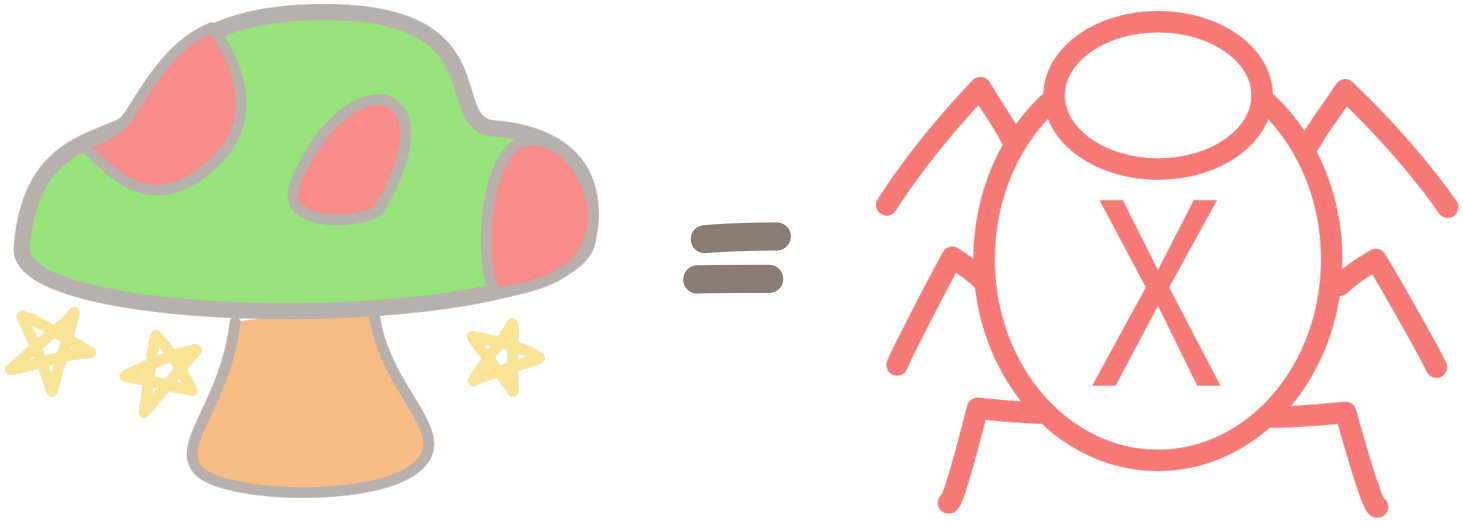
The ease of scaling the generative loop trivializes bulk text generation

On The Topic of Hallucinations

The term "hallucination" implies erroneous, unexplained behavior due to an error or flaw in a cognitive process.

LLMs don't think.

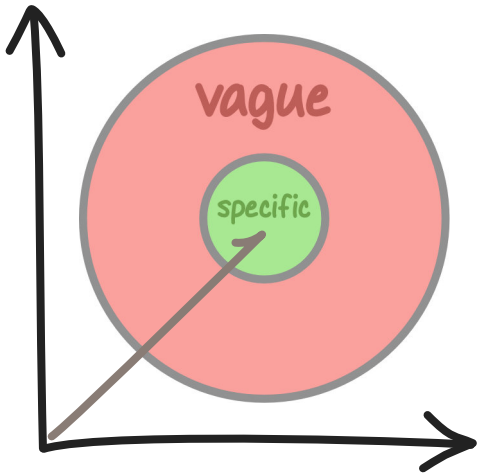
They are machines that take inputs and return outputs.



Hallucinations are bugs

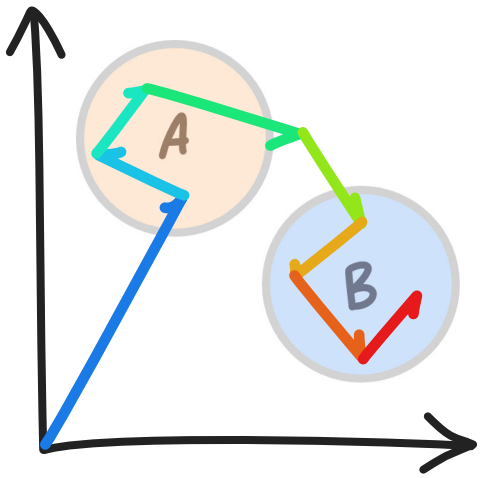
that can be resolved and worked around

Causes of Hallucinations



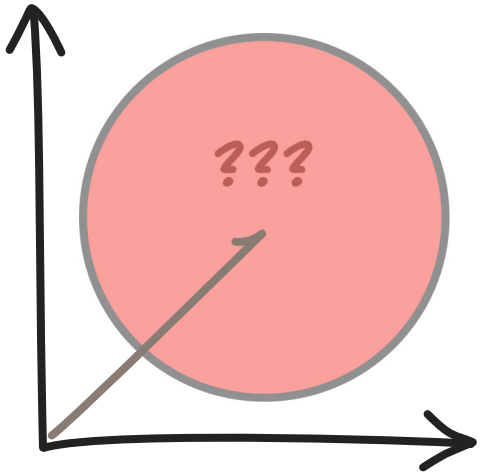
Specificity

A prompt is insufficiently specific and lacks particular details about desired outputs



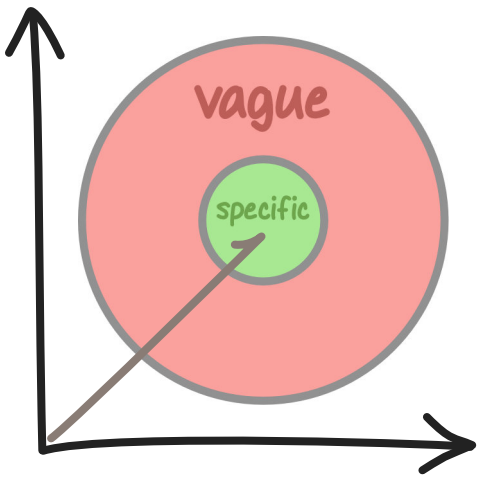
Drift

Generated tokens drift the context out of the desired region in context space

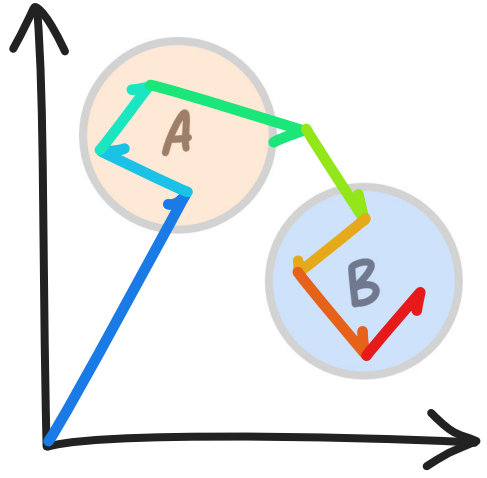


Void Context

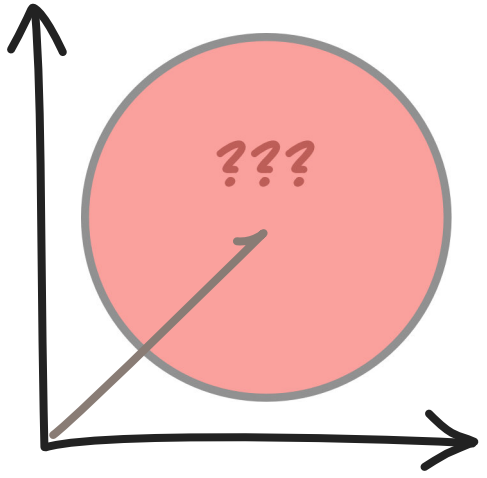
The prompt depends on implicit information not already present in the learned context space



