Warm and Fuzzy

Semantic Search in .NET

Jonathan "J." Tower

Medication Status Lookup

Please enter the name of the medication below and click on a result in the dropdown below.
The medication database does not contain information on, or that applies to any dietary ingredient.
Search Tip: Search for the generic name first (acetaminophen). If the medication is not found, search for the brand name (Tylenol).

Medication Name

Medication name

Additional Information

Additional information, if any, will be displayed here.

TRAILHEAD
TECHNOLOGY PARTNERS

HOME | MEDICATION STATUS LOOKUP | SUPPLEMENT INFORMATION | ABOUT | JOSH+PRO...@TRAILHEADTECHNOLOGY.COM

Medication Status Lookup

Please enter the name of the medication below and click on a result in the dropdown below.
The medication database does not contain information on, or that applies to any dietary ingredient.
Search Tip: Search for the generic name first (acetaminophen). If the medication is not found, search for the brand name (Tylenol).

Medication Name

Medication name

Additional Information

Additional information, if any, will be displayed here.

tylenol ≠ acetaminophen

advil ≠ ibuprofen ≠ NSAID

Medication Status Lookup

Please enter the name of the medication below and click on a result in the dropdown below.
The medication database does not contain information on, or that applies to any dietary ingredient.
Search Tip: Search for the generic name first (acetaminophen). If the medication is not found, search for the brand name (Tylenol).

Medication Name

Medication name

Additional Information

Additional information, if any, will be displayed here.

Learn how to add
AI-powered
semantic search
to your **.NET apps**

TRAILHEAD
TECHNOLOGY PARTNERS

Jonathan "J." Tower

Partner & Principal Consultant

TRAILHEAD
TECHNOLOGY PARTNERS

- Microsoft MVP in .NET
- .NET Foundation Board of Directors
- jtower@trailheadtechnology.com
- trailheadtechnology.com/blog
- jtowermi
- Jonathan “J.” Tower

EXPERT
Consultation

bit.ly/th-offer

github.com/trailheadtechnology/api-security

The Evolution of Fuzzy Search

Exact Match

```
SELECT * FROM Products WHERE Name = 'car'
```

Exact Match

```
SELECT * FROM Products WHERE Name = 'car'
```

User frustration:

"automobile" won't match "car"
"cra" won't match "car"
"ca" won't match "car"
"kar" won't match "car"

Exact Match

```
SELECT * FROM Products WHERE Name = 'car'
```

User frustration:

“automobile” won’t match “car”

“cra” won’t match ”car”

“ca” won’t match ”car”

“kar” won’t match ”car”

Exact Match

```
SELECT * FROM Products WHERE Name = 'car'
```

User frustration:

“automobile” won’t match “car”

“cra” won’t match ”car”

“ca” won’t match ”car”

“kar” won’t match ”car”

Exact Match

```
SELECT * FROM Products WHERE Name = 'car'
```

User frustration:

“automobile” won’t match “car”

“cra” won’t match ”car”

“ca” won’t match ”car”

“kar” won’t match ”car”

LIKE Queries

```
SELECT * FROM Products WHERE Name LIKE '%car%’
```

LIKE Queries

```
SELECT * FROM Products WHERE Name LIKE '%car%’
```

User frustration:

“automobile” won’t match “car”

“cra” won’t match ”car”

“kar” won’t match ”car”

LIKE Queries

```
SELECT * FROM Products WHERE Name LIKE '%car%’
```

User frustration:

“automobile” won’t match “car”

“cra” won’t match ”car”

“kar” won’t match ”car”

LIKE Queries

```
SELECT * FROM Products WHERE Name LIKE '%car%’
```

User frustration:

“automobile” won’t match “car”

“cra” won’t match ”car”

“kar” won’t match ”car”

Levenshtein

`Levenshtein("kitten”, "sitting”) = 3`

L e v e n s h t e i n

L e v e n s h t e i n

		L	e	v	e	n	s	h	t	e	i	n
	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5
L	0.5	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
e	1	0.5	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5
v	1.5	1	0.5	0	0.5	1	1.5	2	2.5	3	3.5	4
e	2	1.5	1	0.5	0	0.5	1	1.5	2	2.5	3	3.5
n	2.5	2	1.5	1	0.5	0	0.5	1	1.5	2	2.5	3
s	3	2.5	2	1.5	1	0.5	0	0.5	1	1.5	2	2.5
h	3.5	3	2.5	2	1.5	1	0.5	0	0.5	1	1.5	2
t	4	3.5	3	2.5	2	1.5	1	0.5	0	0.5	1	1.5
e	4.5	4	3.5	3	2.5	2	1.5	1	0.5	0	0.5	1
i	5	4.5	4	3.5	3	2.5	2	1.5	1	0.5	0	0.5
n	5.5	5	4.5	4	3.5	3	2.5	2	1.5	1	0.5	0

Video: Substitution as 1 and cost of deletion or insertion as 0.5

Rules (configurable):

Substitutions cost 1

Deletion or insertion costs 1

Ex:

kitten → **s**itten: 1

sitt**e**n → sitt**i**n: 1

sittin → sittin**g**: 1

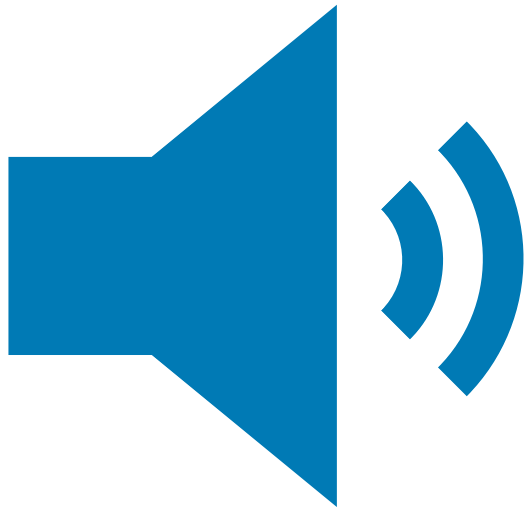
Total: 3

Soundex

Creates **4-character code** based on how they **sound**, not how they're spelled

Rules:

- Keeps the first letter of the word
- Converts the rest into numbers representing consonant sounds
- Drops vowels and silent letters
- Words that sound similar → same code



Ex:
"Smith" → S530
"Smyth" → S530

"Robert" → R163
"Rupert" → R163

From Fuzzy to Semantic Search

Fuzzy Search

Find things that **look similar**

From Fuzzy to Semantic Search

Fuzzy Search

Find things that **look similar**

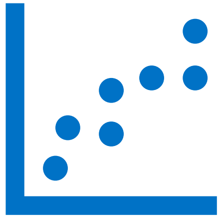
Semantic Search

Find things that **mean the same**

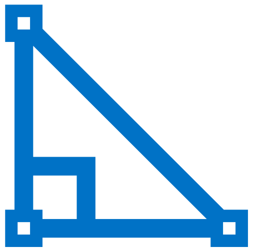
Core Concepts of Semantic Search

Core Concepts of Semantic Search

Vectors



Embeddings

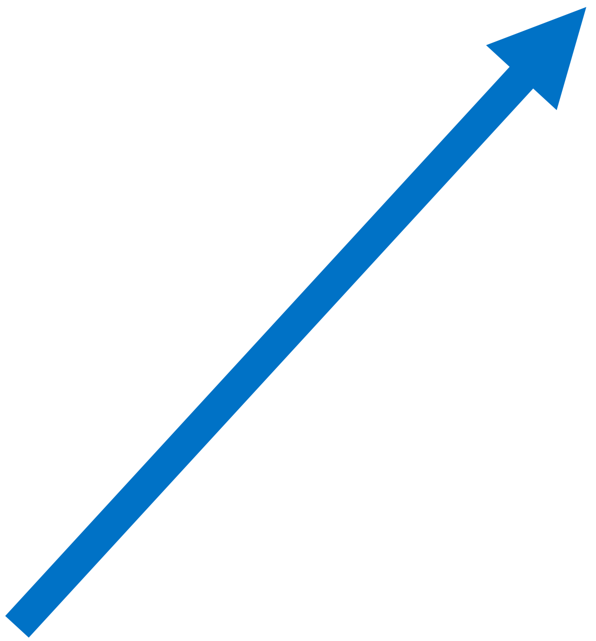


Cosine Differences

Vectors

Vectors

1. Direction
2. Magnitude

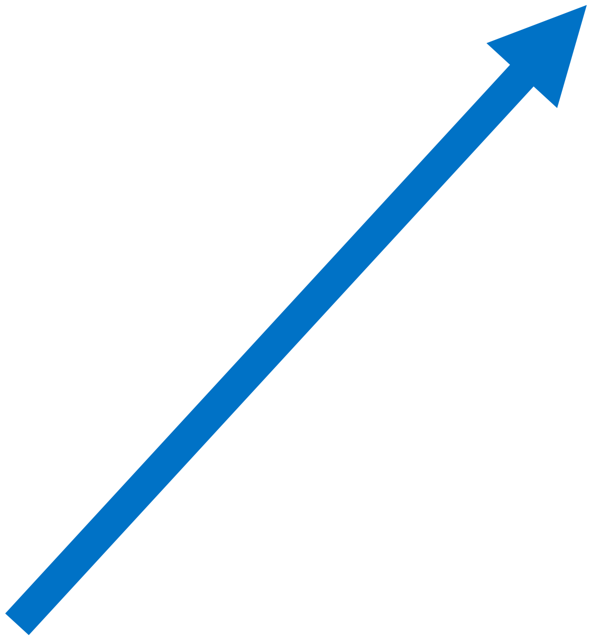


Vectors

1. Direction
2. Magnitude

Ex:

1 mile northwest



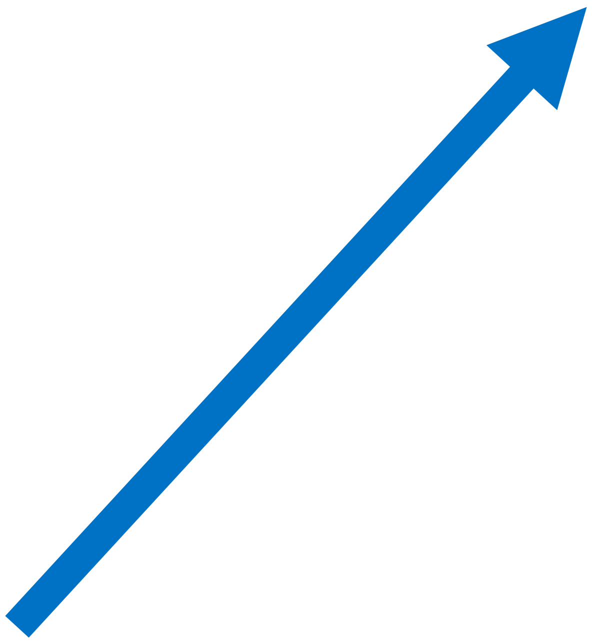
Vectors

1. Direction
2. Magnitude

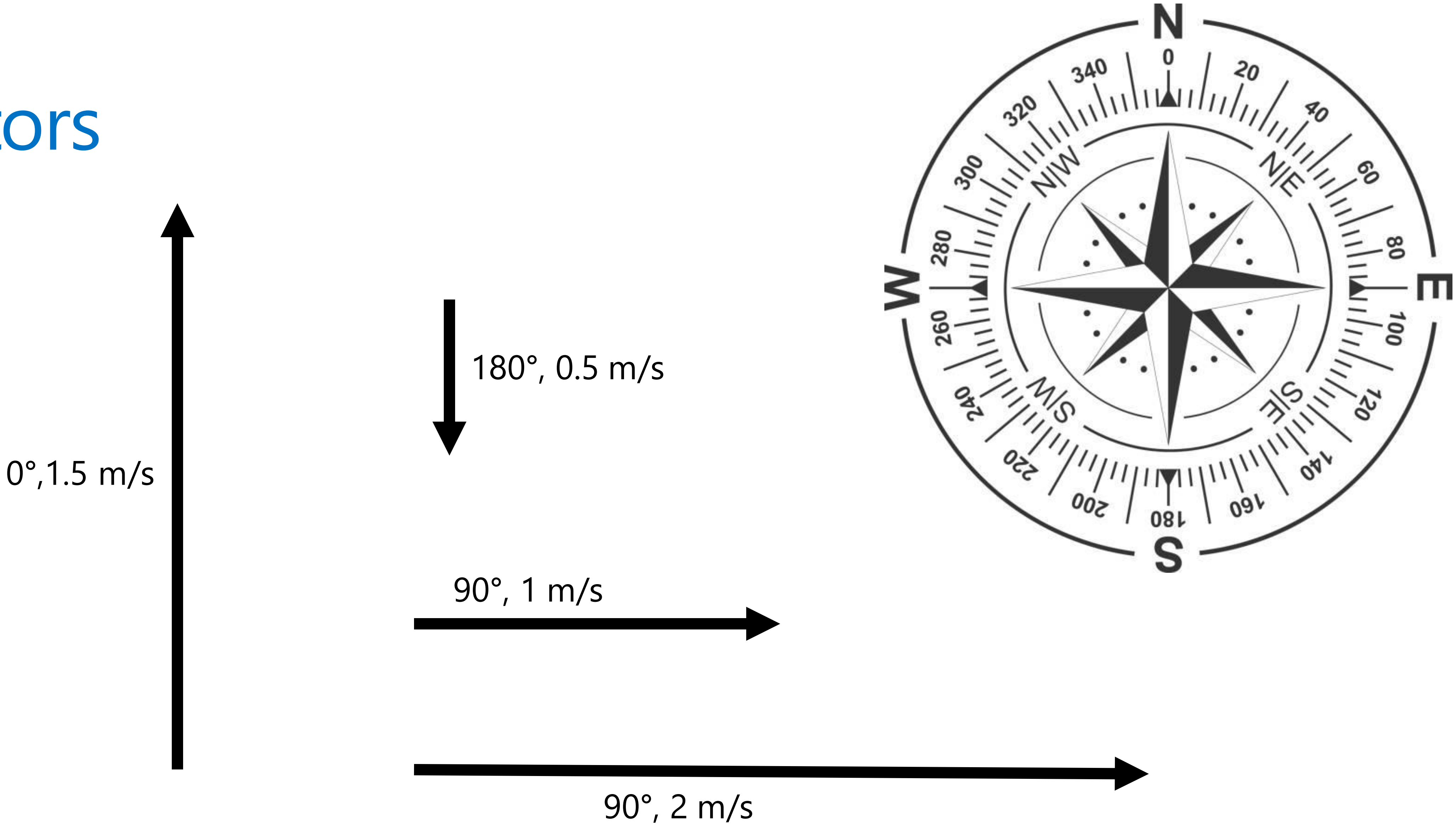
Ex:

