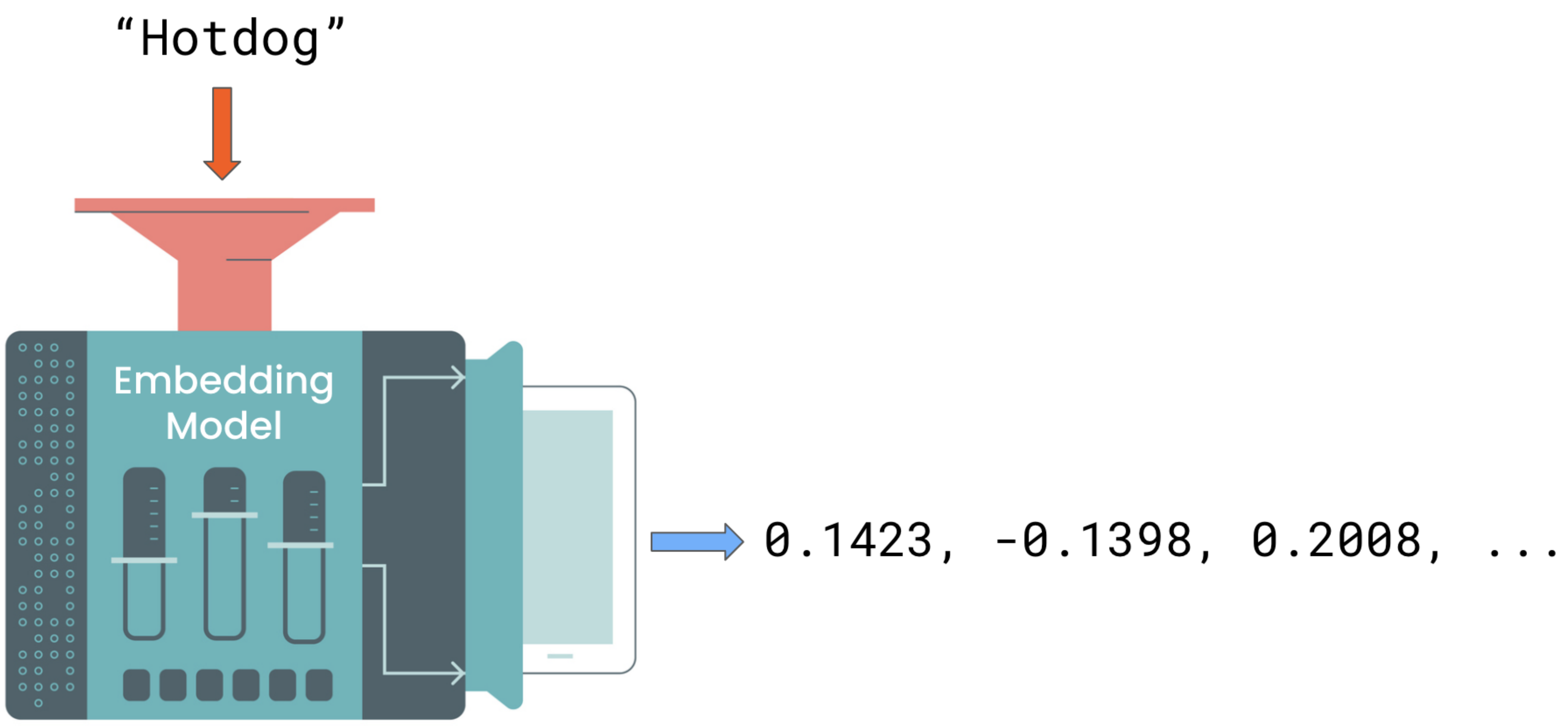
The wonderful world of embeddings!