Specificity



Drift



Inferred Context

CAUSE

A prompt is insufficiently specific and lacks particular details about desired outputs

Generated tokens drift the context out of the desired region in context space

The prompt depends on implicit information not already present in the learned context space

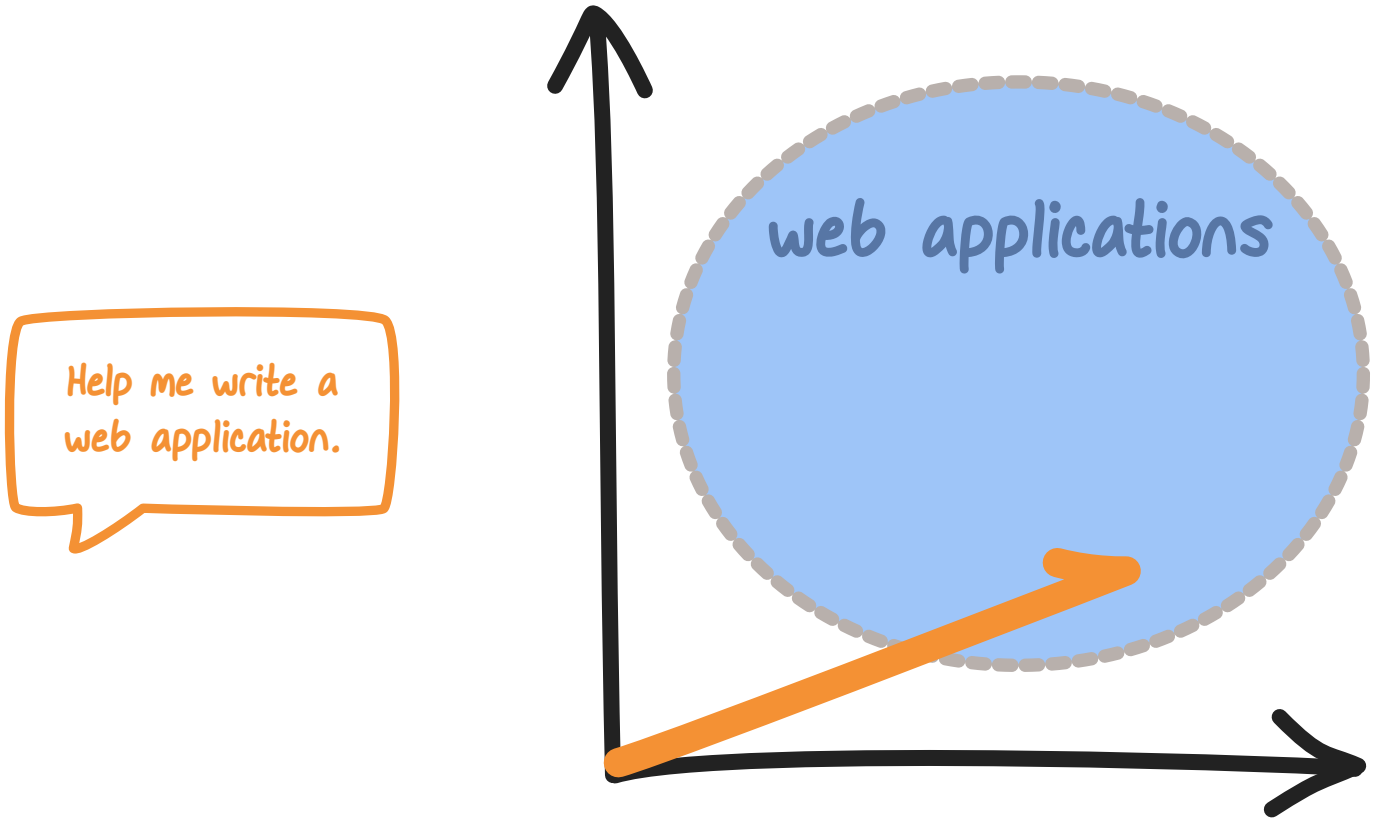
EFFECT

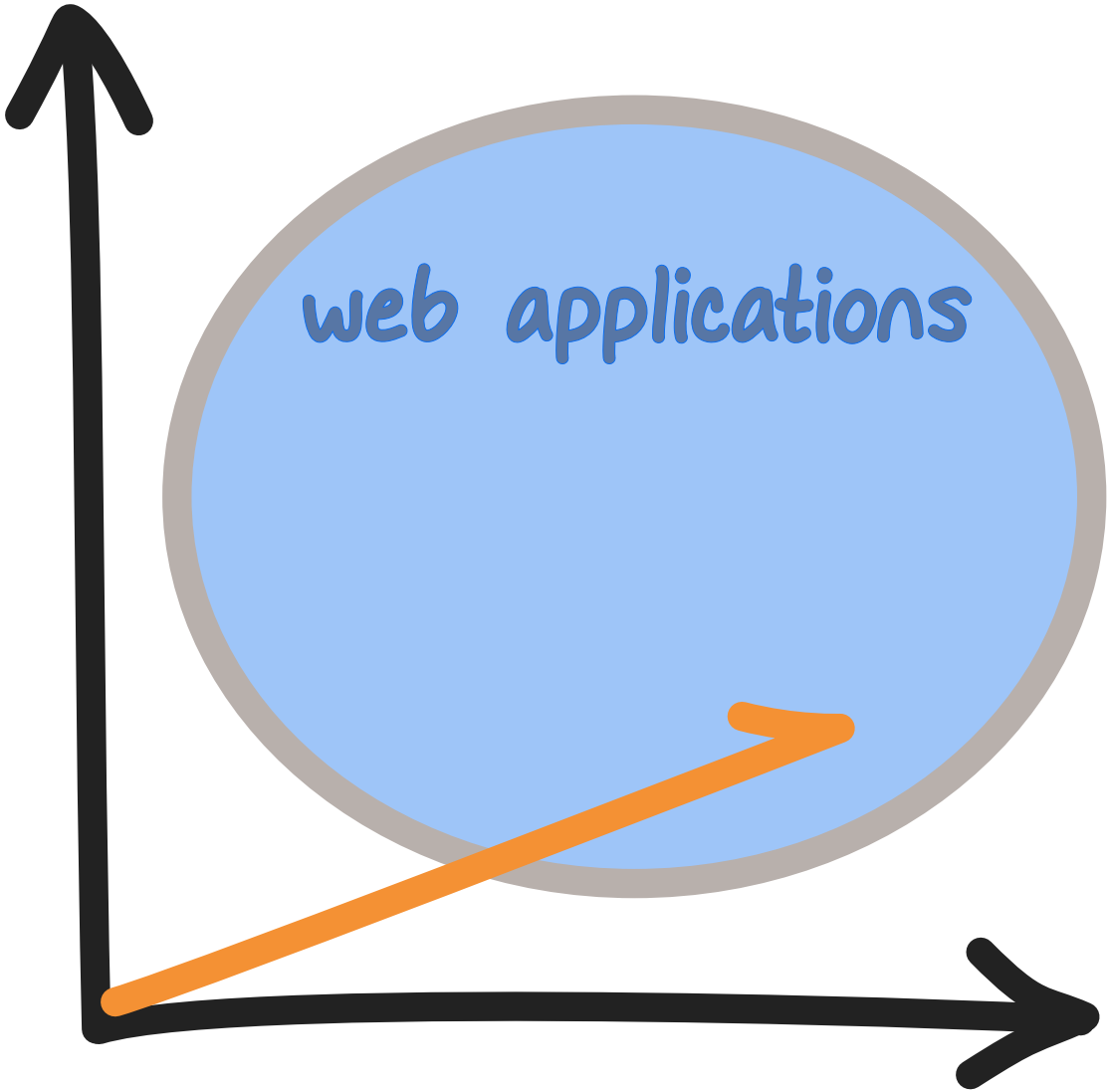
LLM outputs will be vaguely correct but not in the desired format or with the desired level of detail

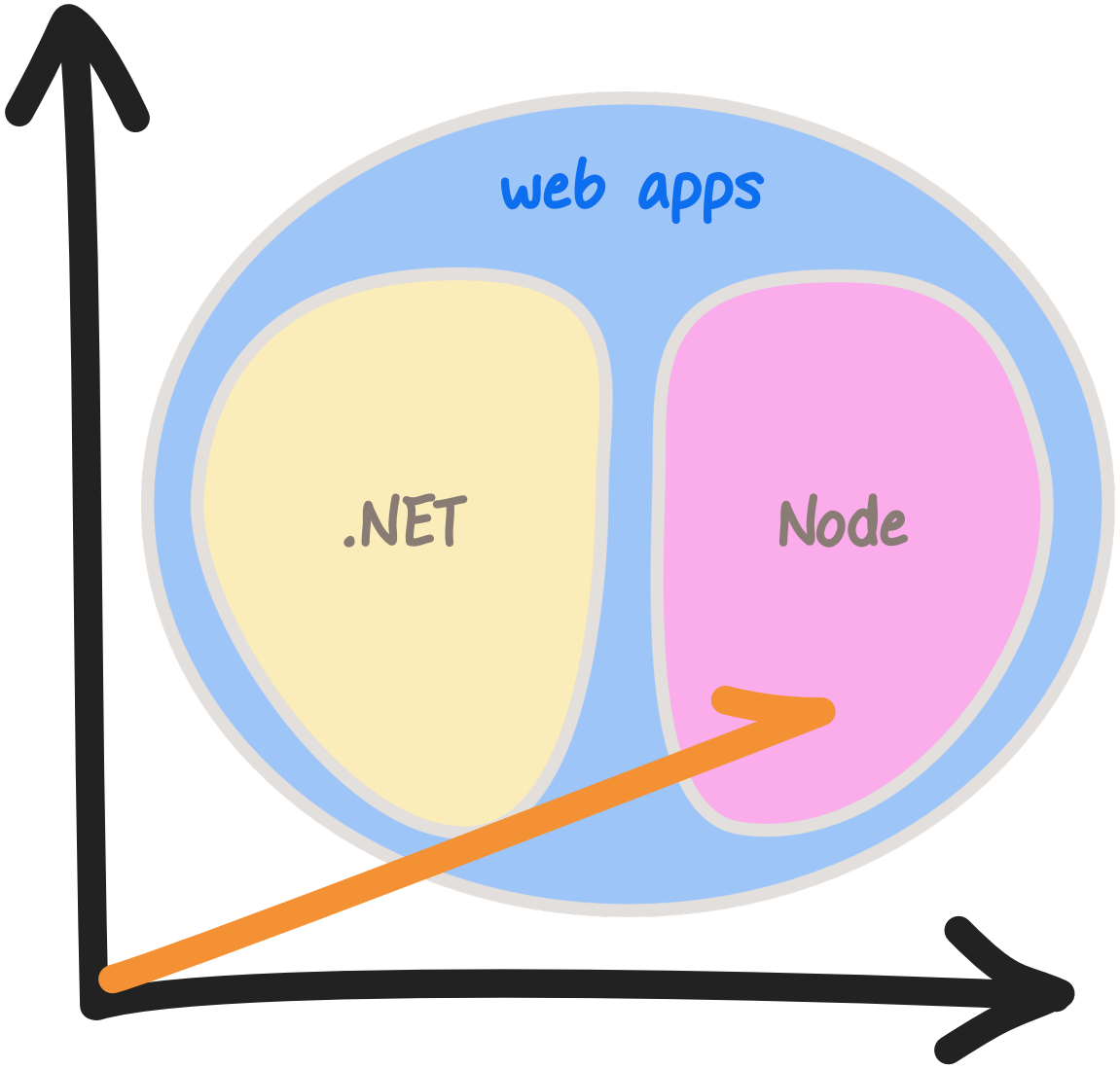
LLM outputs will drift away from the desired output as more tokens are generated