1 mile northwest

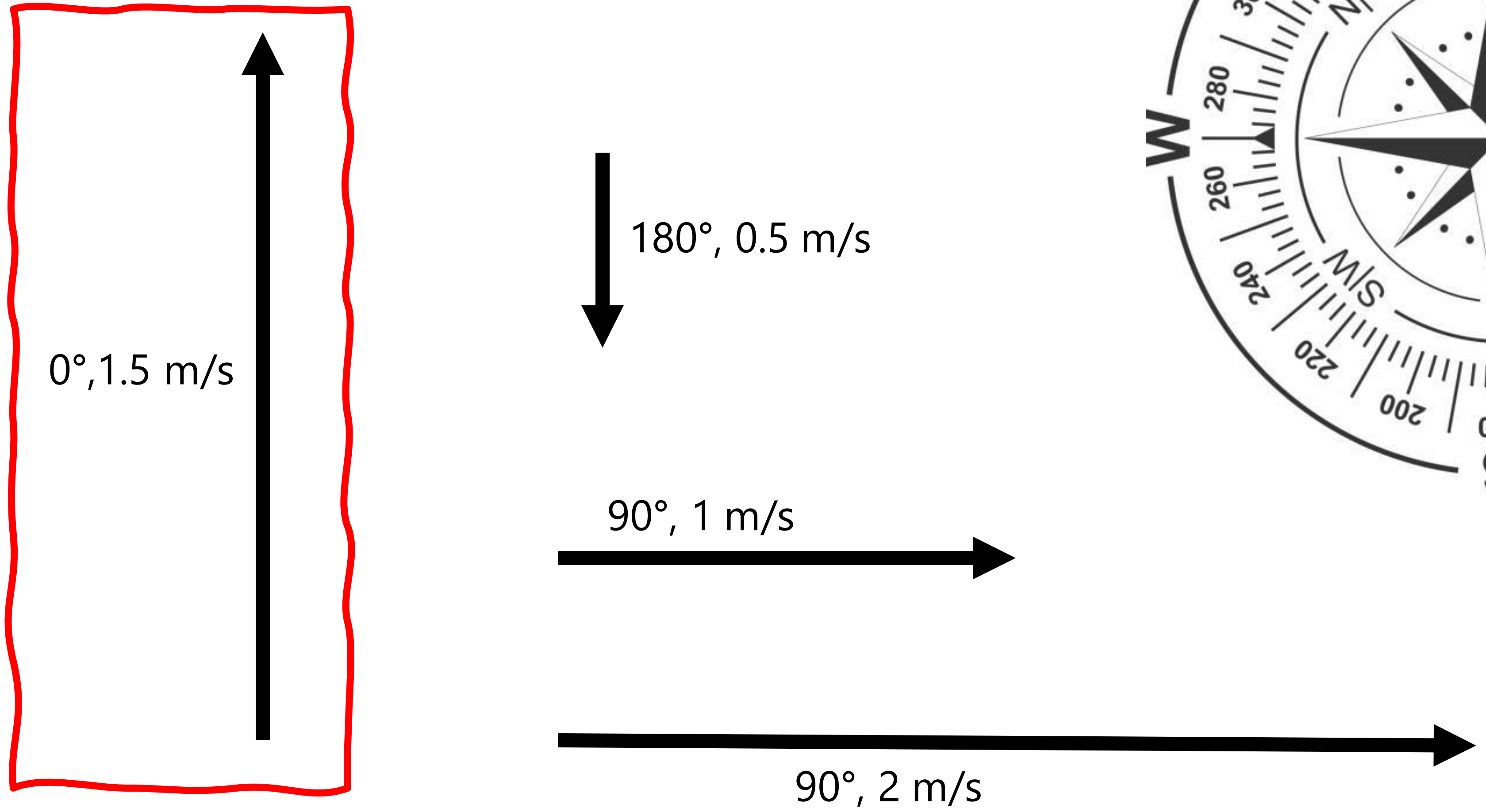
1 m/s at 130°

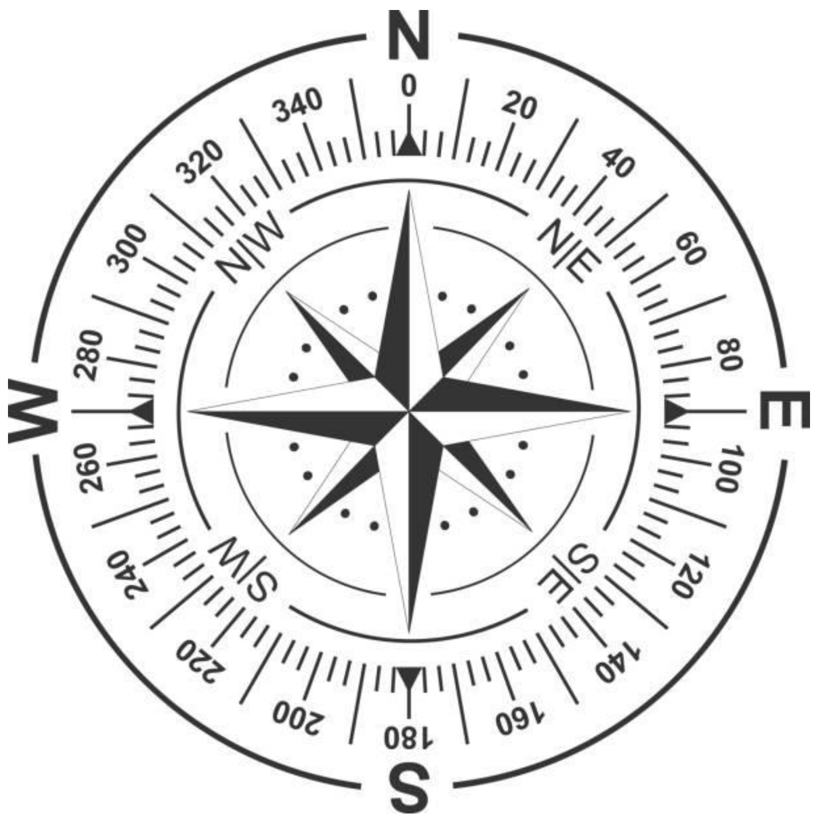


Vectors

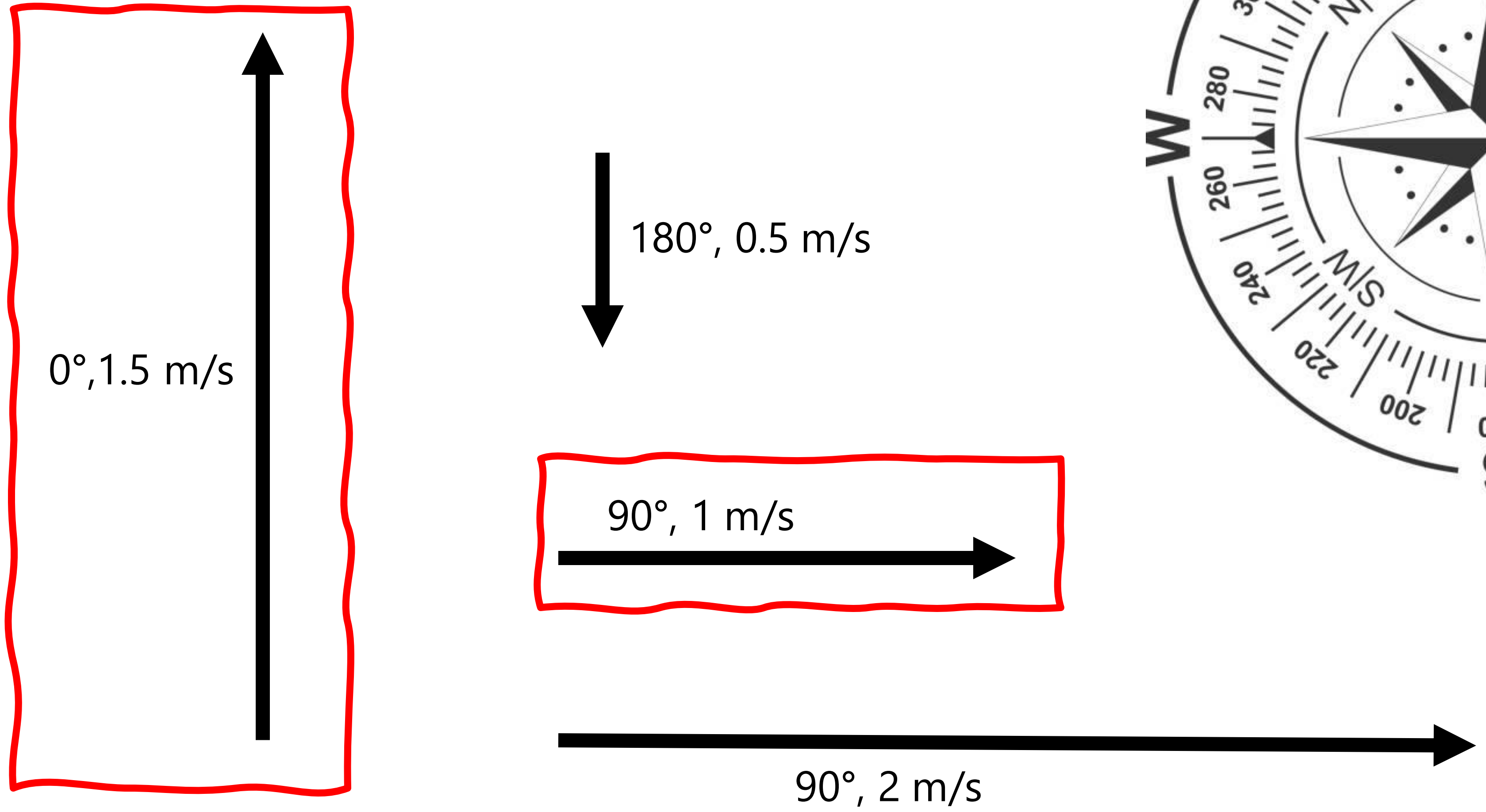


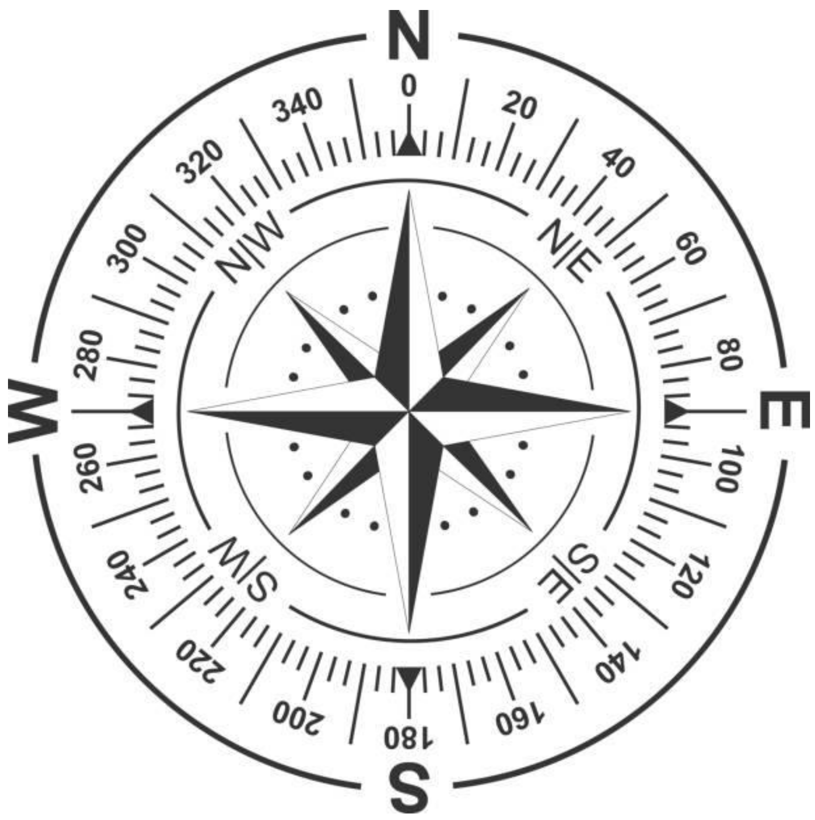
Vectors


0°,1.5 m/s
180°, 0.5 m/s
90°, 1 m/s
90°, 2 m/s

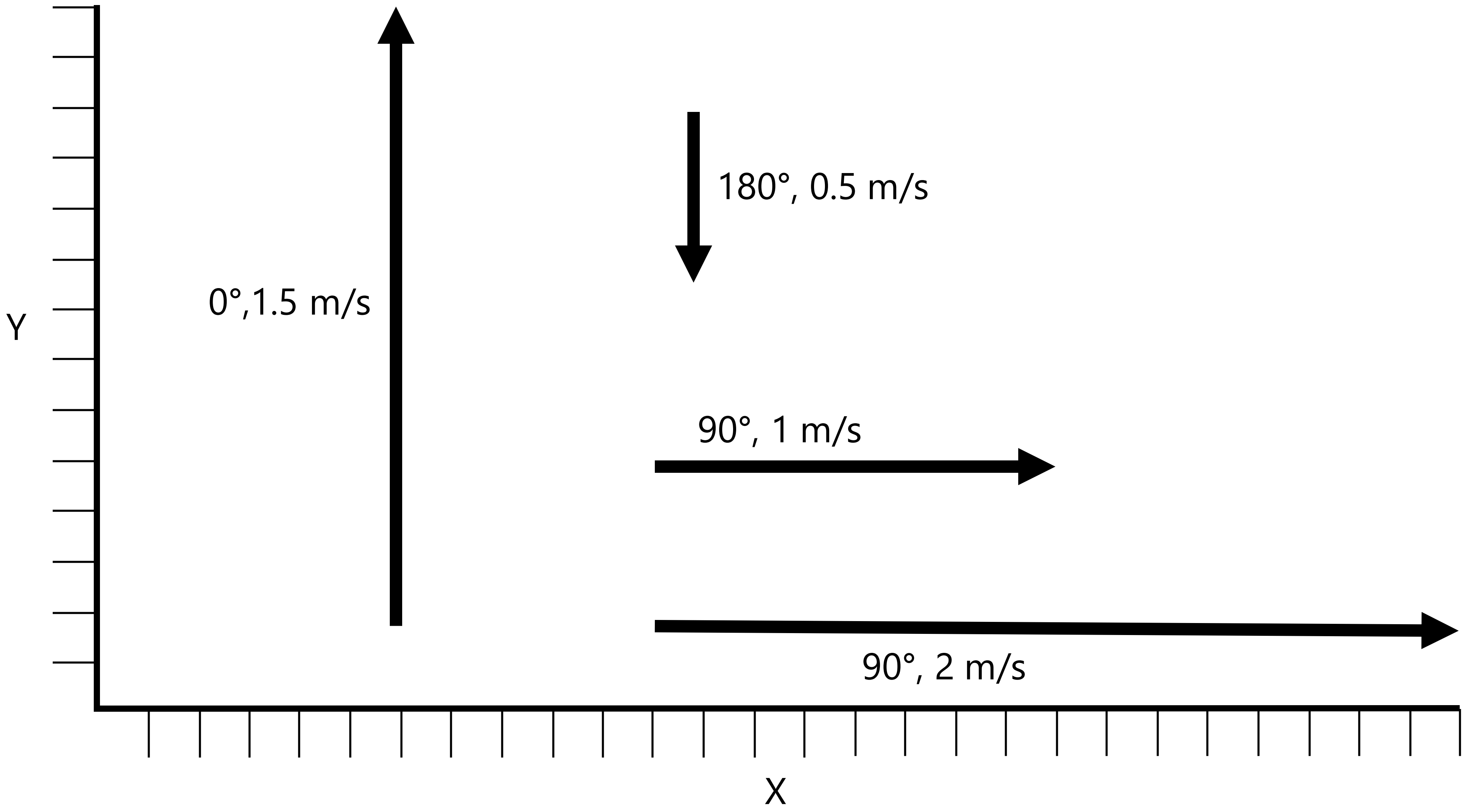

N
E
S
W
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
N/E
S/E
S/W
N/W

Vectors

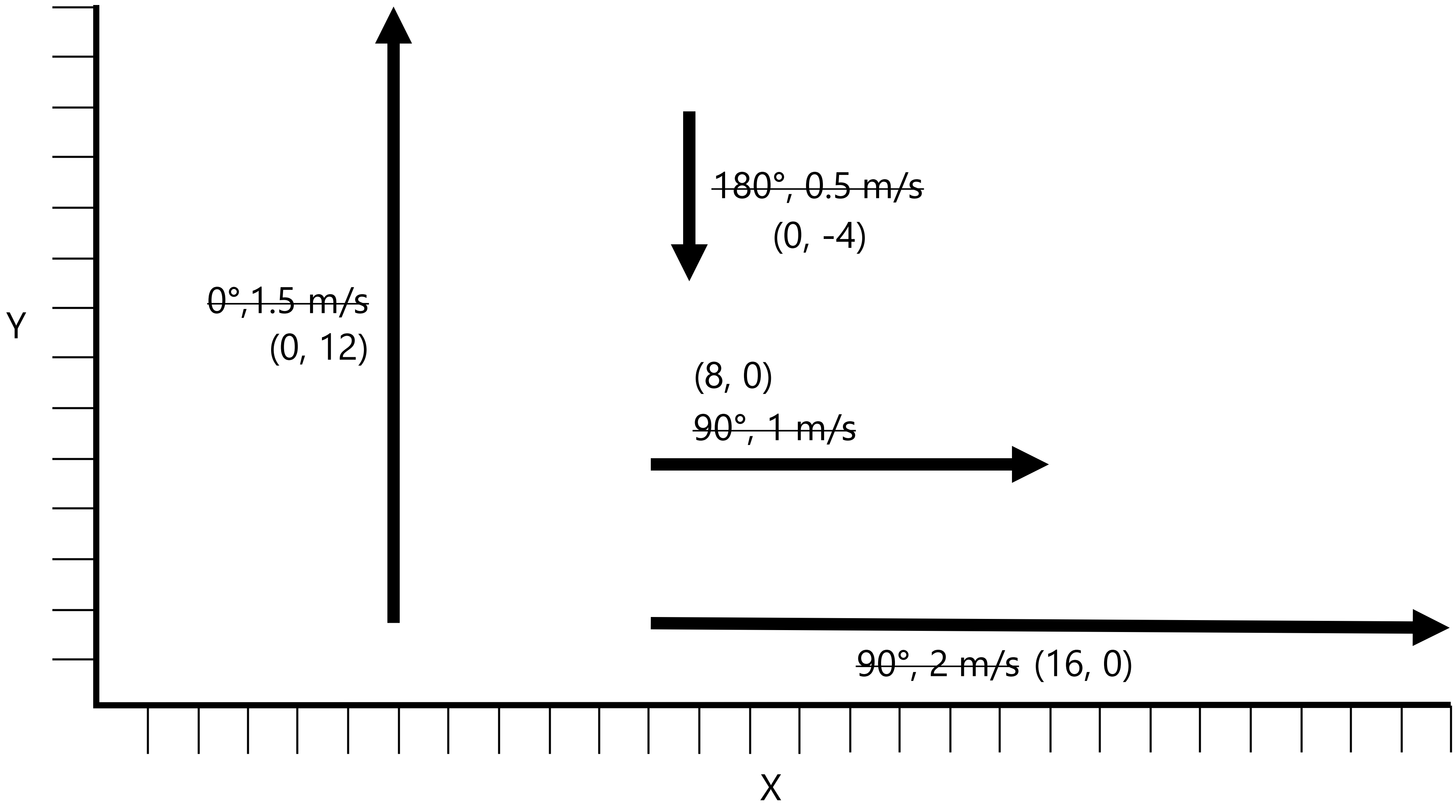




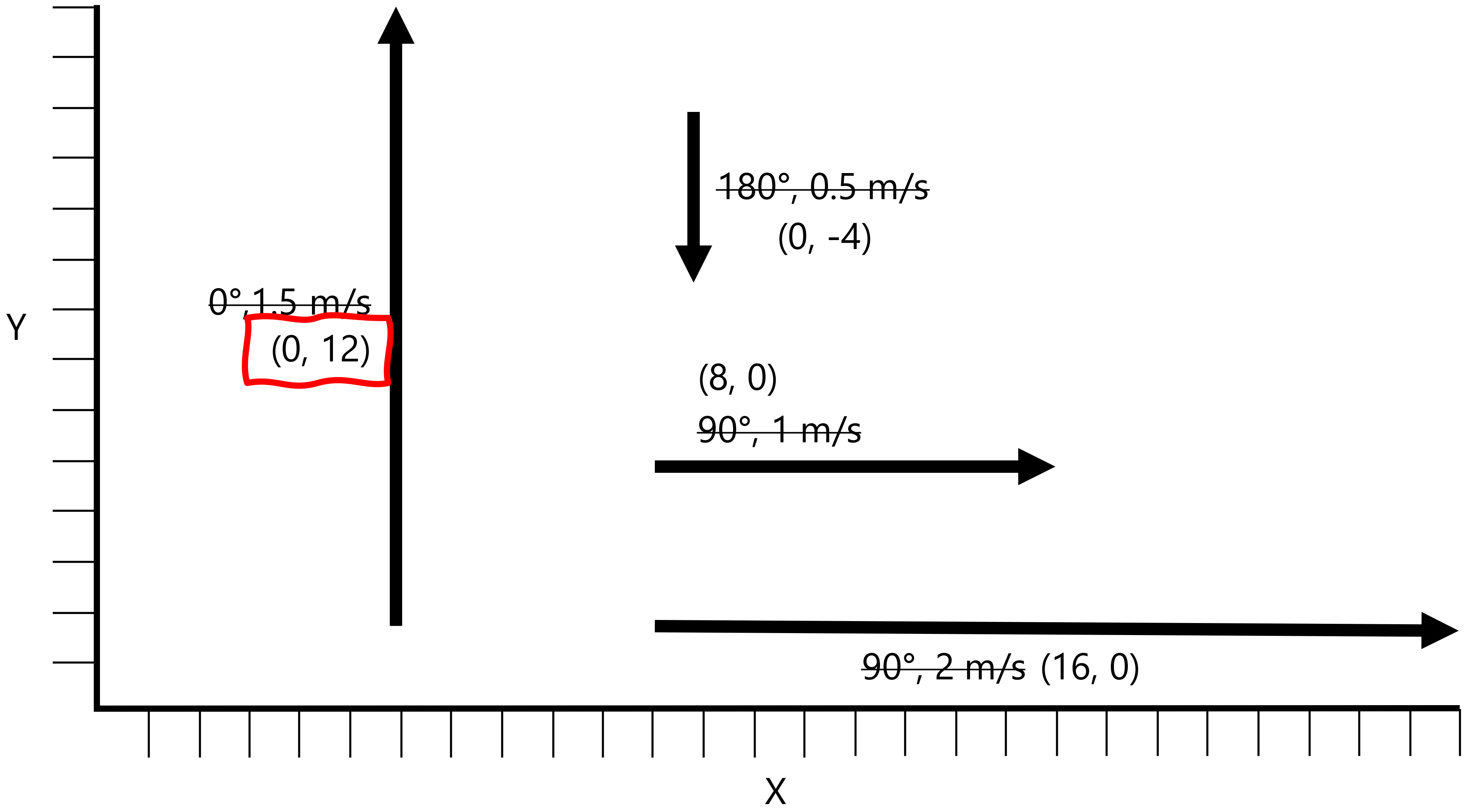
Vectors