INTRODUCTION TO EMBEDDINGS WITH THE OPENAI API

Emmanuel Pire
Senior Software Engineer, DataCamp

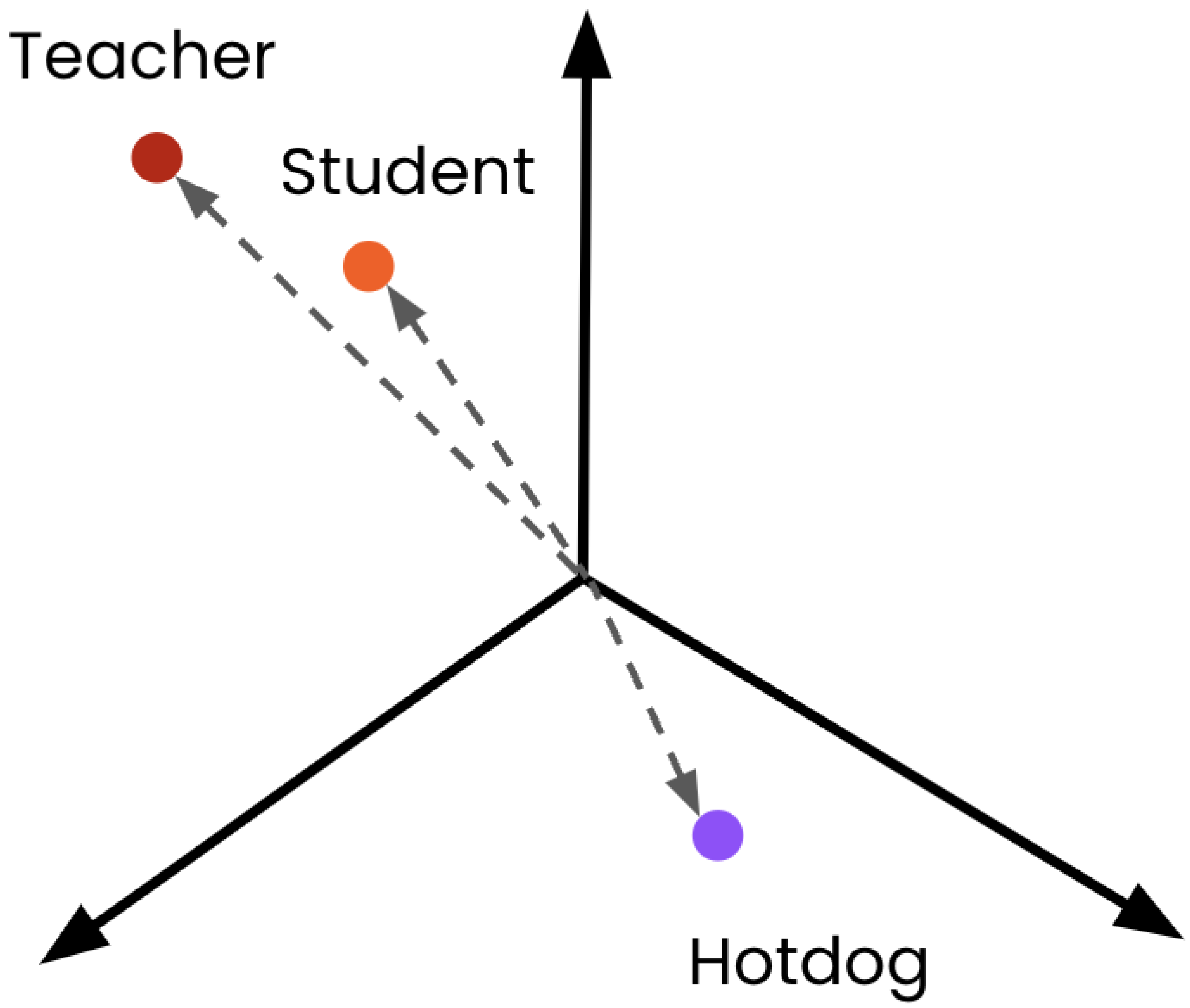
What are embeddings?