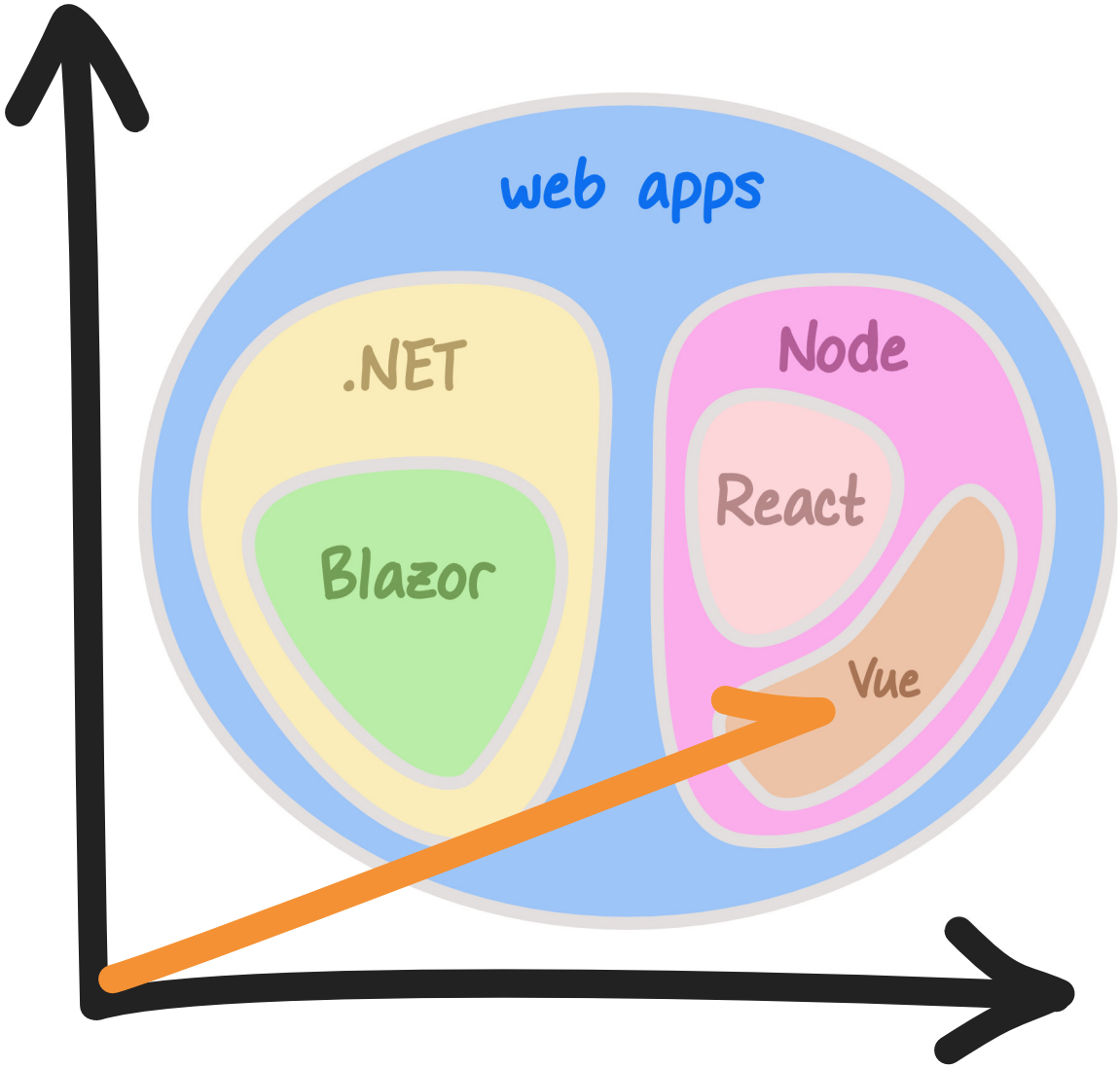
LLM outputs will be insufficient or incorrect, pulling from unintended composite spaces

Example: Specificity Hallucinations




web applications


web apps
.NET
Node


web apps
.NET
Blazor
Node
React
Vue

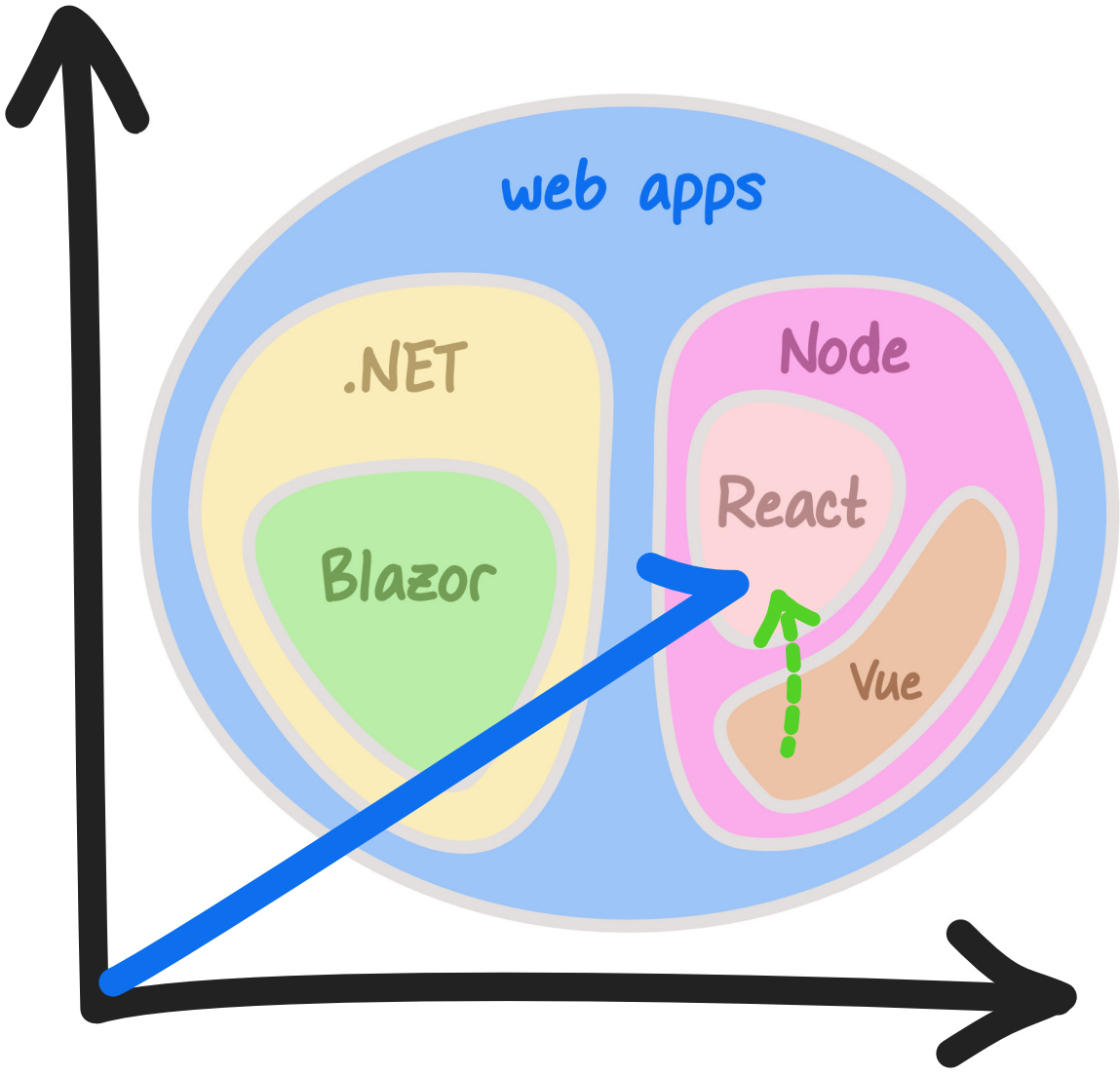
Help me use NodeJS and React to write a web application.