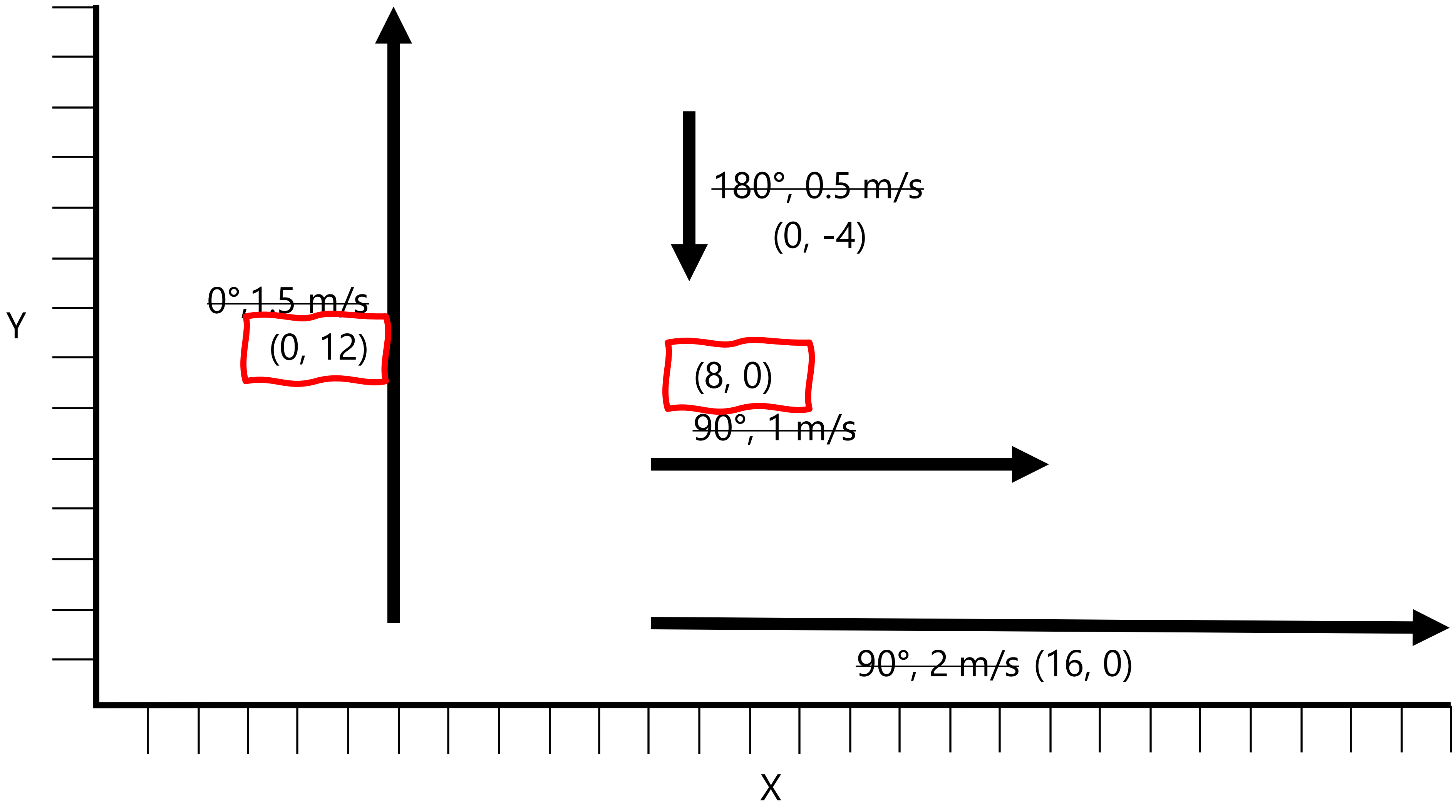
Vectors



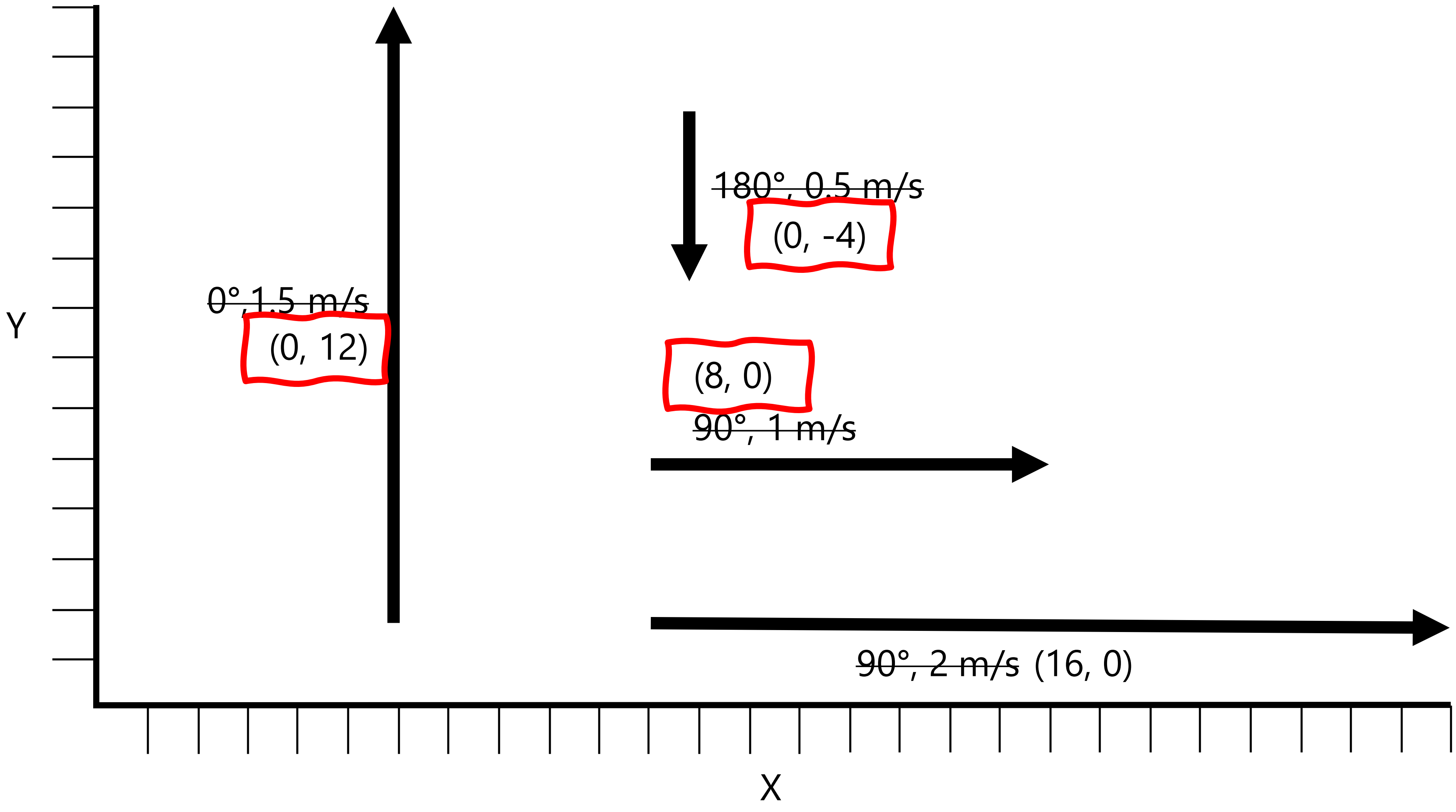
Vectors



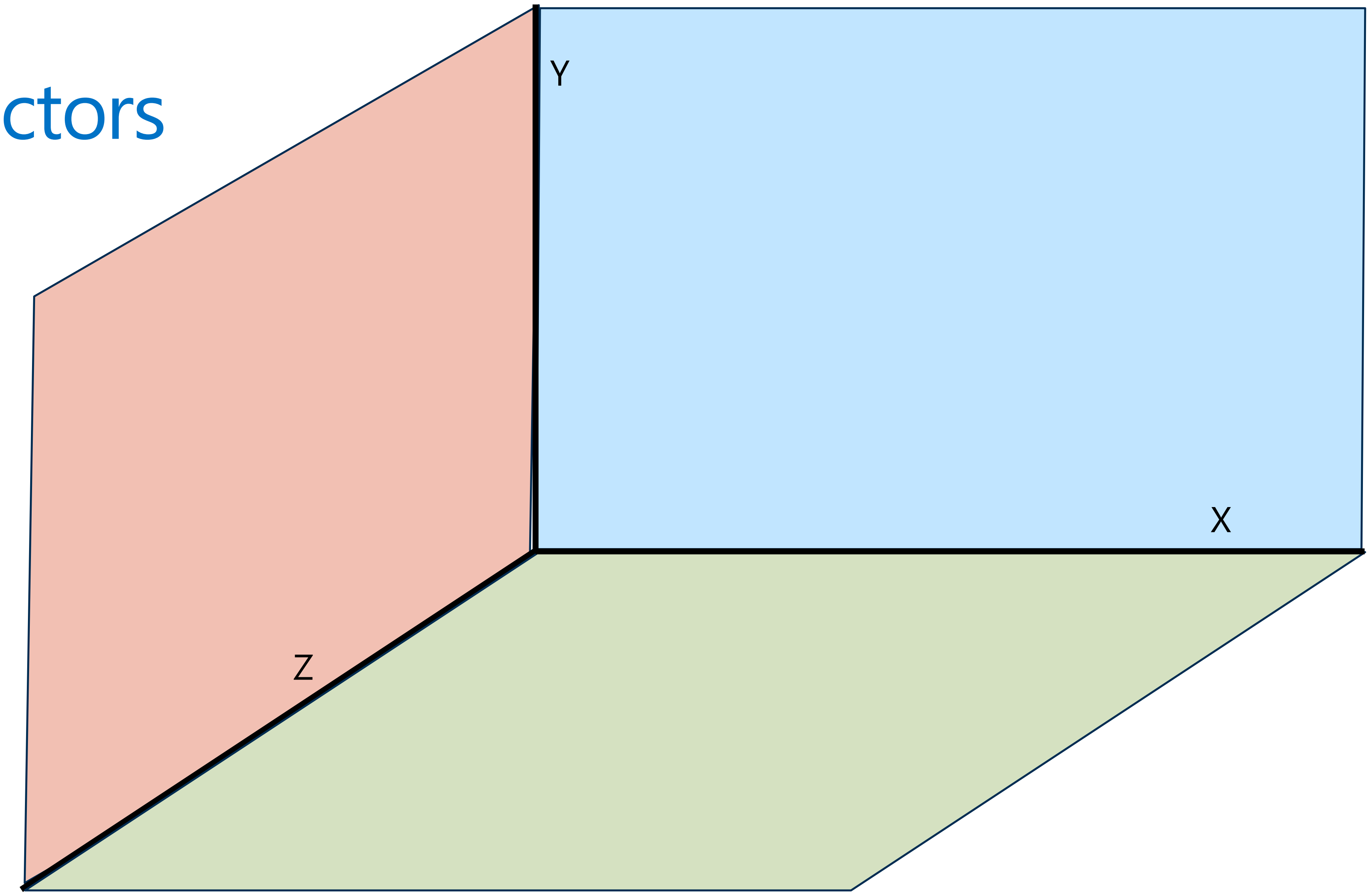
Vectors



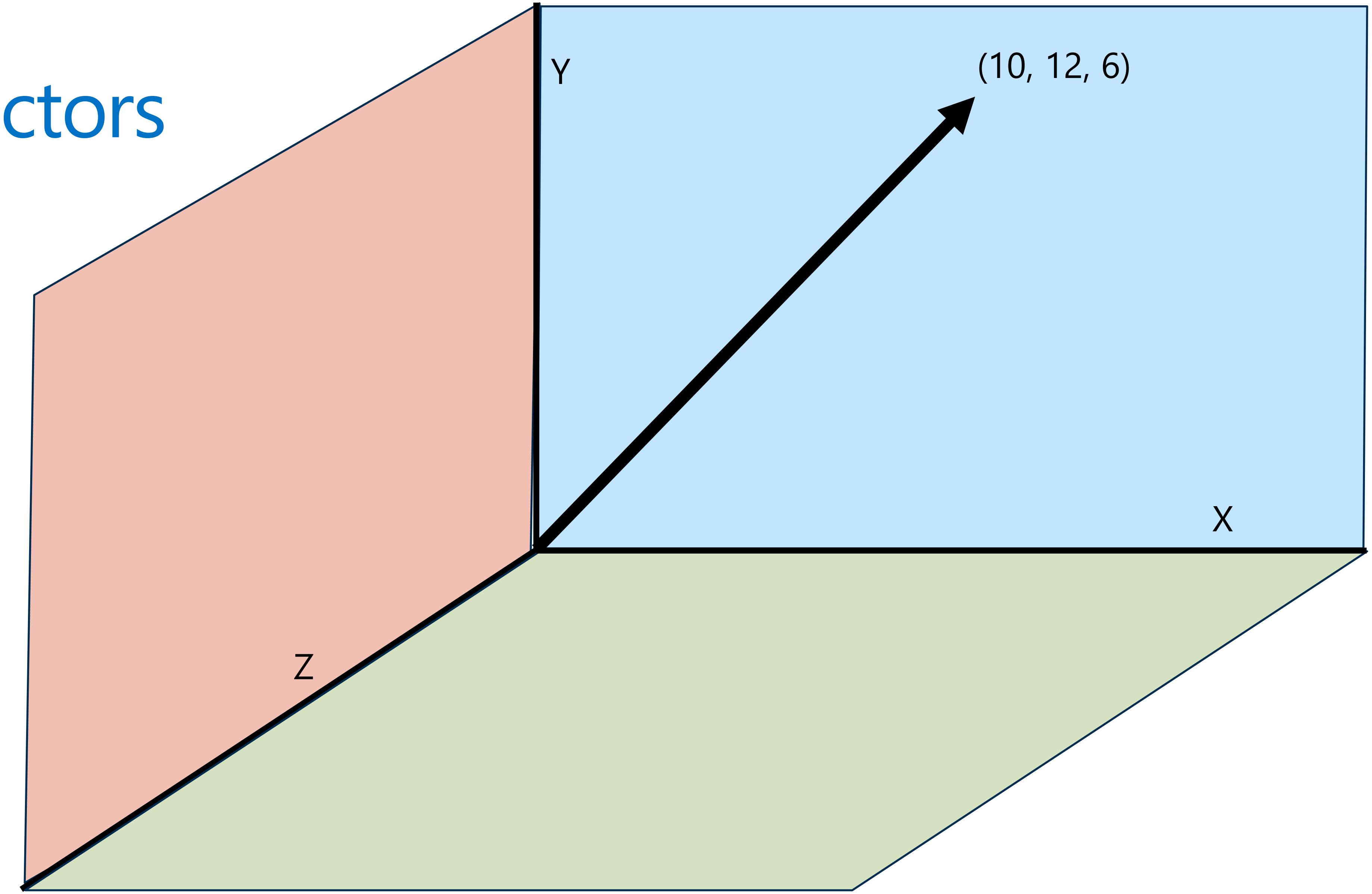
Vectors



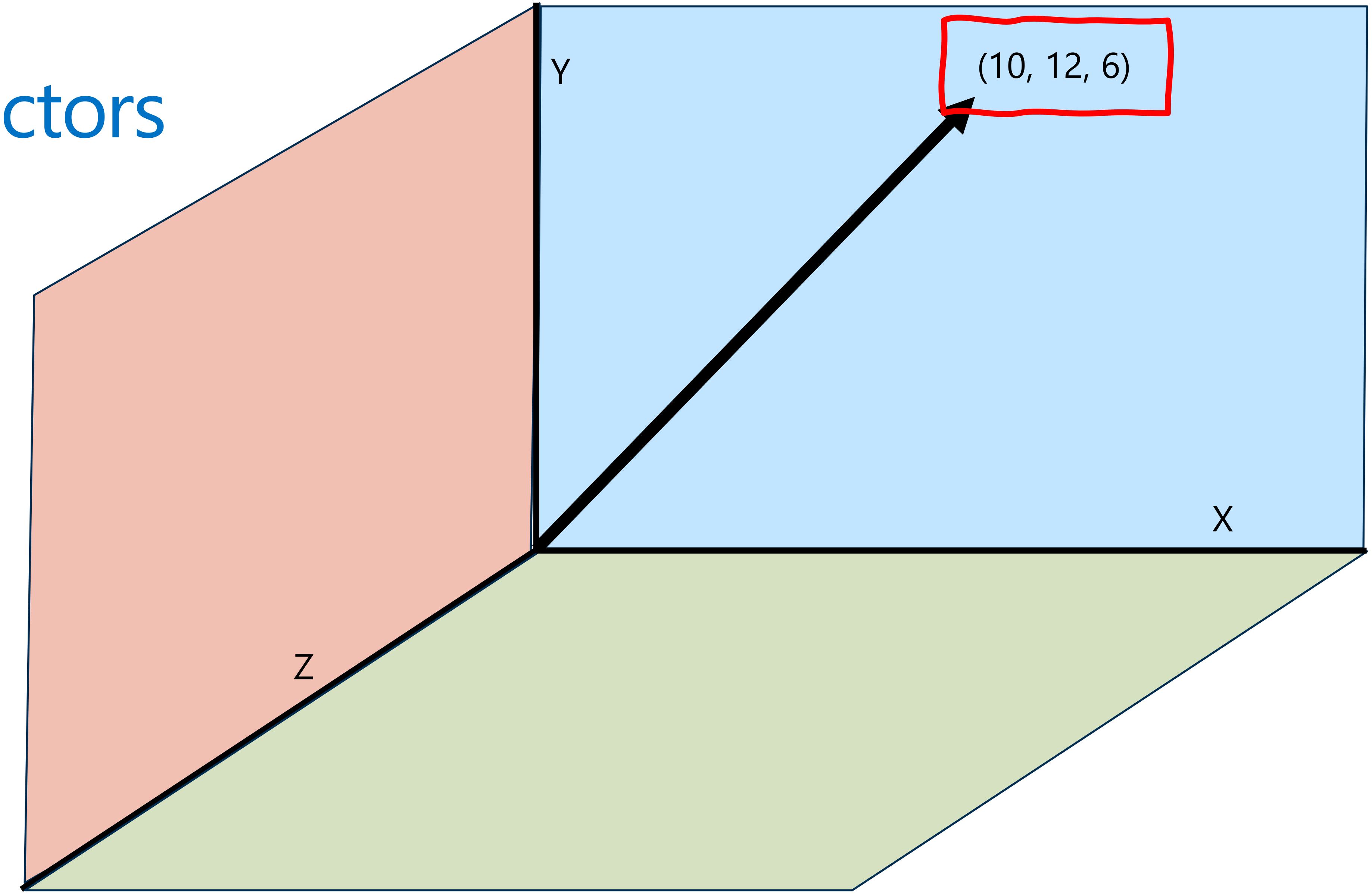
3D Vectors



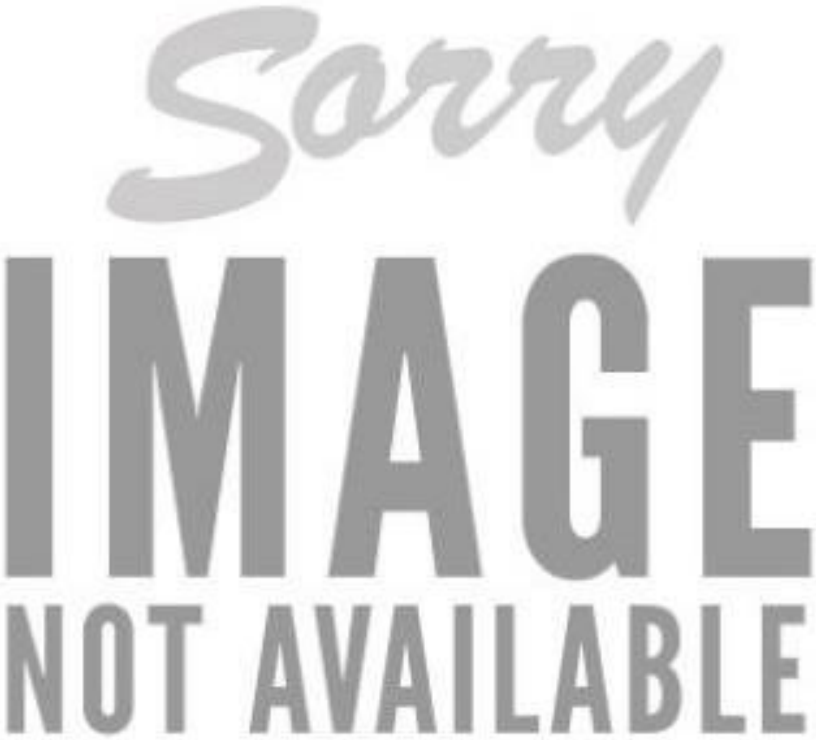
3D Vectors



3D Vectors



4D+ Vectors



4D+ Vectors

Dimensions	Sample Vector
2	(10, 12)
3	(10, 12, 6)
4	(10, 12, 6, 4)
5	(10, 12, 6, 4, 10)
6	(10, 12, 6, 4, 10, 3)
7	(10, 12, 6, 4, 10, 3, 144)
N	(10, 12, 6, 4, 10, 3, 144, ...)