- Concept from **N**atural **L**anguage **P**rocessing (**NLP**)
- Numerical representation of text



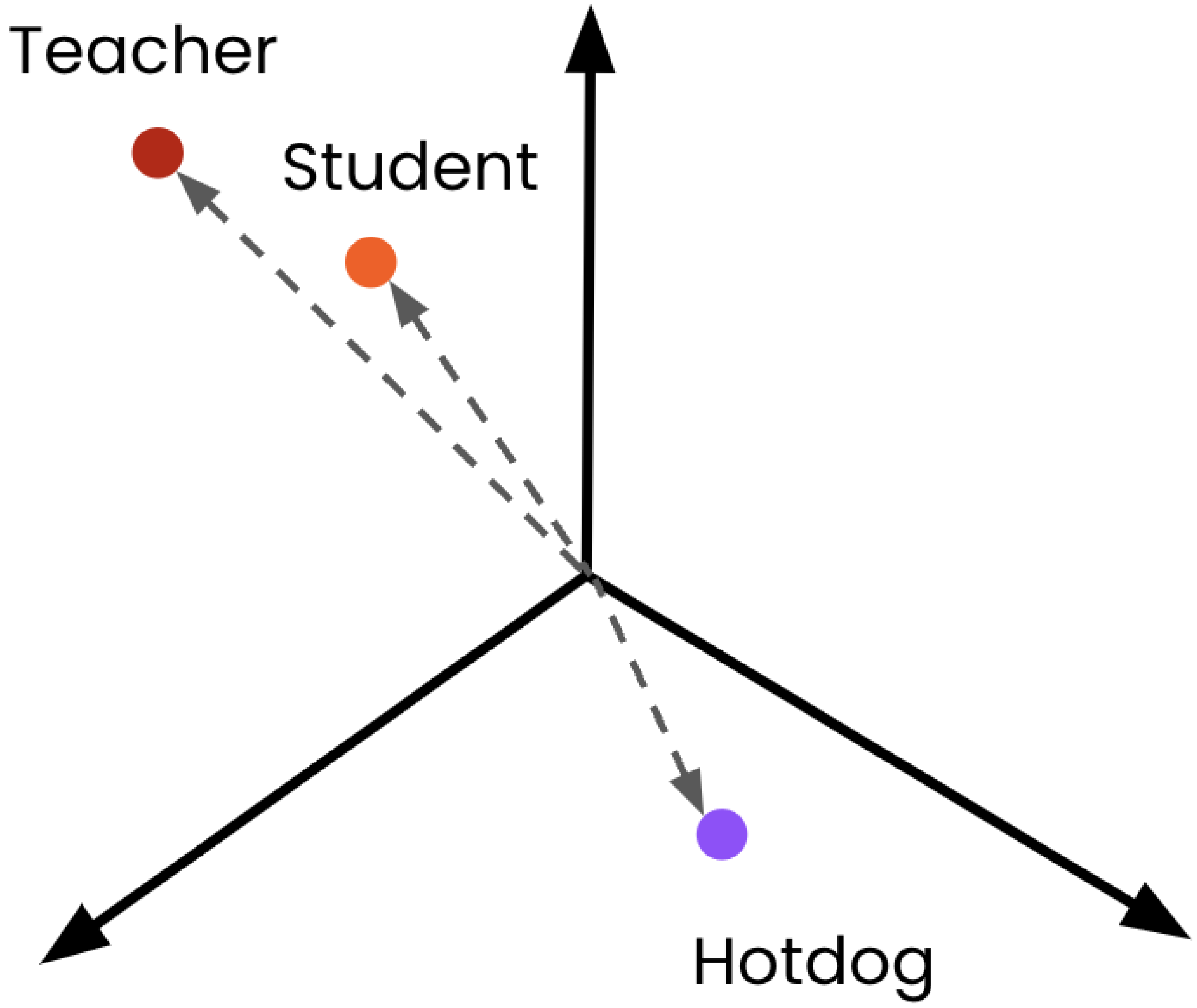
What are embeddings?