ADDS

+ Language
+ Framework

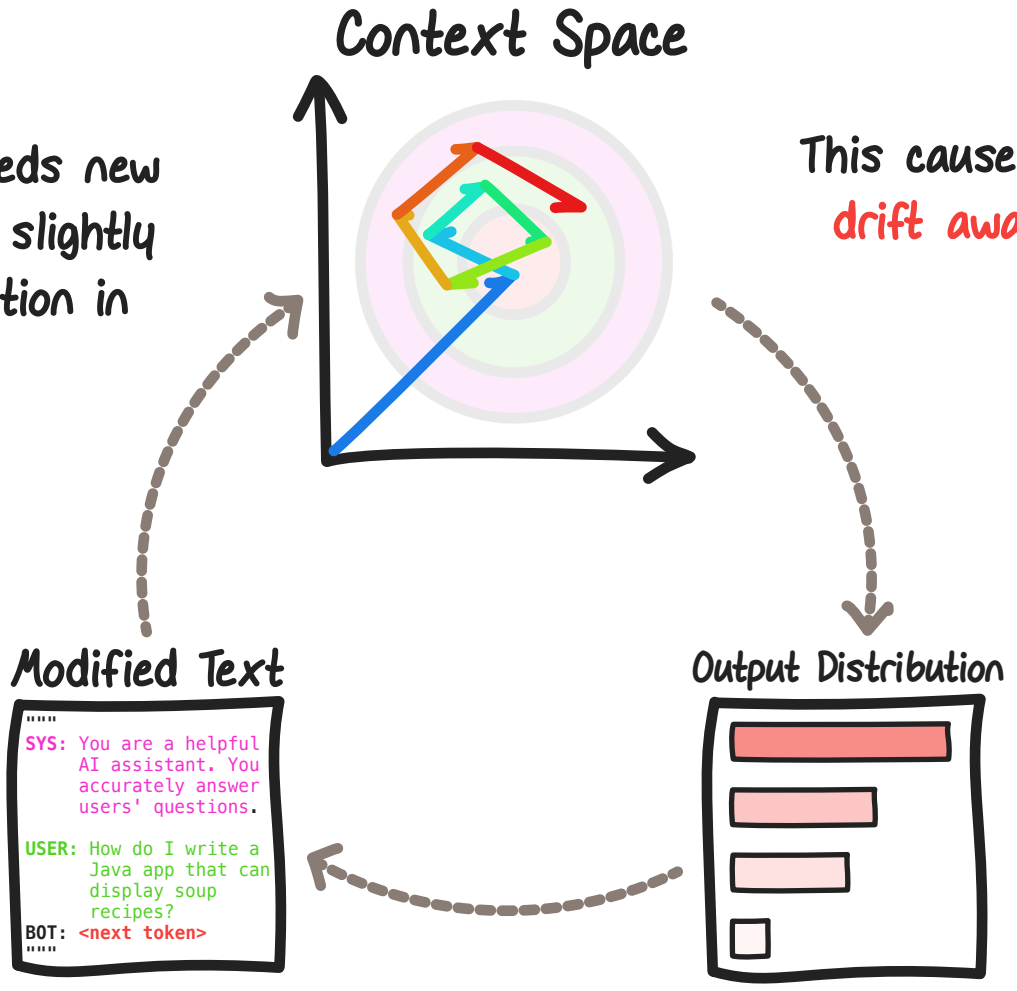
LACKS

- Structure?
- Purpose?
- Style?



Context Drift

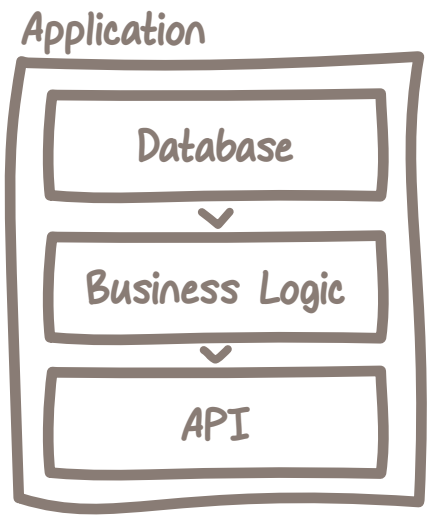
The core generative loop feeds new tokens back into attention, slightly modifying the prompt position in context space