4D+ Vectors

Dimensions	Sample Vector
2	(10, 12)
3	(10, 12, 6)
4	(10, 12, 6, 4)
5	(10, 12, 6, 4, 10)
6	(10, 12, 6, 4, 10, 3)
7	(10, 12, 6, 4, 10, 3, 144)
N	(10, 12, 6, 4, 10, 3, 144, ...)

LLMS today:
384 to 3,000
dimension vectors

Embeddings

Embeddings

Storing meaning using vectors

An **embedding** is just a **vector** that points in a direction **representing meaning**

The **closer** two embeddings point in the same direction, the more **similar their meaning**

Embeddings

Storing meaning using vectors

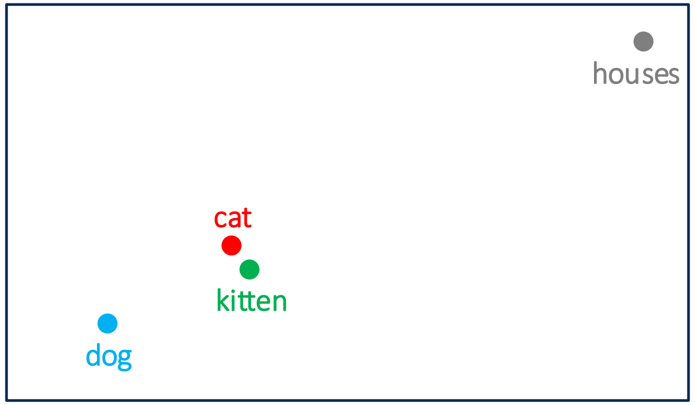
An **embedding** is just a **vector** that points in a direction **representing meaning**

The **closer** two embeddings point in the **same direction**, the more **similar their meaning**

Embeddings

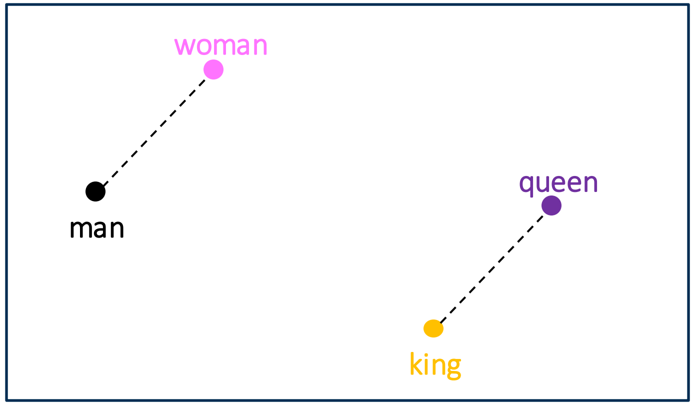
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

→ Simplified for Visualization



man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

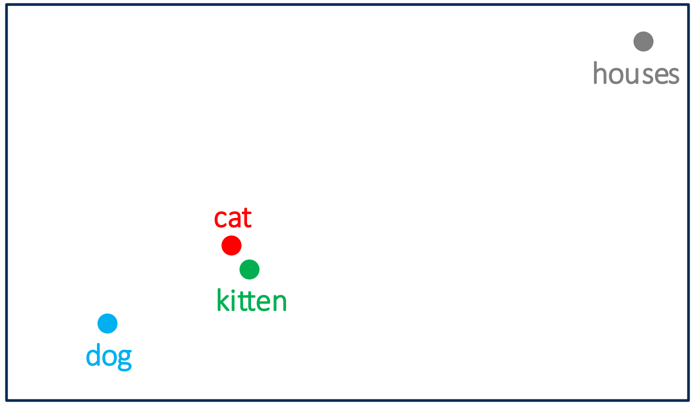
→ Simplified for Visualization



Embeddings

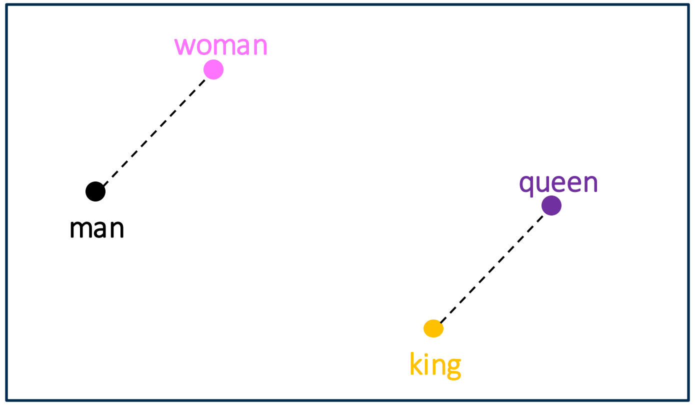
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

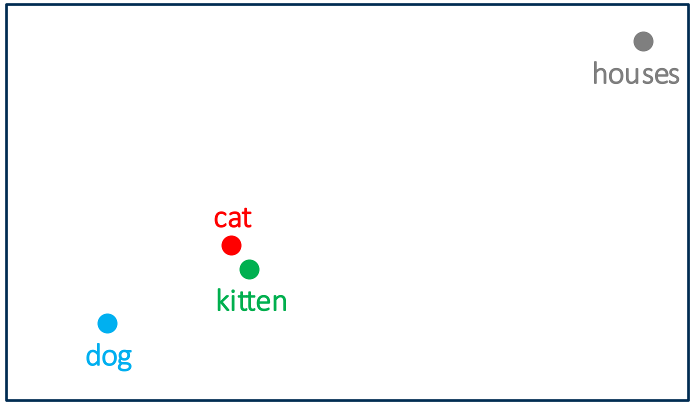
Simplified for Visualization



Embeddings

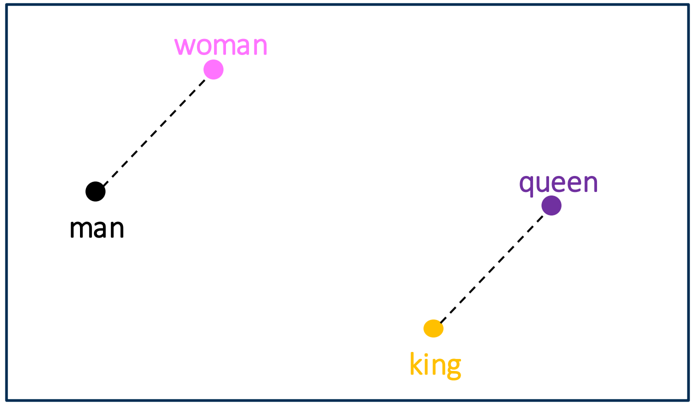
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



	Living being	Feline	Human	Gender	Royalty	Verb	Plural
man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

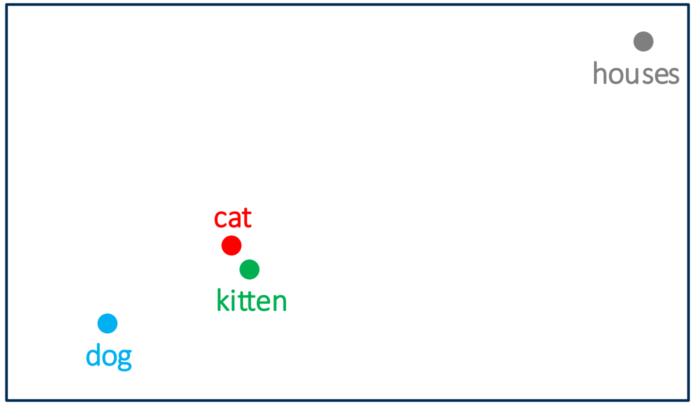
Simplified for Visualization



Embeddings

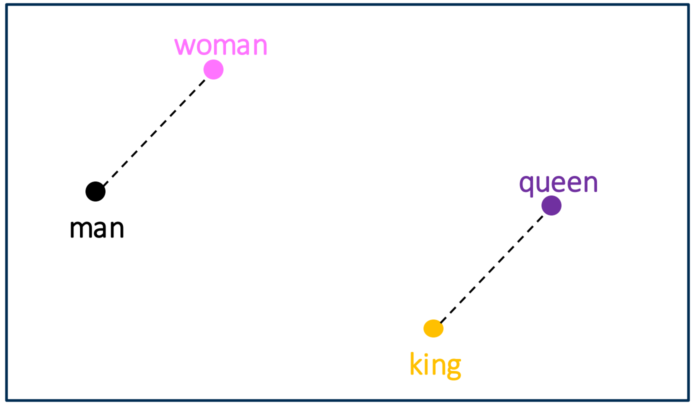
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

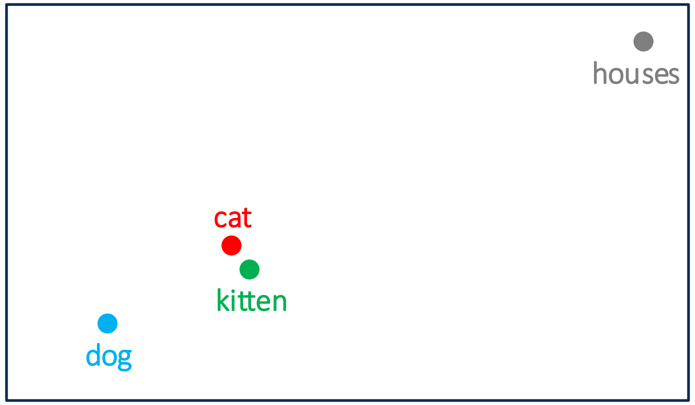
Simplified for Visualization



Embeddings

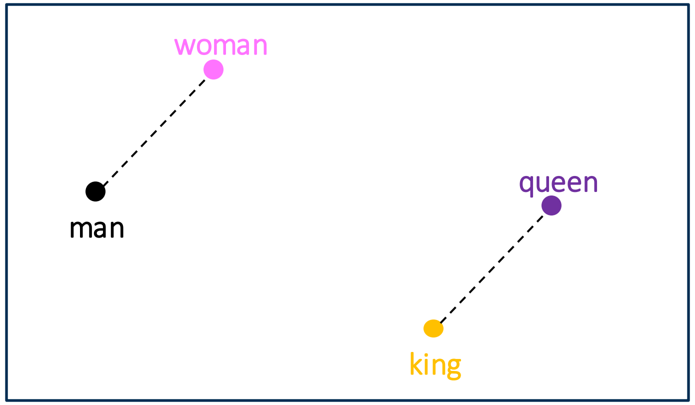
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



	Living being	Feline	Human	Gender	Royalty	Verb	Plural
man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

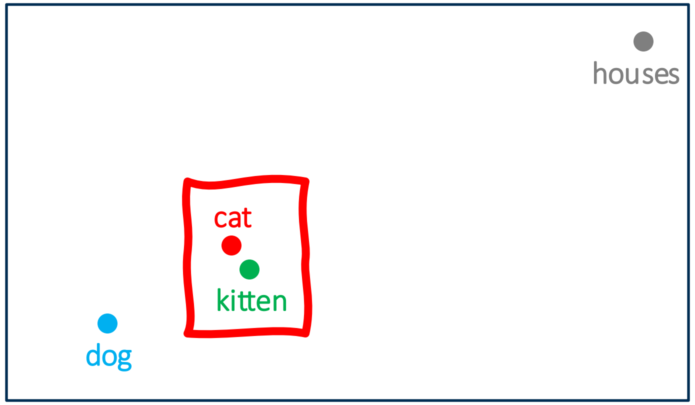
Simplified for Visualization



Embeddings

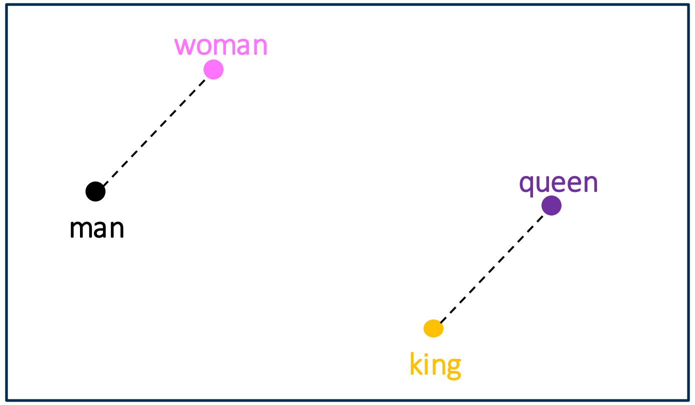
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

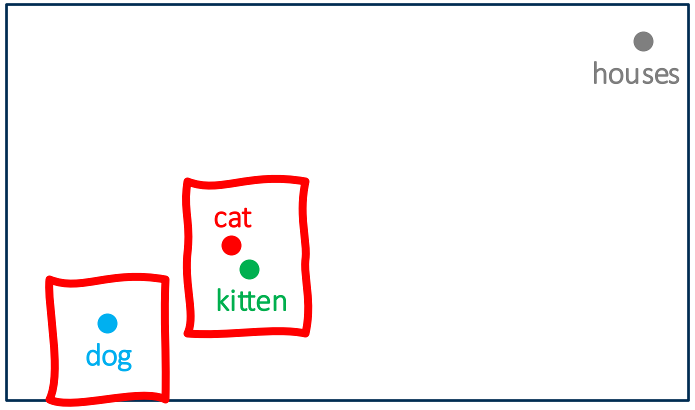
Simplified for Visualization



Embeddings

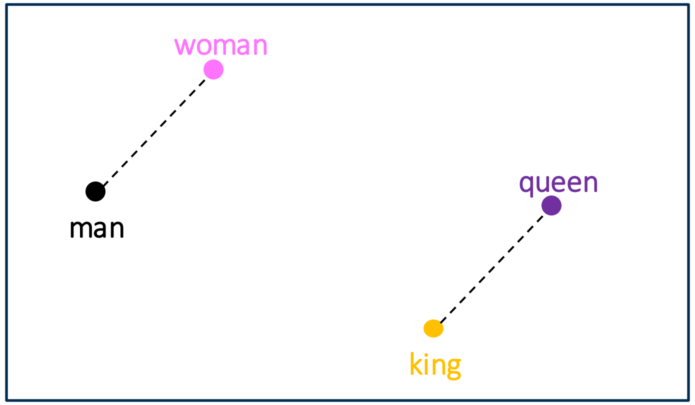
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

Simplified for Visualization



Embeddings

Storing meaning using vectors

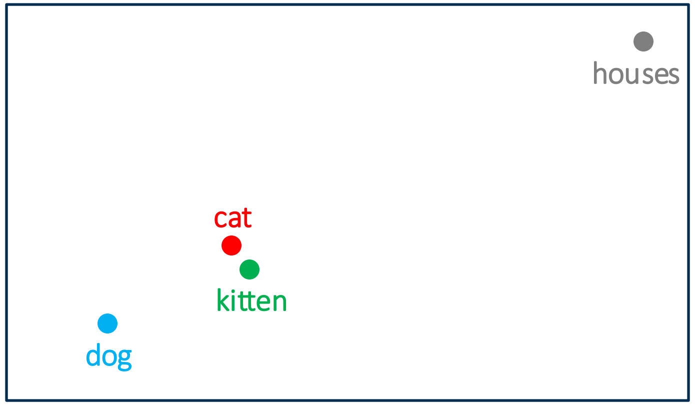
An **embedding** is just a **vector** that points in a direction **representing meaning**

The **closer** two embeddings point in the same direction, the more **similar their meaning**

Embeddings

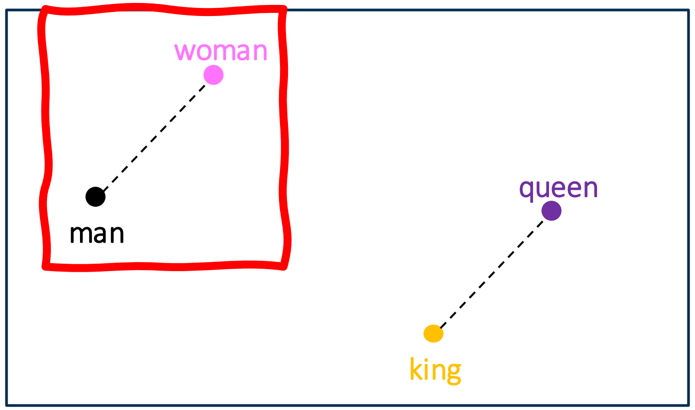
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

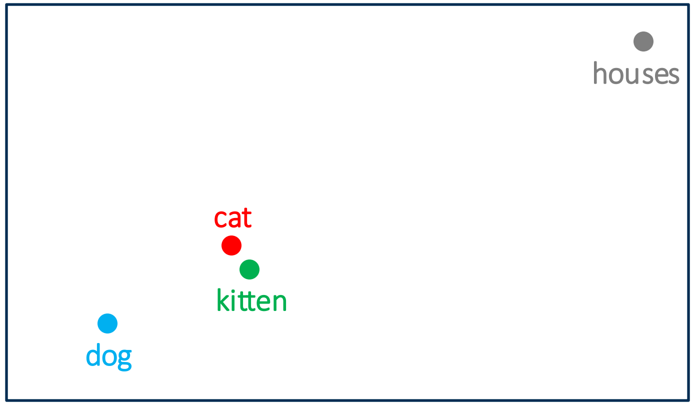
Simplified for Visualization



Embeddings

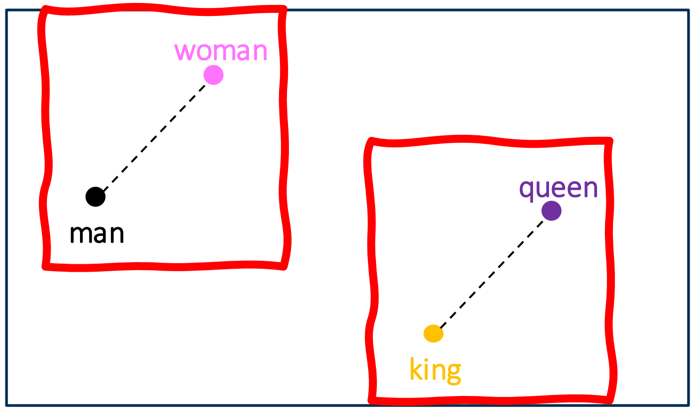
	Living being	Feline	Human	Gender	Royalty	Verb	Plural
cat →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
kitten →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
dog →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
houses →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

Simplified for Visualization



man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman →	0.7	-0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.6	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.9	-0.5	-0.9