- Text is mapped onto a *multi-dimensional* **vector space**
- The numbers outputted by the model are the text's location in the space
- Similar words appear *closer together*
- Dissimilar words appear *further away*



Why are embeddings useful?

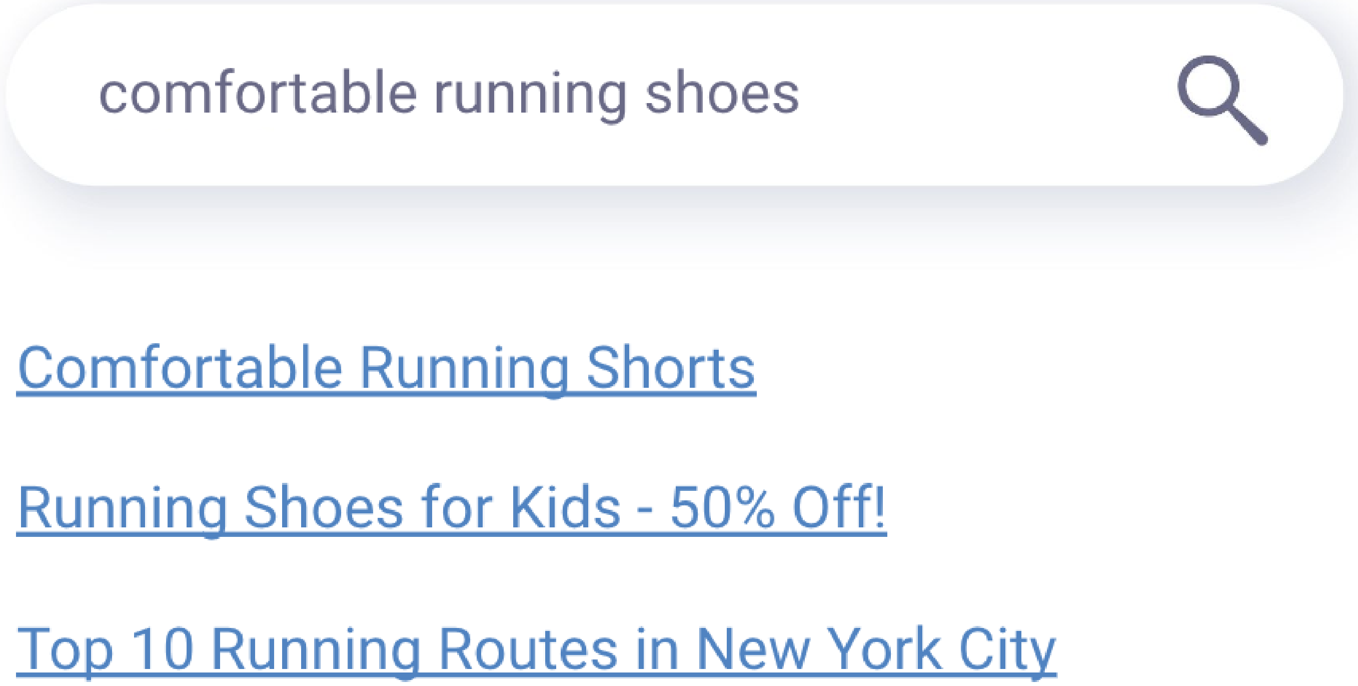
- Embeddings allow *semantic meaning* to be captured
- **Semantic meaning:** context and intent behind text



- **Example:**
  - "Which way is it to the supermarket?"
  - "Could I have directions to the shop?"