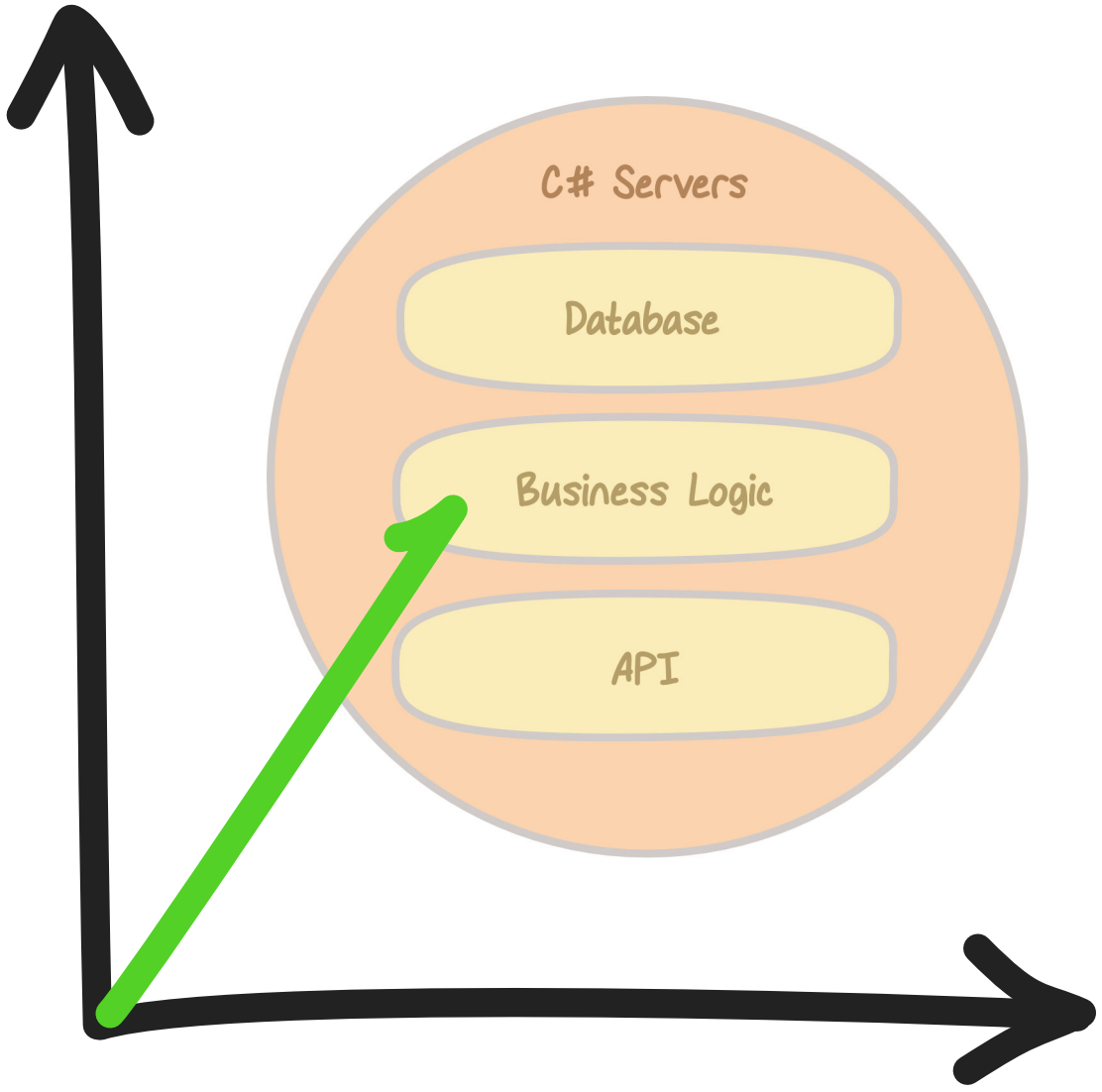


This causes the context to naturally drift away from the initial prompt over time

Example: Drift Hallucinations

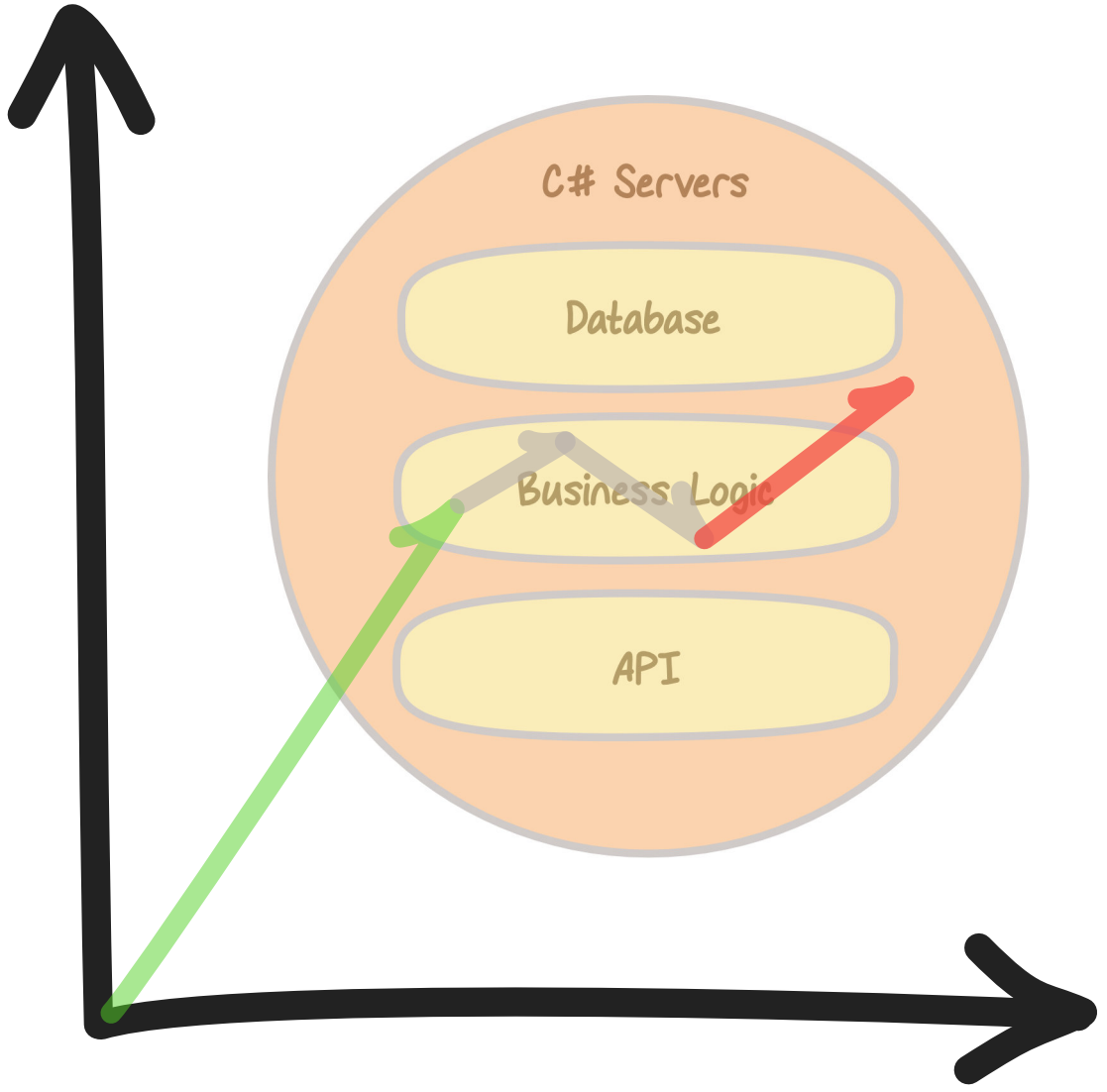
Let's write the C# business logic for <FEATURE>