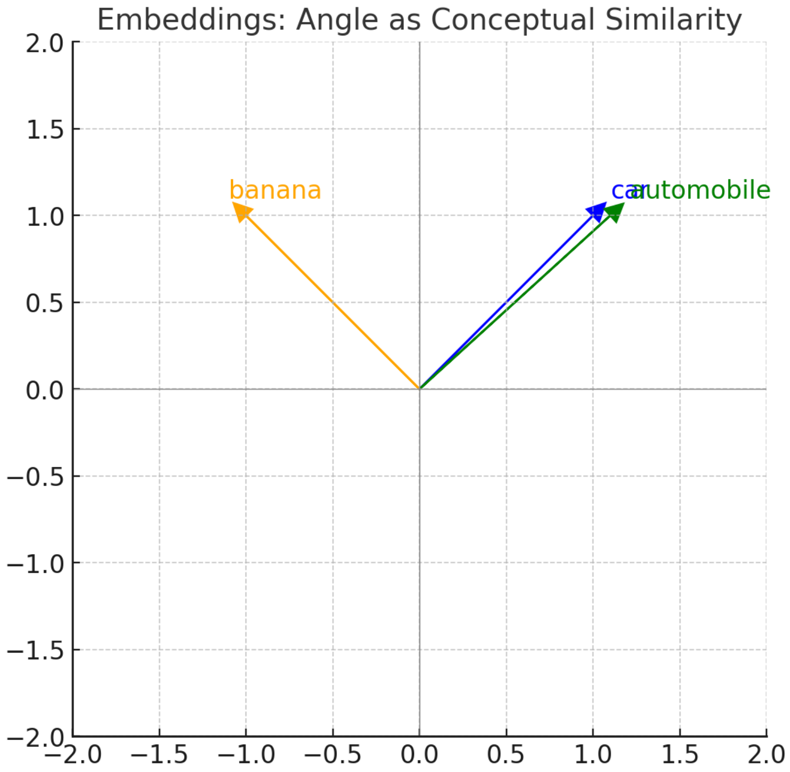
Simplified for Visualization



Cosine Similarity

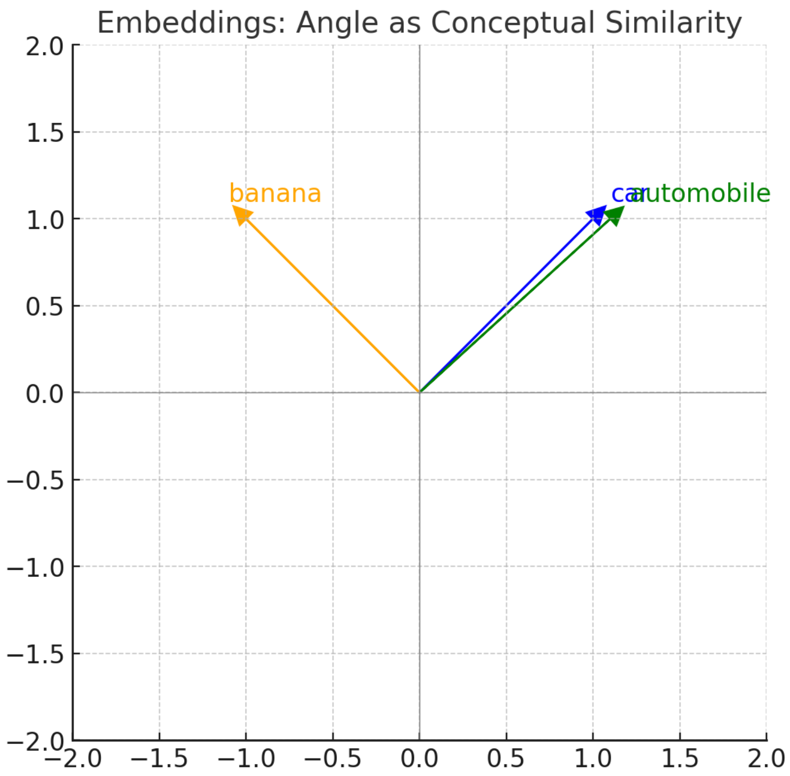
Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$



Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\ ||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

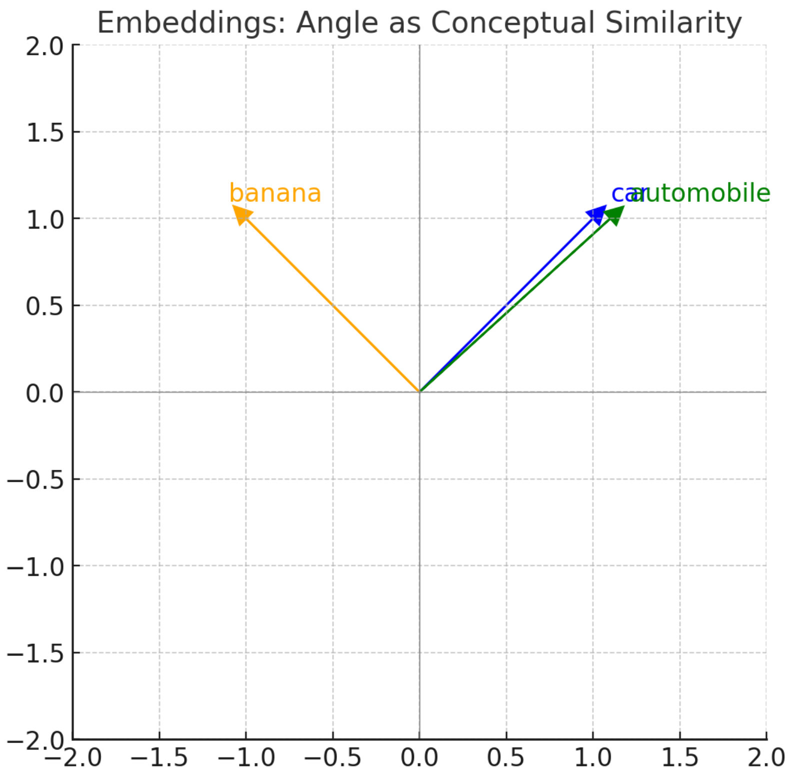
$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$

Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

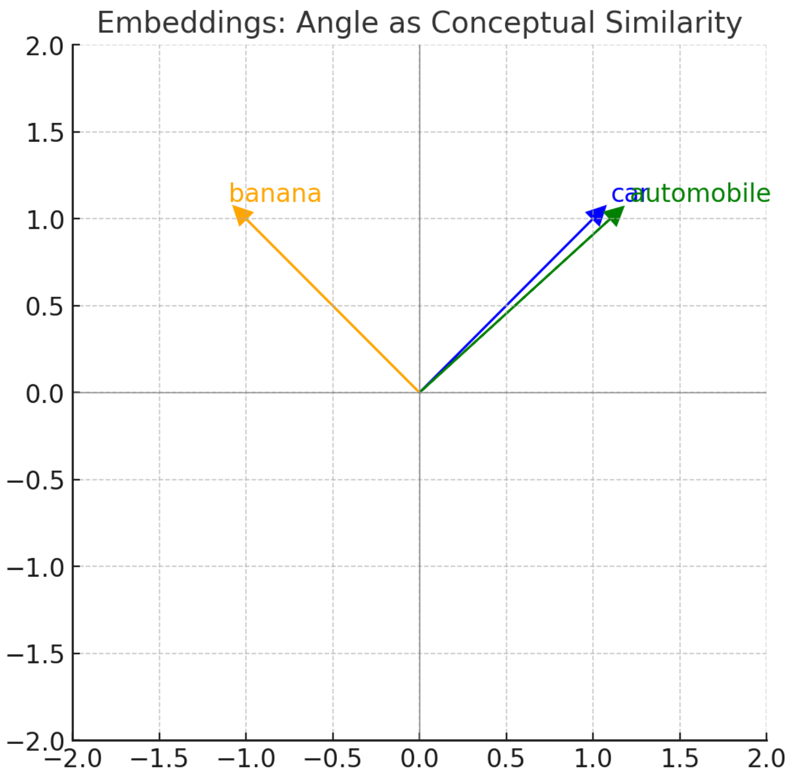
$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$

Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \, ||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

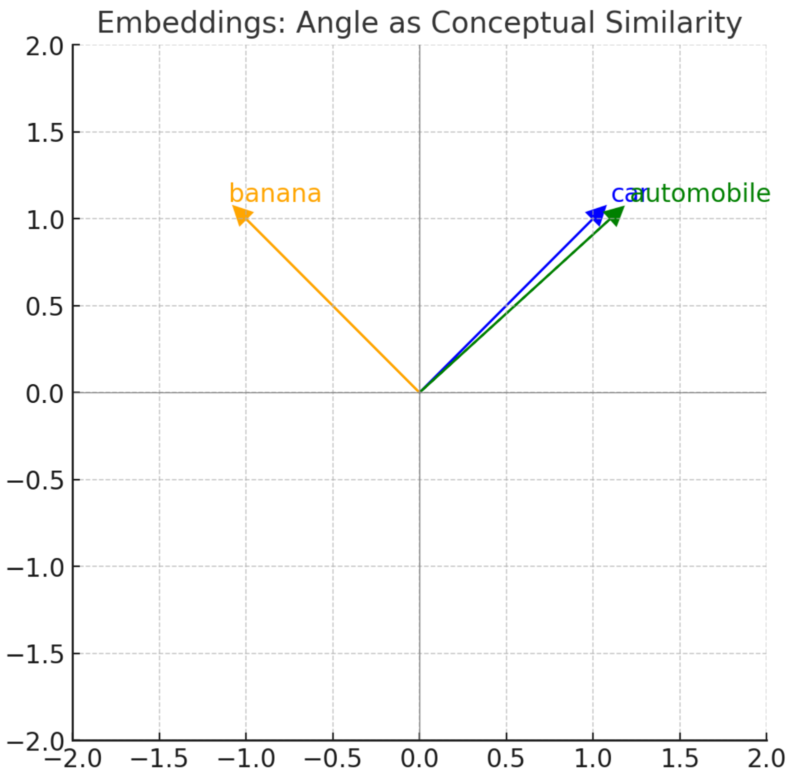
$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$

Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

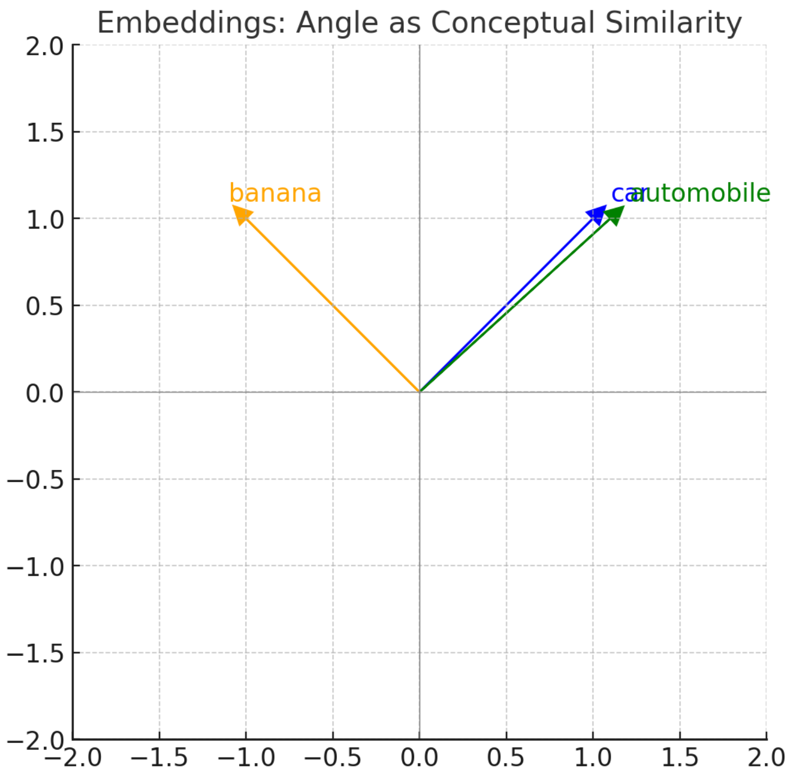
$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$

Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

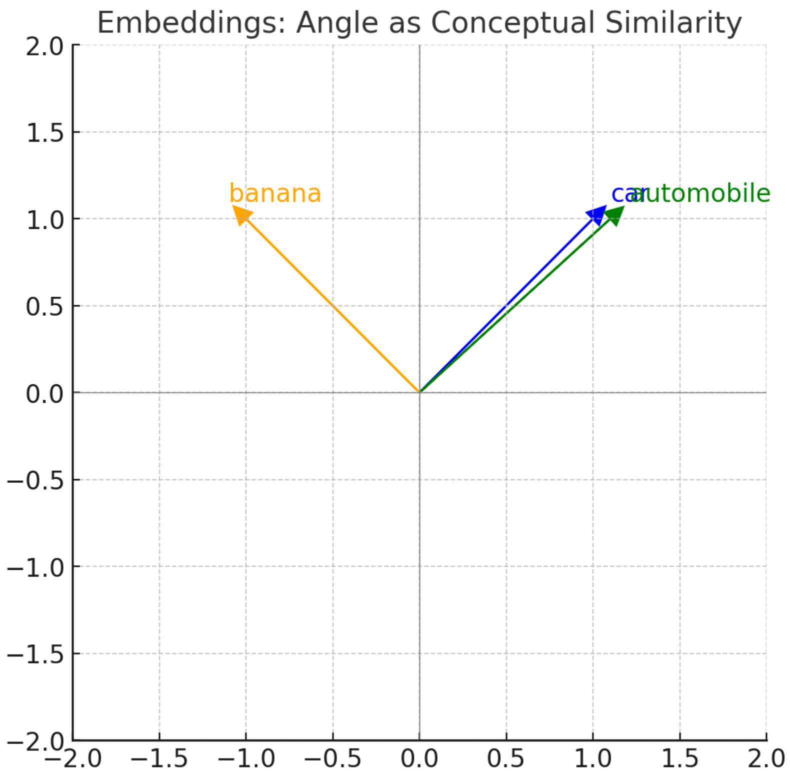
$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$

Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

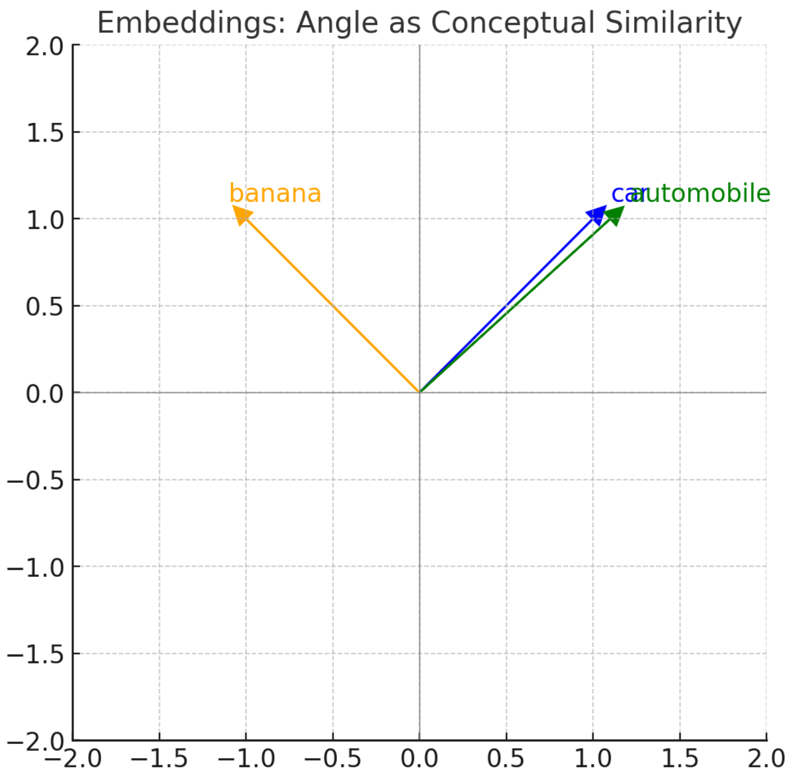
$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$

Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

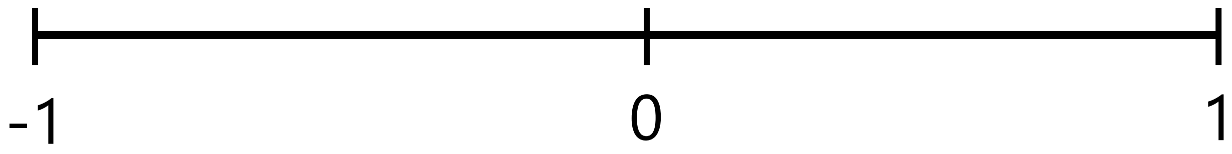
$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

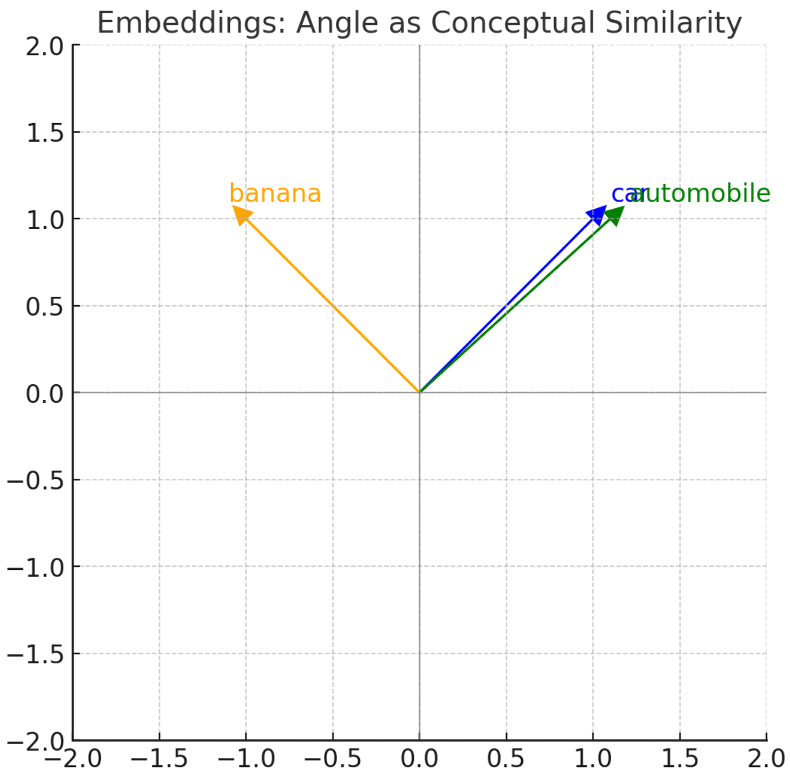
$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$



Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$



$$A = [2, 3], \quad B = [4, -1]$$

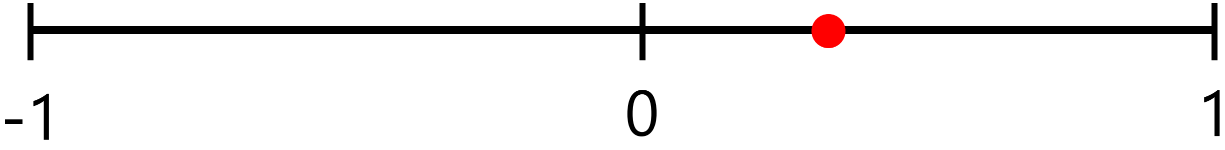
$$A \cdot B = (2 \times 4) + (3 \times -1) = 8 - 3 = 5$$

$$||A|| = \sqrt{2^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13} \approx 3.606$$

$$||B|| = \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \approx 4.123$$

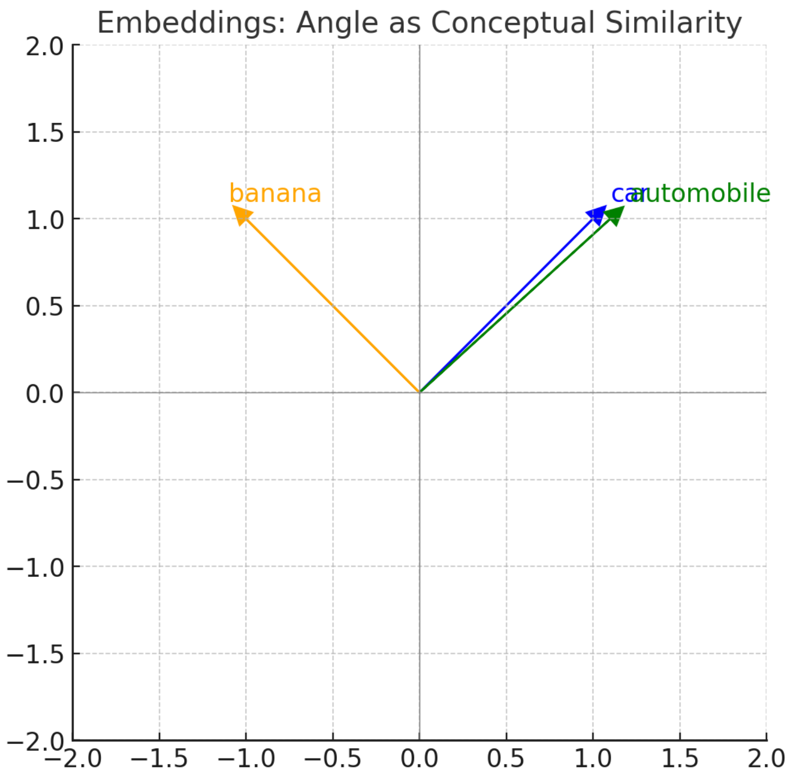
$$||A|| \times ||B|| \approx 3.606 \times 4.123 \approx 14.85$$

$$\cos(\theta) = \frac{5}{14.85} \approx 0.34$$



Cosine Similarity

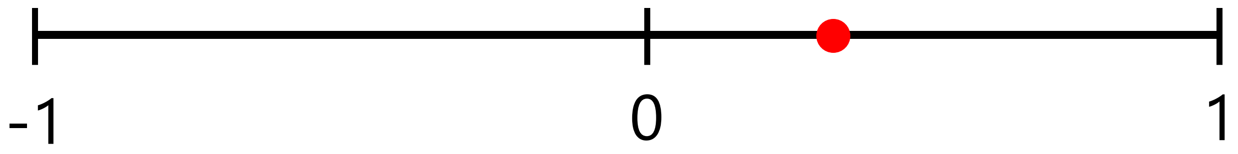
$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \, ||B||}$$



cos(0°) = 1 → perfect match

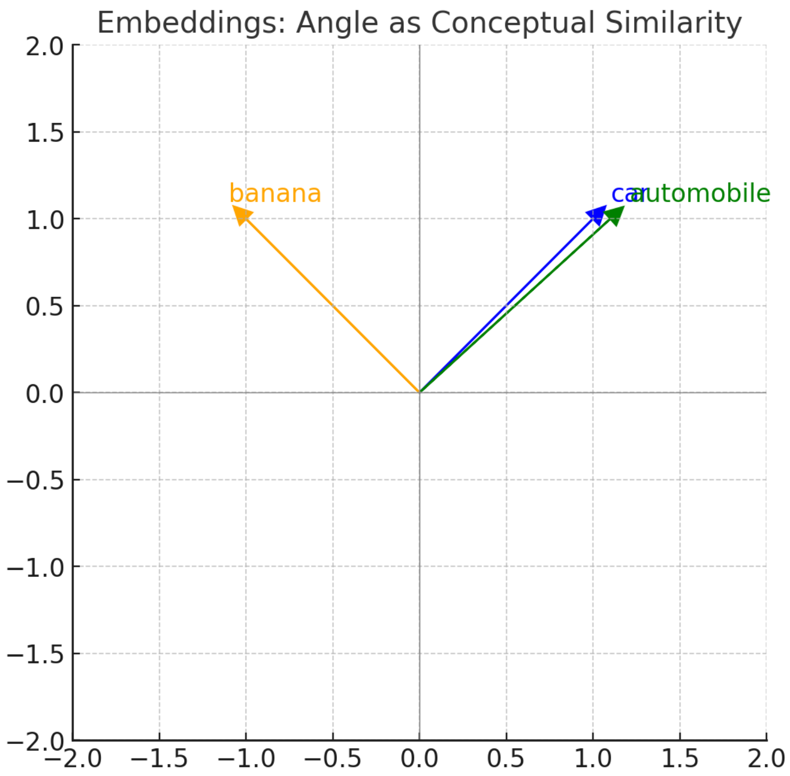
cos(90°) = 0 → no relation

cos(180°) = -1 → opposite



Cosine Similarity

$$\text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{||A||\ ||B||}$$



cos(0°) = 1 → perfect match

cos(90°) = 0 → no relation

cos(180°) = -1 → opposite

"car" → vector A

"automobile" → vector B

Their angle is tiny → high similarity

"car" vs. "banana" → angle ~90° → not related.