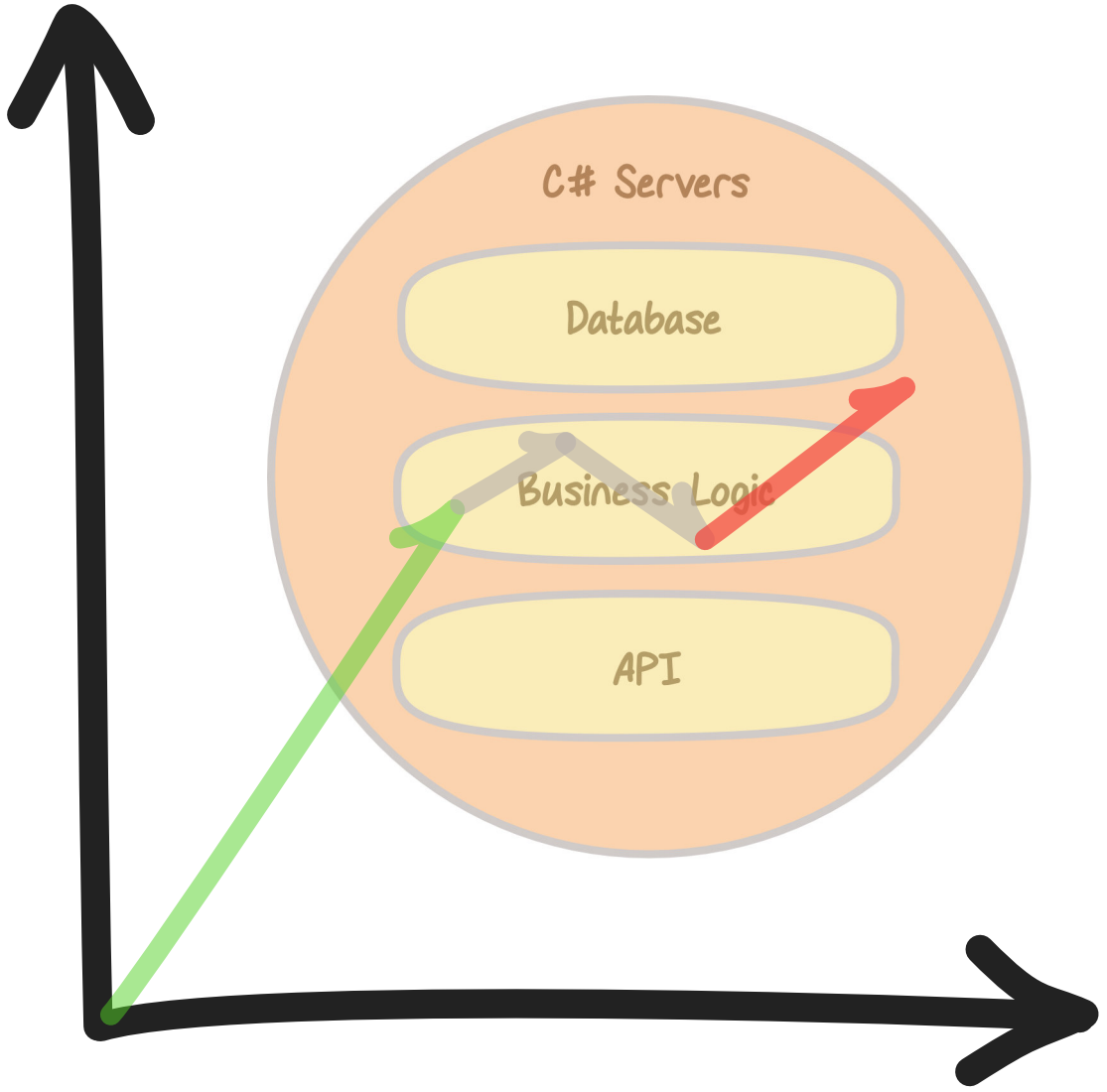
Let's write the C# business logic for <FEATURE>

- Generates boilerplate
- Writes the logic
- Language server shows that the import for data layer requirements fails



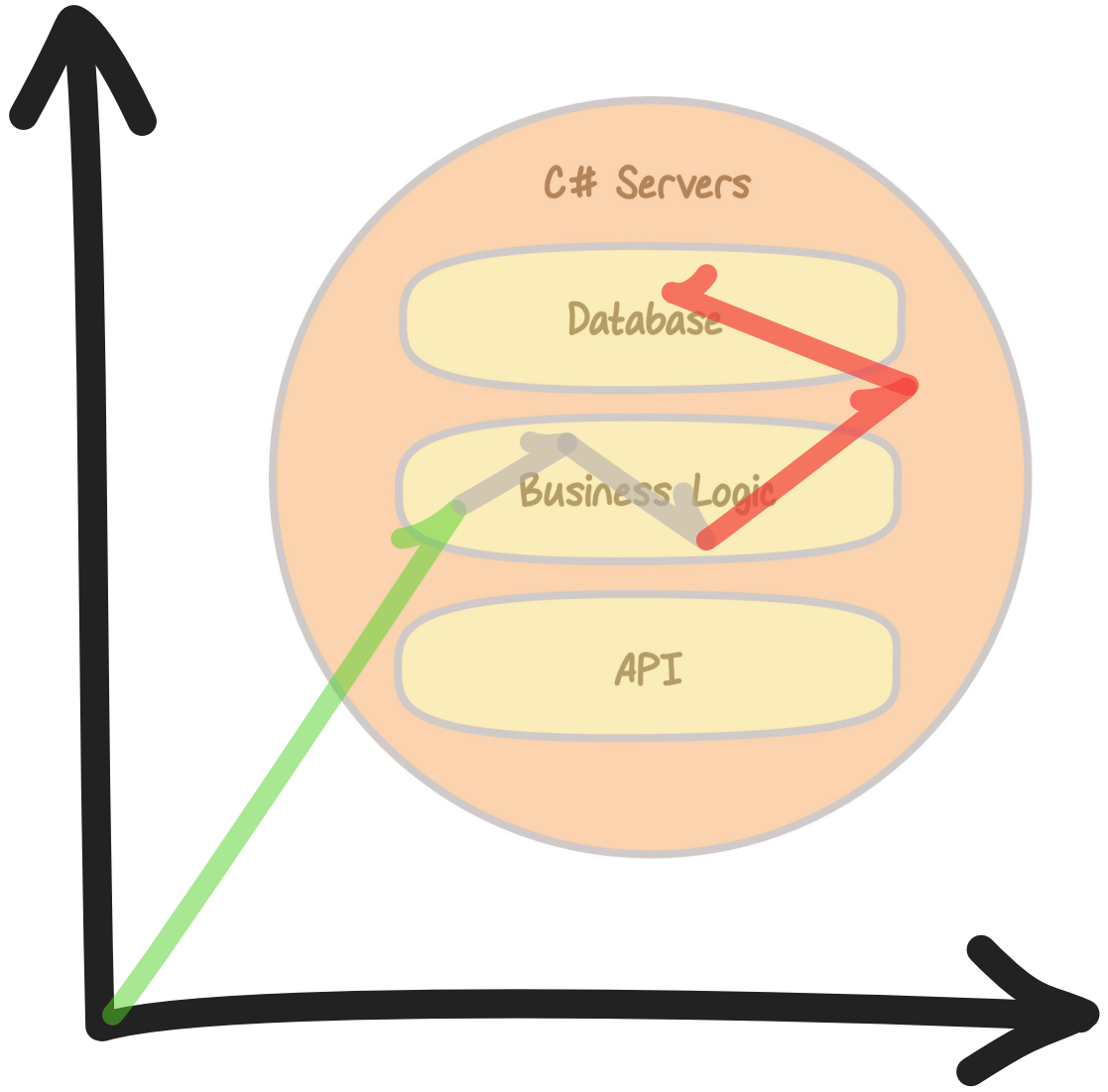
Let's write the C# business logic for <FEATURE>

- Generates boilerplate
- Writes the logic
- Language server shows that the import for data layer requirements fails



Let's write the C# business logic for <FEATURE>

- Generates boilerplate
- Writes the logic
- Language server shows that the import for data layer requirements fails
- **Begins writing random data entities**



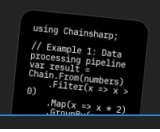
Example: Inferred Context

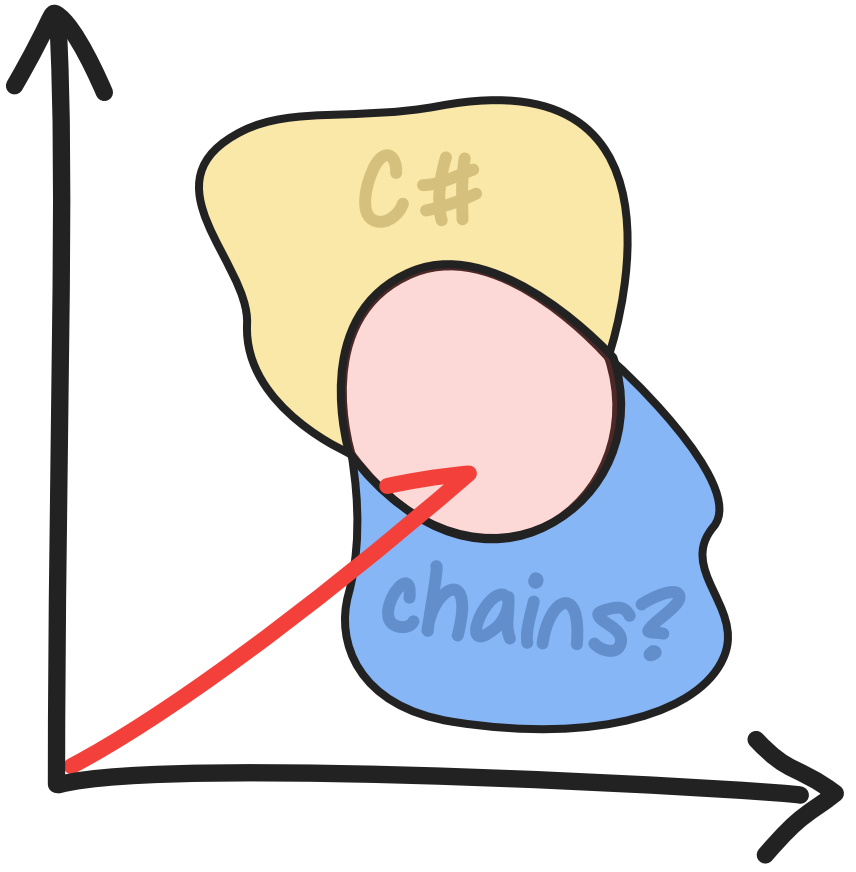
Please write a snippet of Chainsharp workflow code

Based on the name "Chainsharp," I'd infer this is likely a C# library focused on method chaining or fluent APIs. Here's what I imagine Chainsharp-style code might look like:

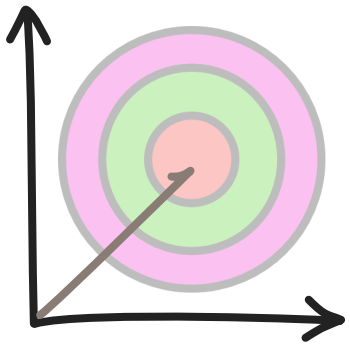
Chainsharp-Style Code Example

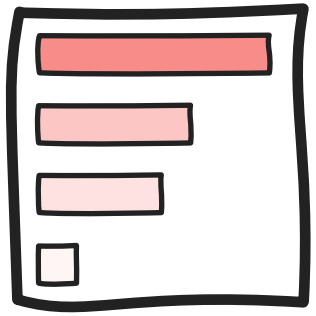
Code



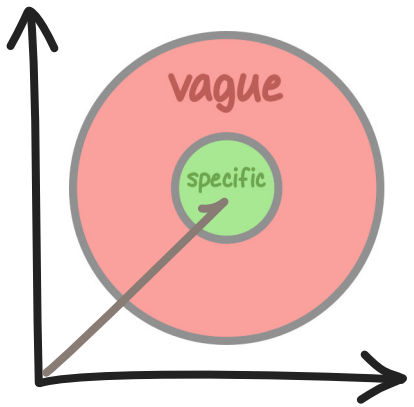


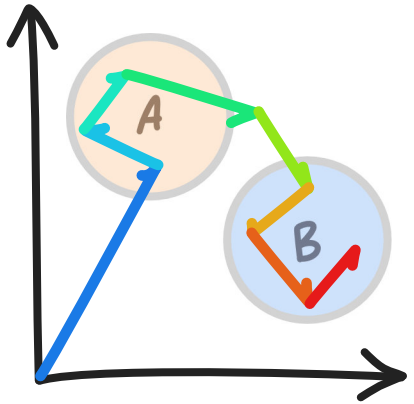
How can we use the strengths of the architecture

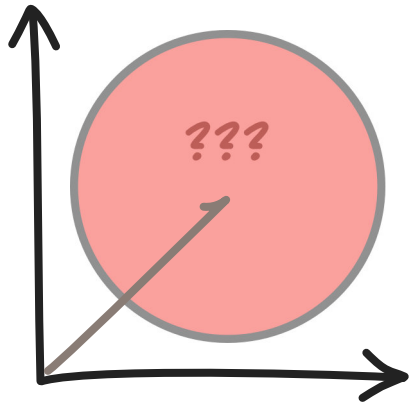




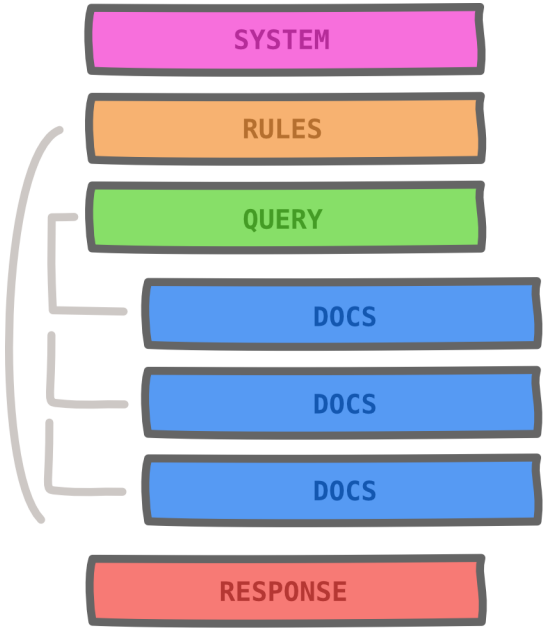
to address its weaknesses?







Prompt Engineering



Model Context Protocol

Anthropic MCP Spec


Pause!

Why

does the introduction of this tool change how we define our work?