Semantic Search Tools in the .NET Ecosystem

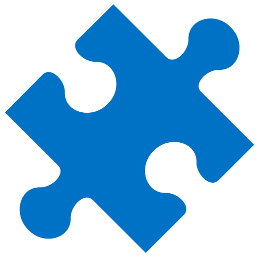
Frameworks & Libraries

Microsoft.Extensions.AI

Semantic Kernel

ML.NET

Frameworks & Libraries



Microsoft.Extensions.AI

Microsoft
Agent Framework

Semantic Kernel

ML.NET

ML.NET

Frameworks & Libraries

Microsoft.Extensions.AI

Microsoft
Agent Framework

Semantic Kernel

ML.NET

Embedding Models

OpenAI API

Ollama

Hugging Face

Embedding Models

OpenAI API

⭐ Ollama

Hugging Face

Vector Databases

Cosmos DB

Redis

Qdrant

Pinecone,
Weaviate, Milvus

NOTE: SQL Server 2025 includes a vector data type

Vector Databases

Cosmos DB

Redis

Qdrant

Pinecone,
Weaviate, Milvus

NOTE: SQL Server 2025 includes a vector data type

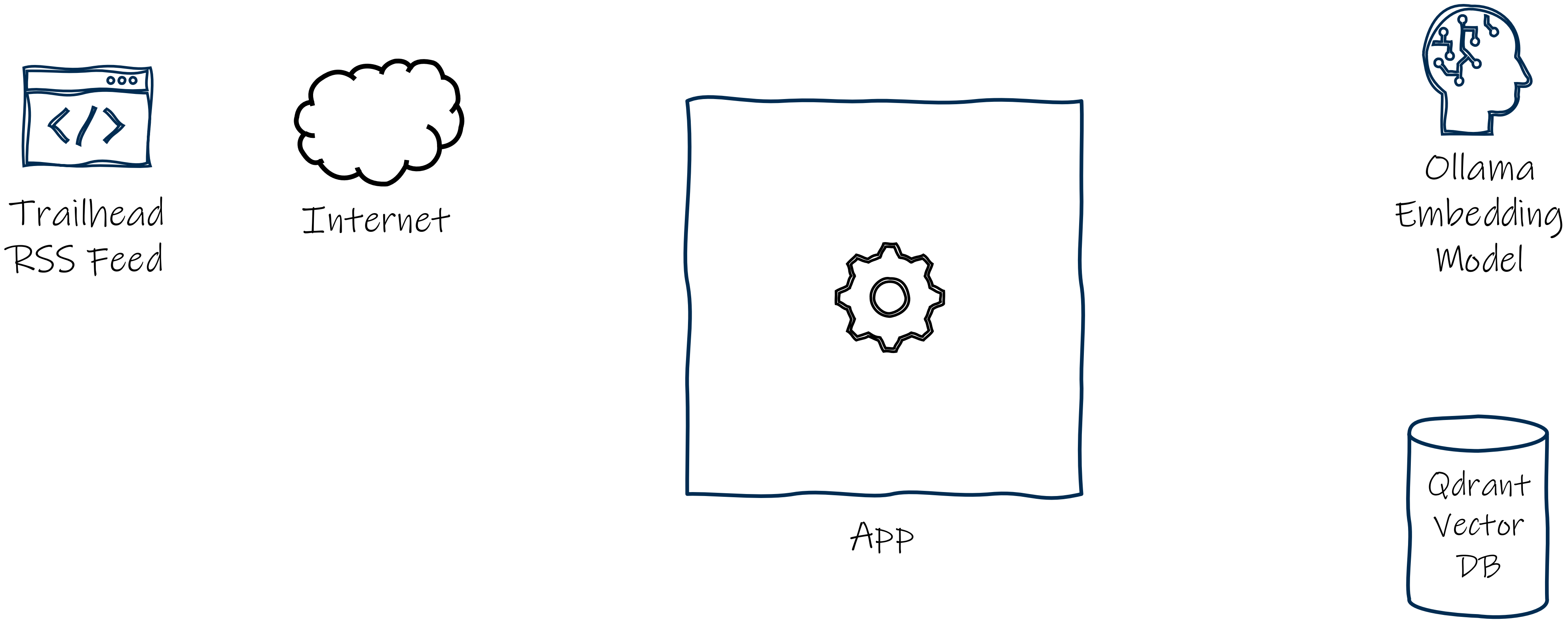
Cloud Services

Azure AI Search

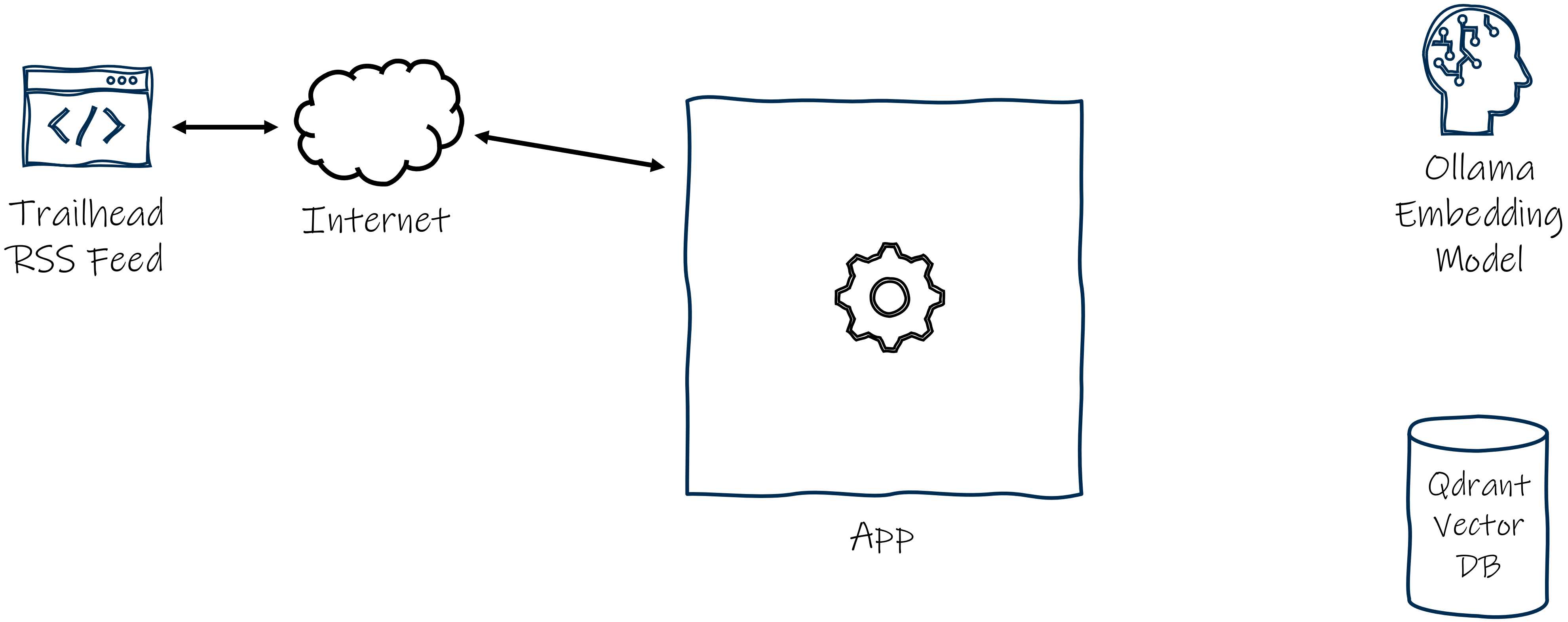
ElasticSearch

Implementing Semantic Search in .NET

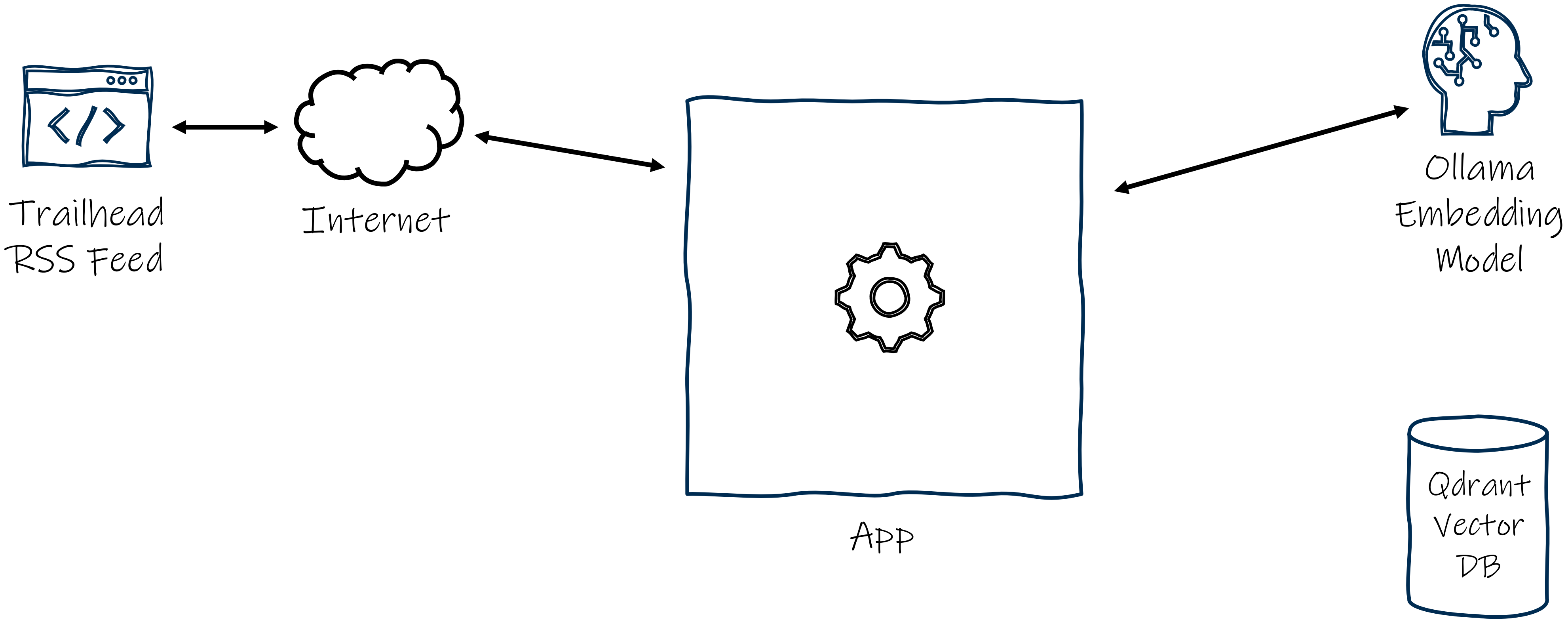
Semantic Search Process



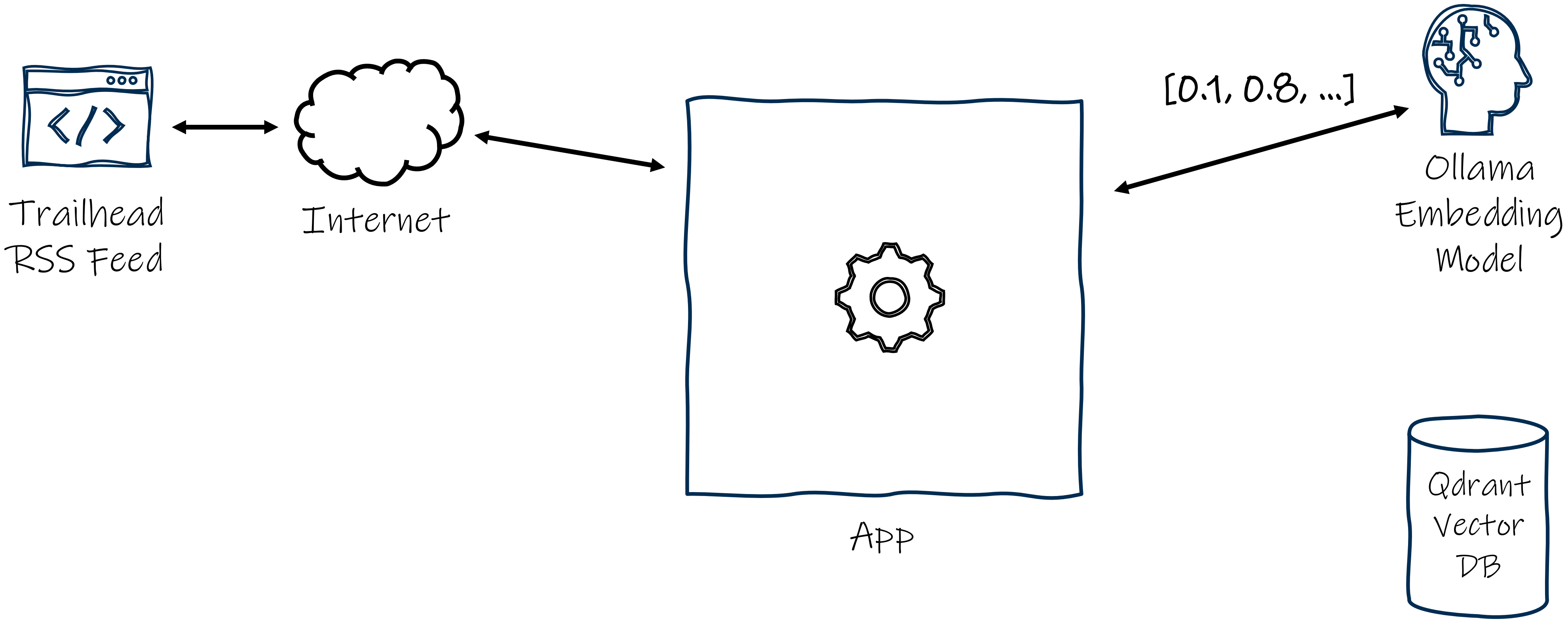
Semantic Search Process



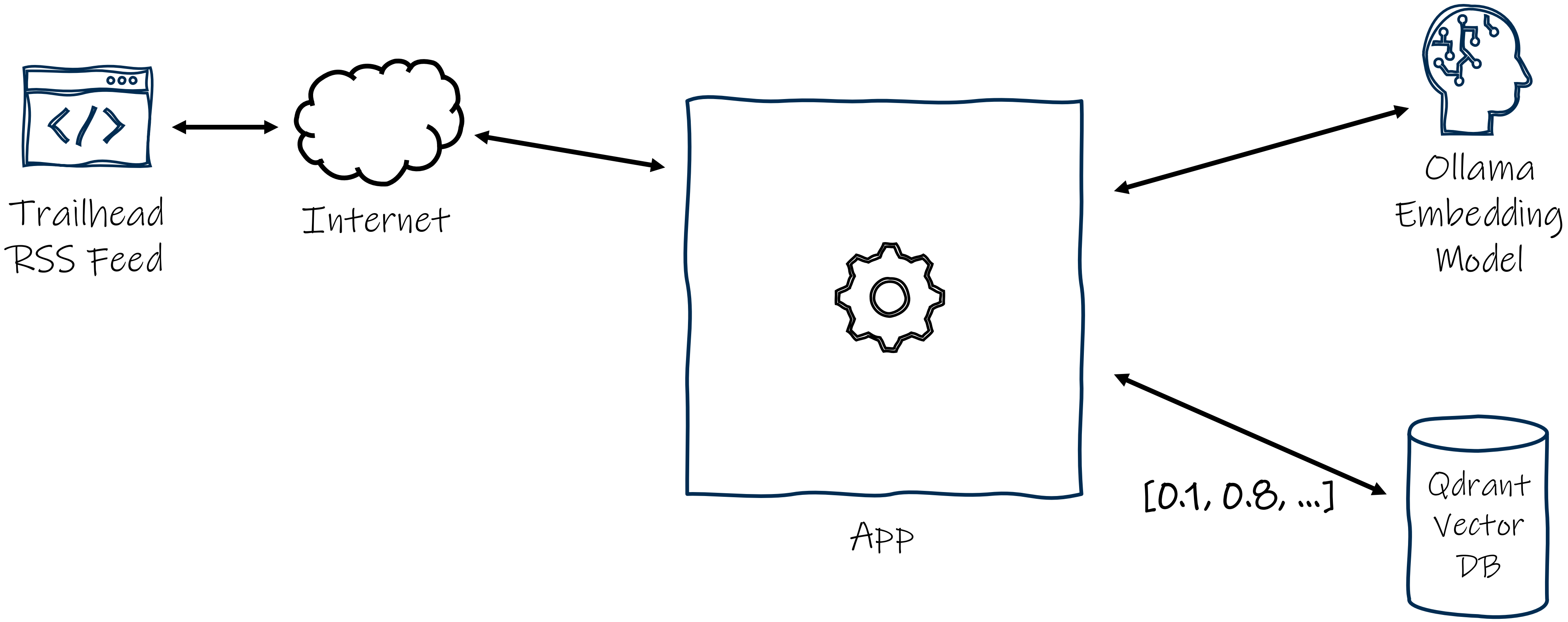
Semantic Search Process



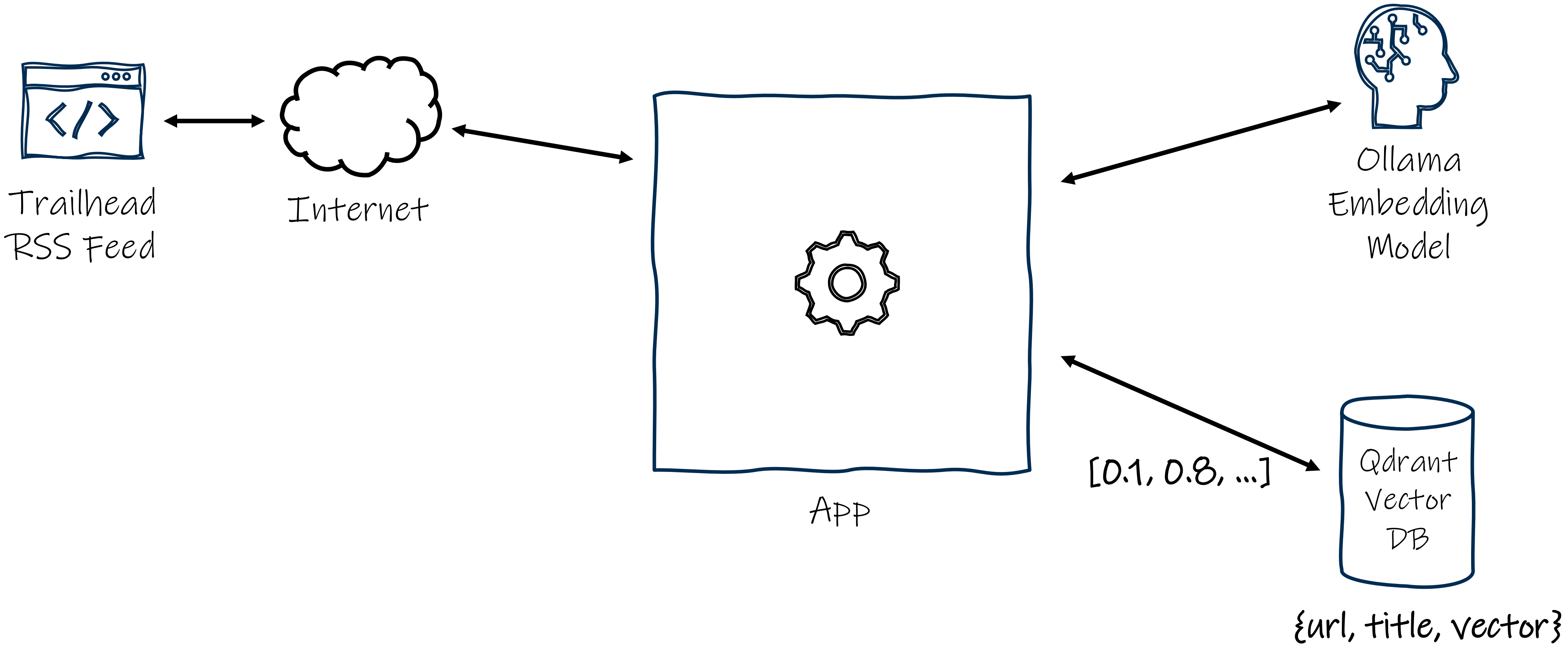
Semantic Search Process



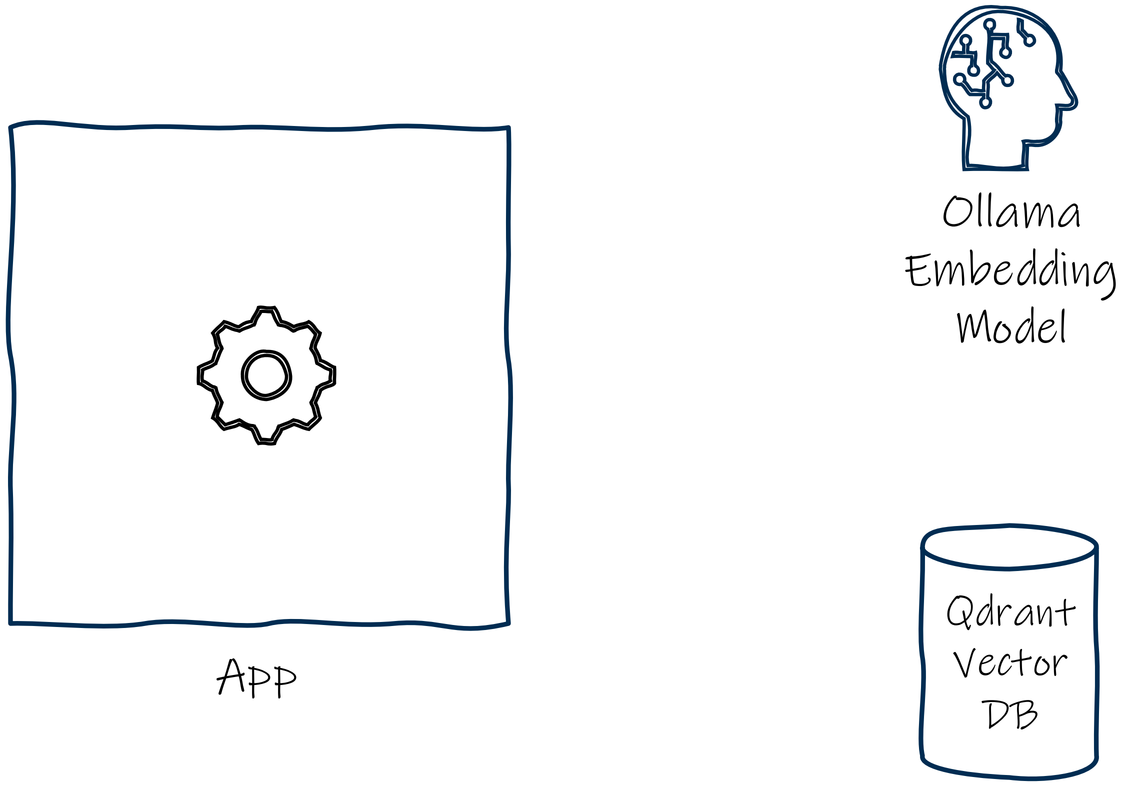
Semantic Search Process



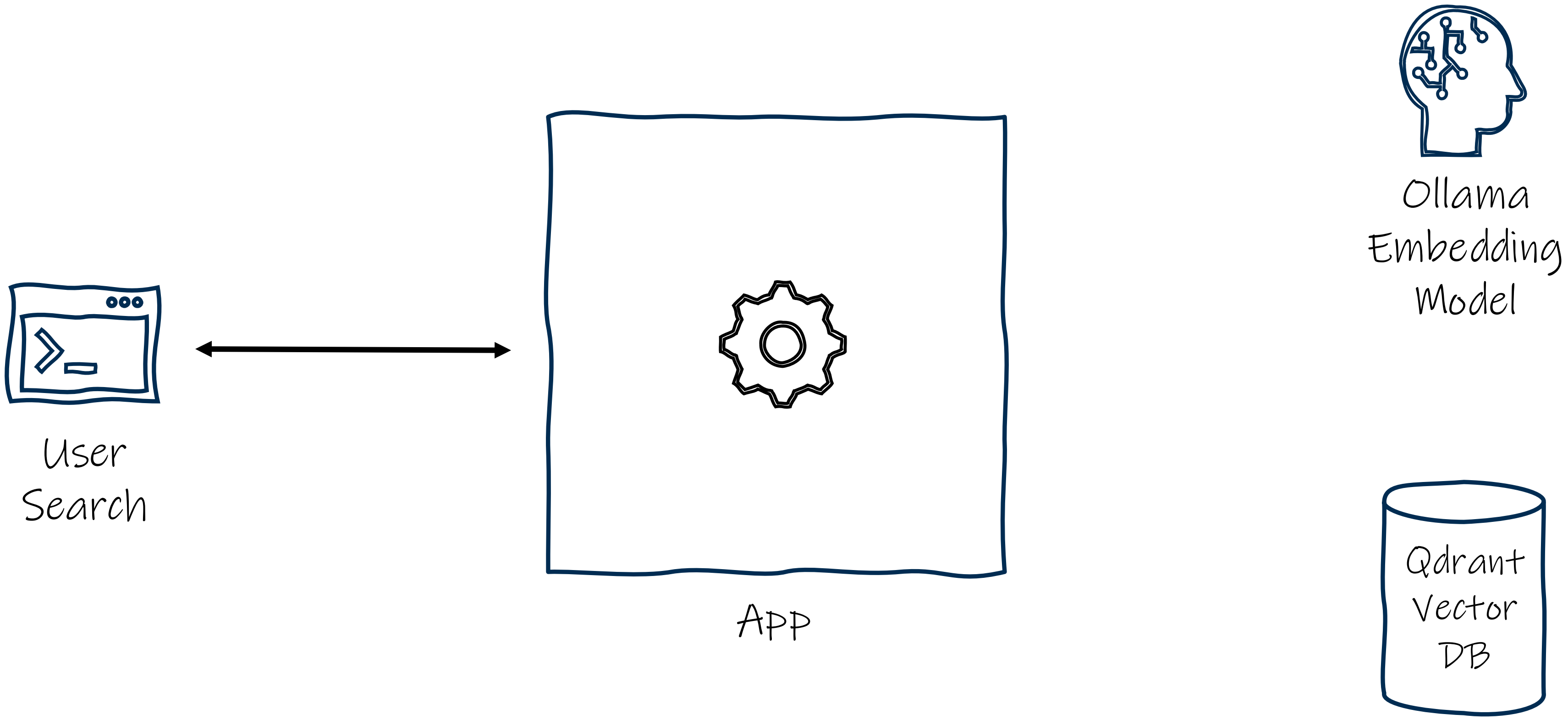
Semantic Search Process



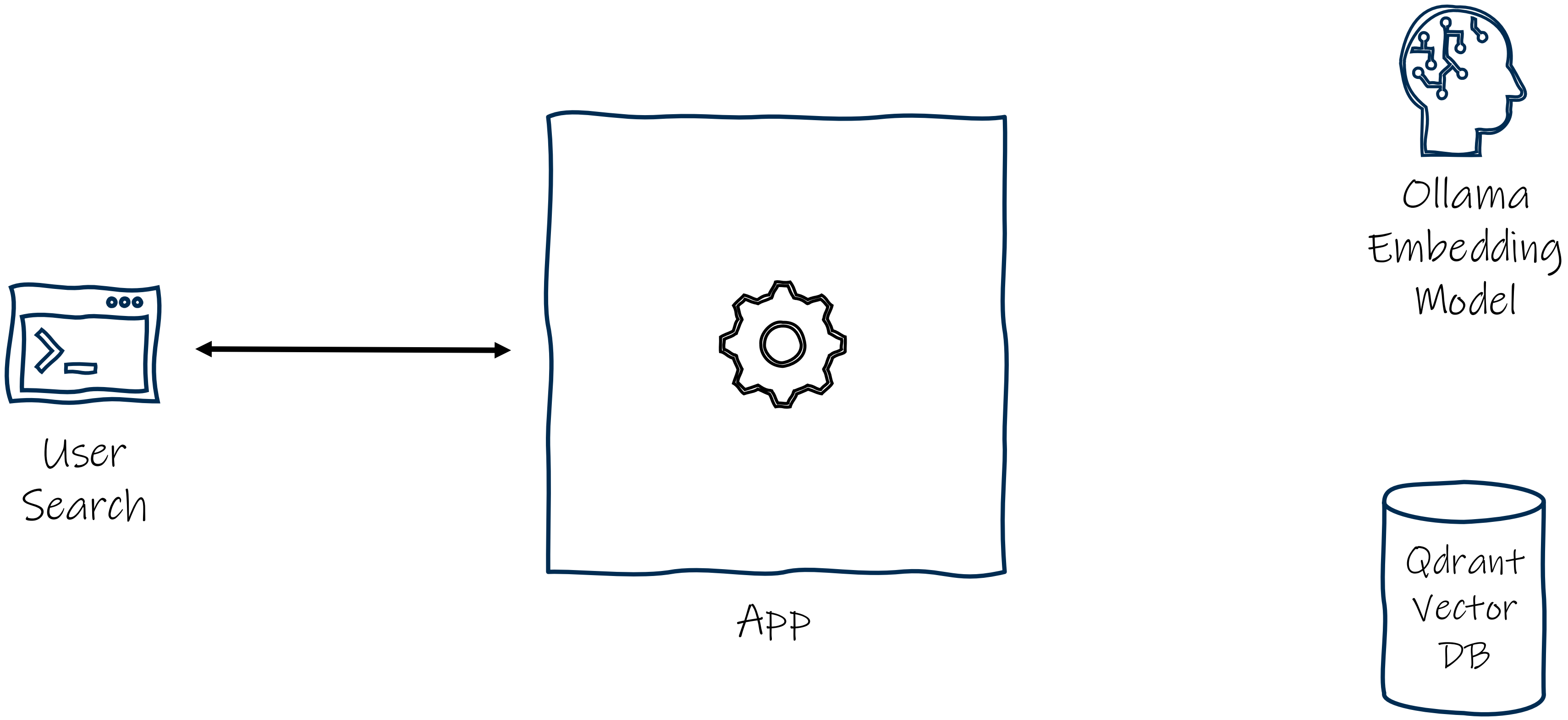
Semantic Search Process



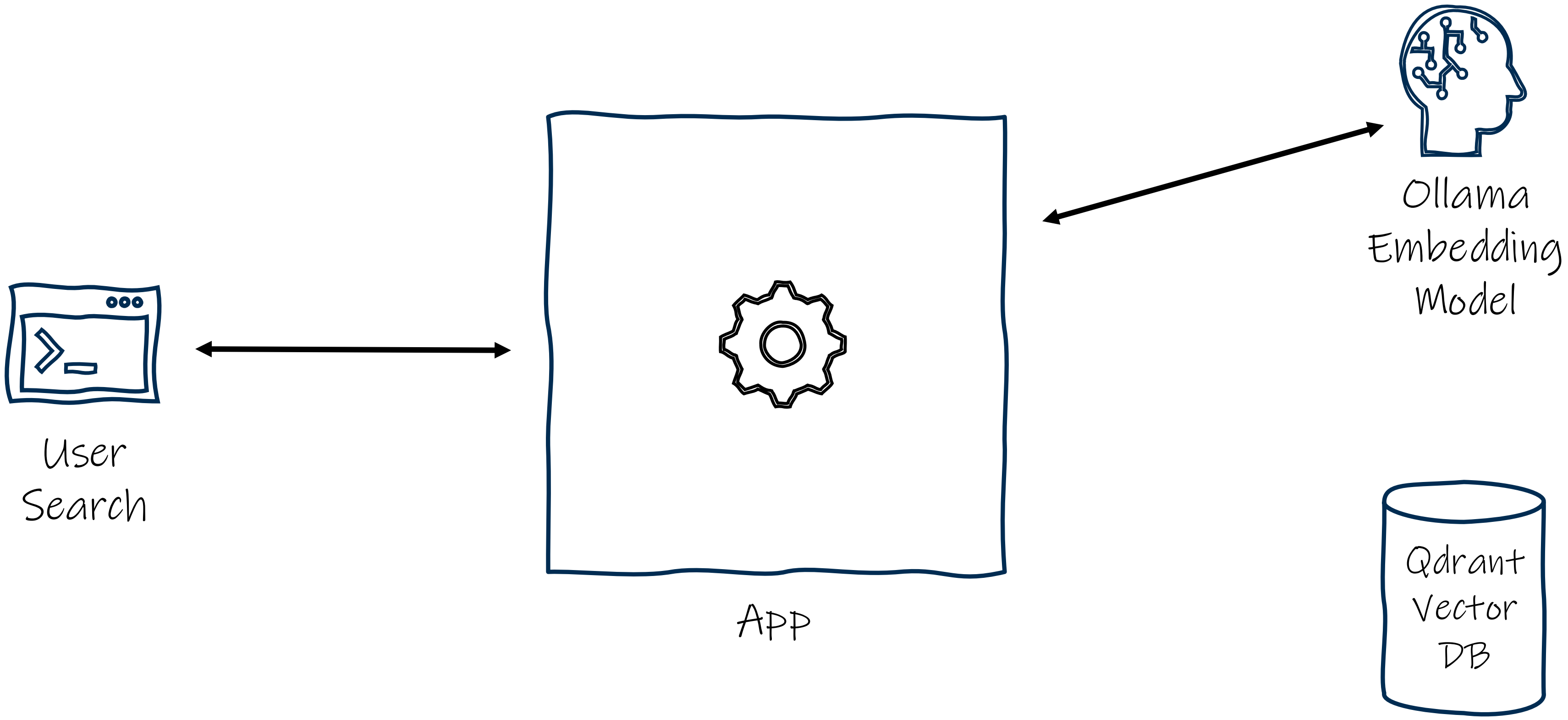
Semantic Search Process



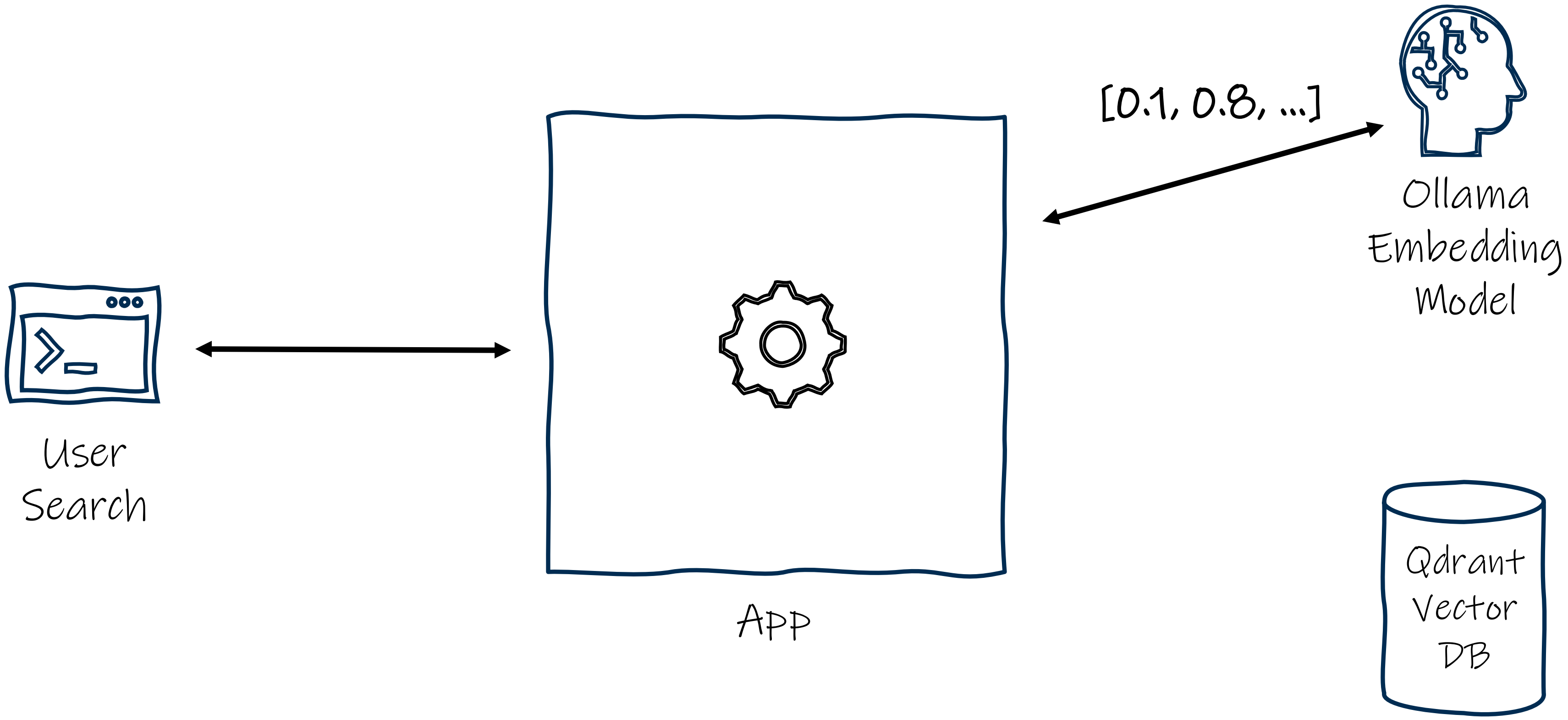
Semantic Search Process



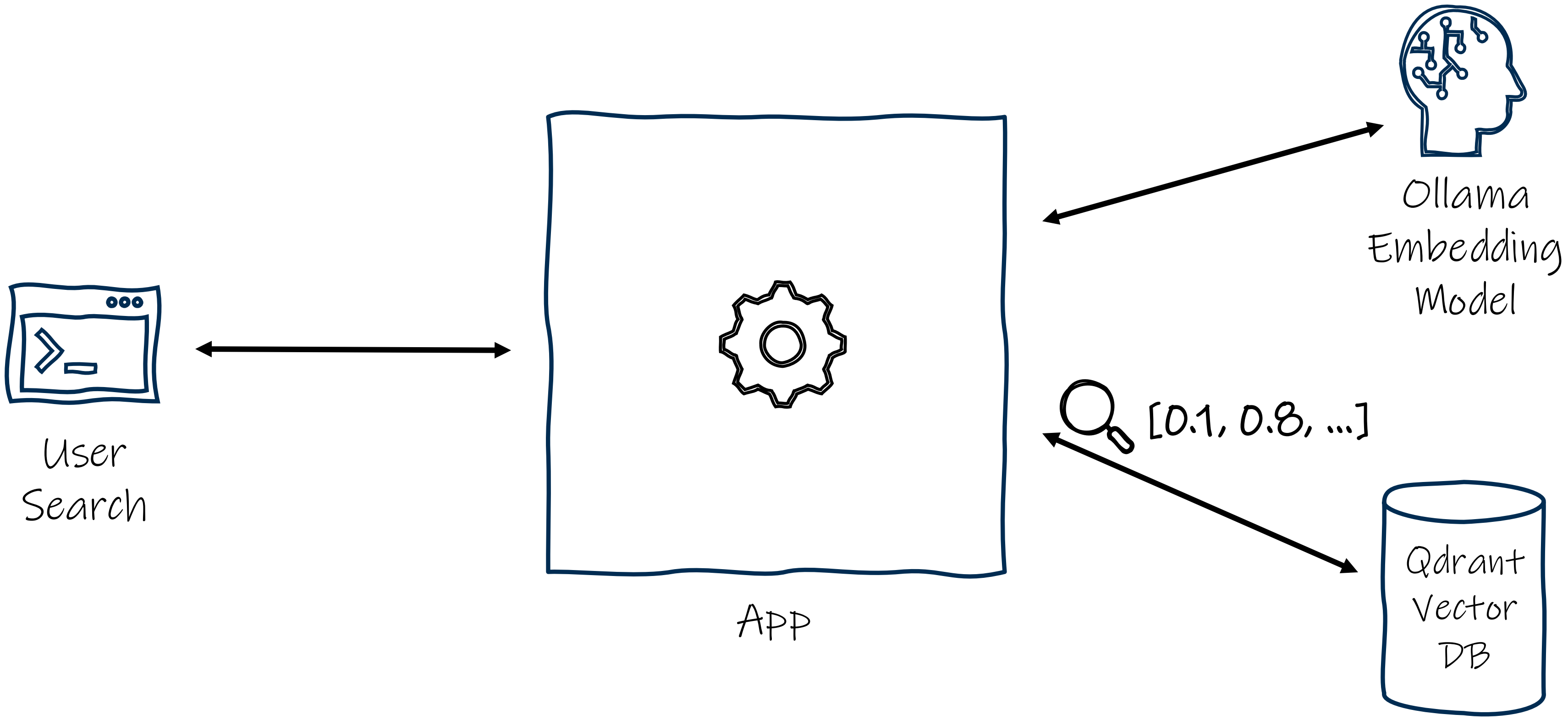
Semantic Search Process



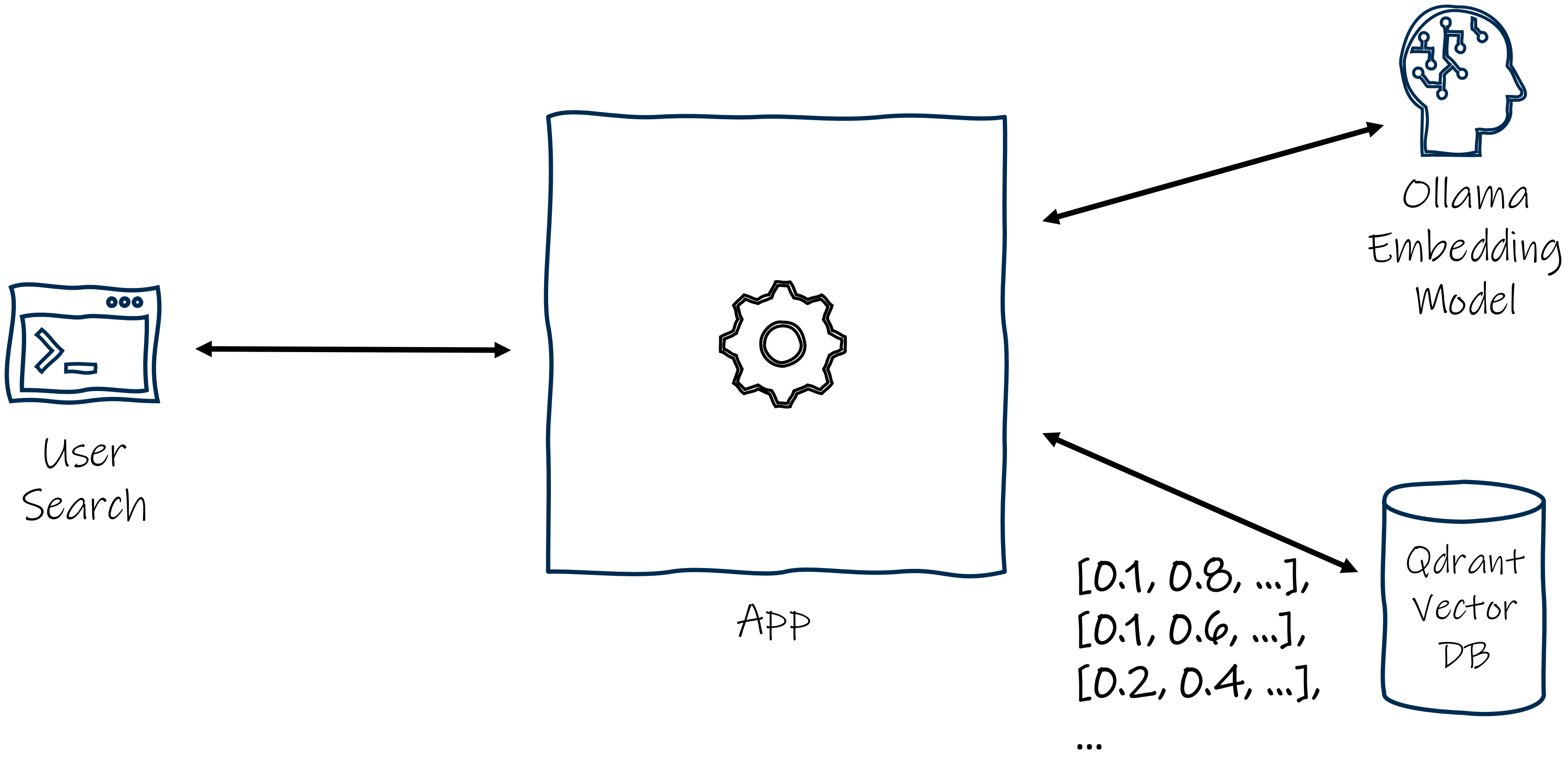
Semantic Search Process



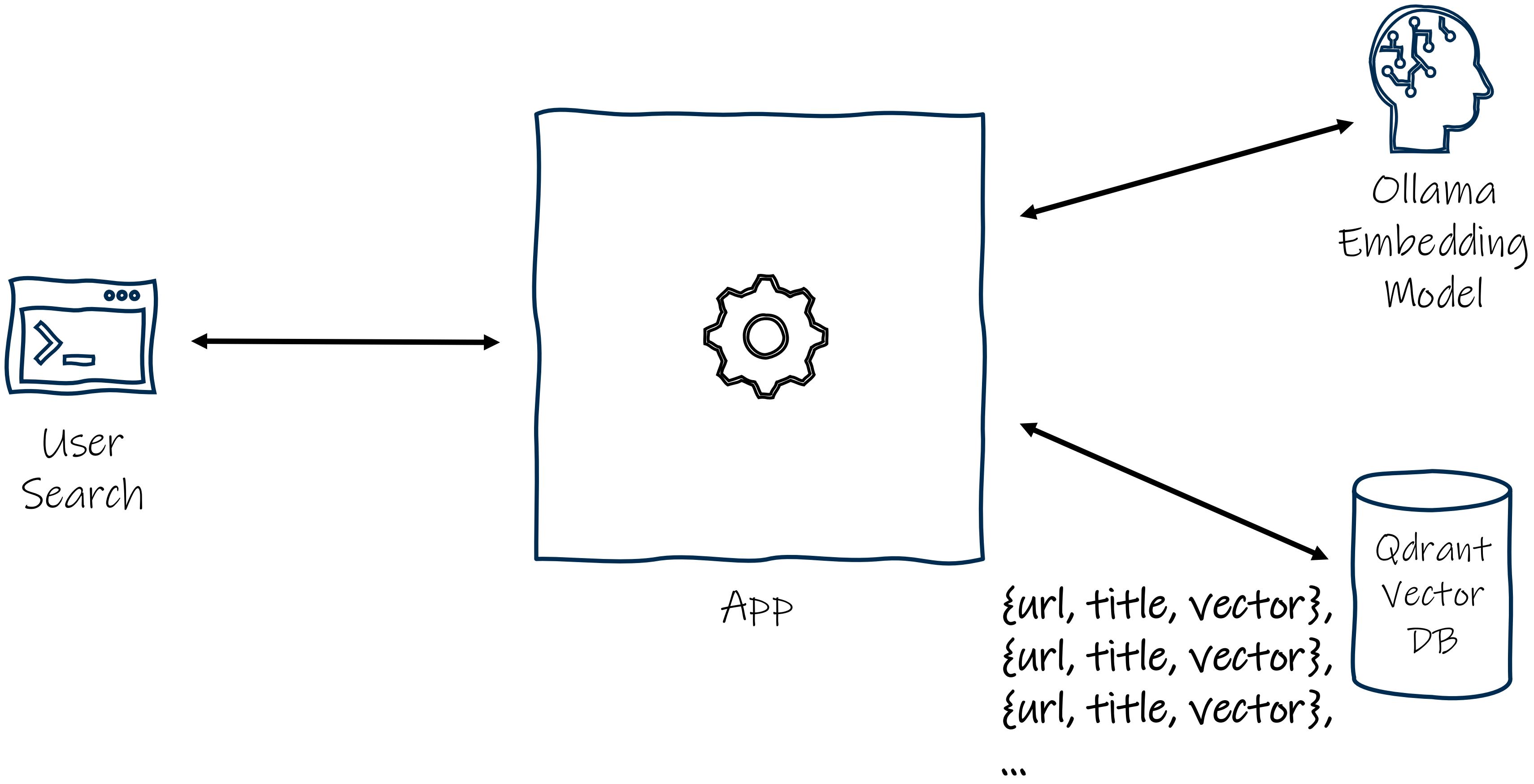
Semantic Search Process



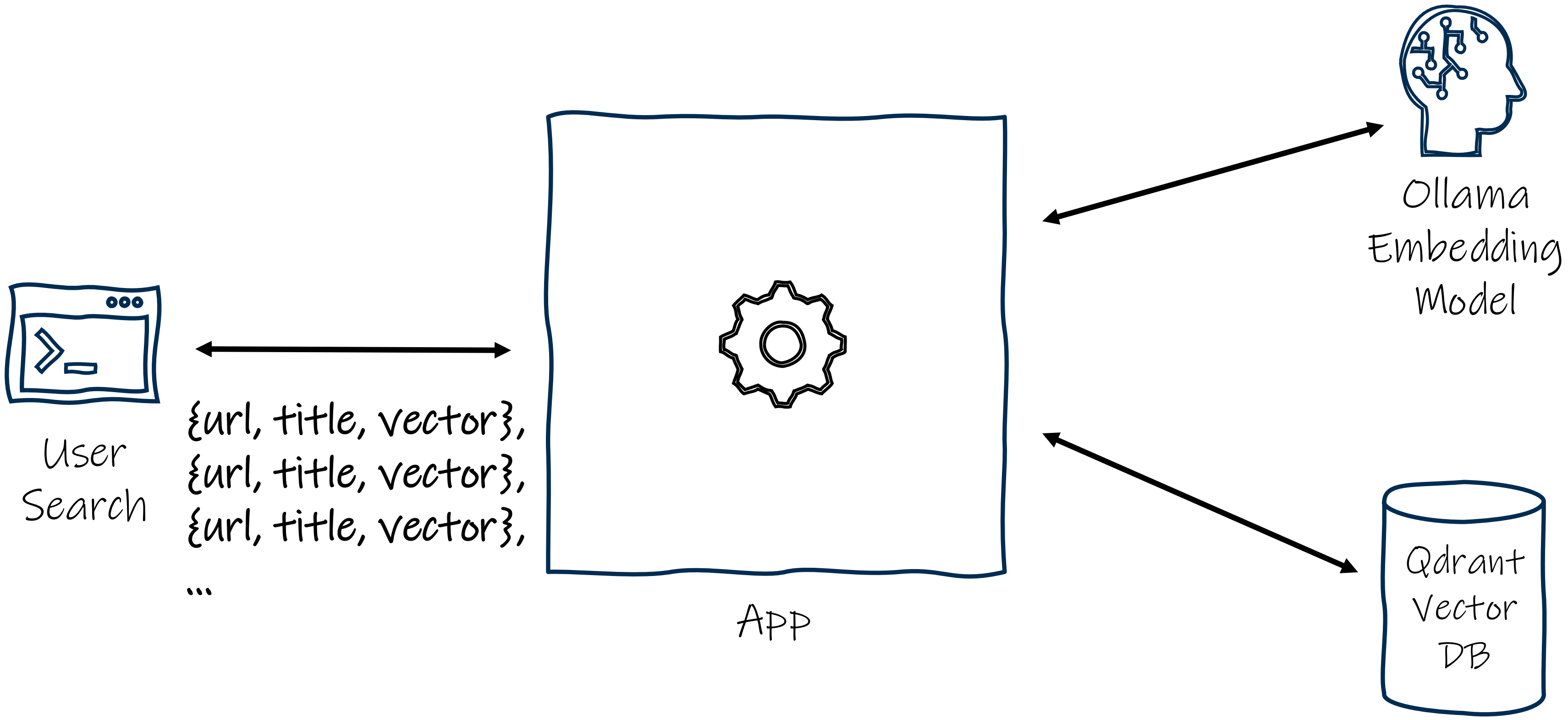
Semantic Search Process



Semantic Search Process



Semantic Search Process



LIVE DEMO

Practical Considerations

Cost & Latency Trade-Offs

Option	Cost	Latency	Accuracy
Local Models	✅ free (typically)	⚠️ medium	⚠️ medium
Cloud Models	⚠️ higher	✅ low	✅ high
Hybrid	⚖️ balanced	⚖️ balanced	⚖️ balanced

Scalability & Storage

Store in a Vector DB

Index Vectors

Scalability & Storage

1 vector = 1 KB

Scalability & Storage

1 KB

1 vector = 1 KB

GBs

1M vectors = GBs

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

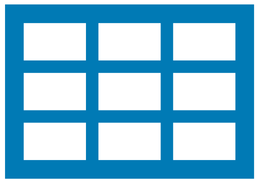
Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

Quality & Model Choice

Type	Pros	Cons	Use Cases
Small embeddings (384–768 dims)	✅ Fast ✅ Cheap ✅ Lower storage	⚠️ Less nuance ⚠️ Lower accuracy	Quick search, lightweight apps, prototyping
Large embeddings (1024–3000 dims)	✅ Higher accuracy ✅ Captures subtle meaning	⚠️ More compute ⚠️ Higher storage/cost	Production search, nuanced queries, RAG
Domain-specific models	✅ Tuned for specific language (legal, medical, finance, etc.) ✅ Often best results	⚠️ May miss general queries ⚠️ Limited availability	Specialized industries, enterprise apps

When NOT to Use Semantic Search

Tiny Datasets



Structured Lookups

Strict Regulatory environments

When NOT to Use Semantic Search

Tiny Datasets

Structured Lookups

Strict Regulatory environments

When NOT to Use Semantic Search

Tiny Datasets

Structured Lookups

Strict Regulatory environments

When NOT to Use Semantic Search

Tiny Datasets

Structured Lookups

Strict Regulatory environments

Summing Up

1. Semantic search **searches meaning**, not just words or parts of words
2. Powered by **vectors** and **embeddings**
3. Many **tools exist** such MEAI, Ollama, Azure OpenAI API, Qdrant, etc.
4. Balance tradeoffs of **local** vs **hosted** models.

Thanks! Questions?

Jonathan "J." Tower

- Microsoft MVP in .NET
- .NET Foundation Board of Directors
- jtower@trailheadtechnology.com
- trailheadtechnology.com/blog
- jtowermi
- Jonathan "J." Tower

EXPERT

Consultation

bit.ly/th-offer

github.com/trailheadtechnology/api-security