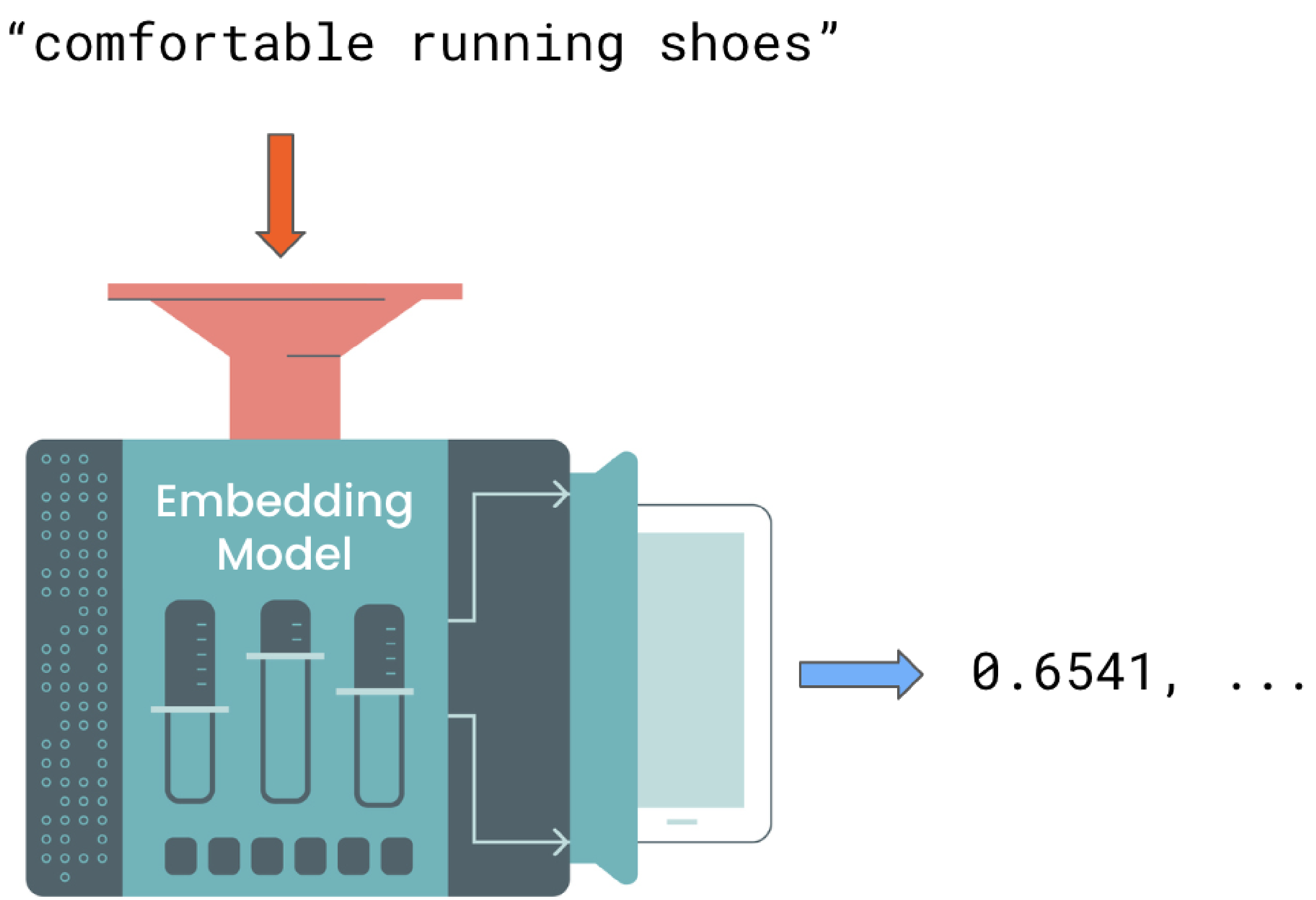
Semantic search engines

- Traditional search engines
  - Use **keyword** pattern matching
  - May miss the true intent
  - Will miss word variations



Semantic search engines

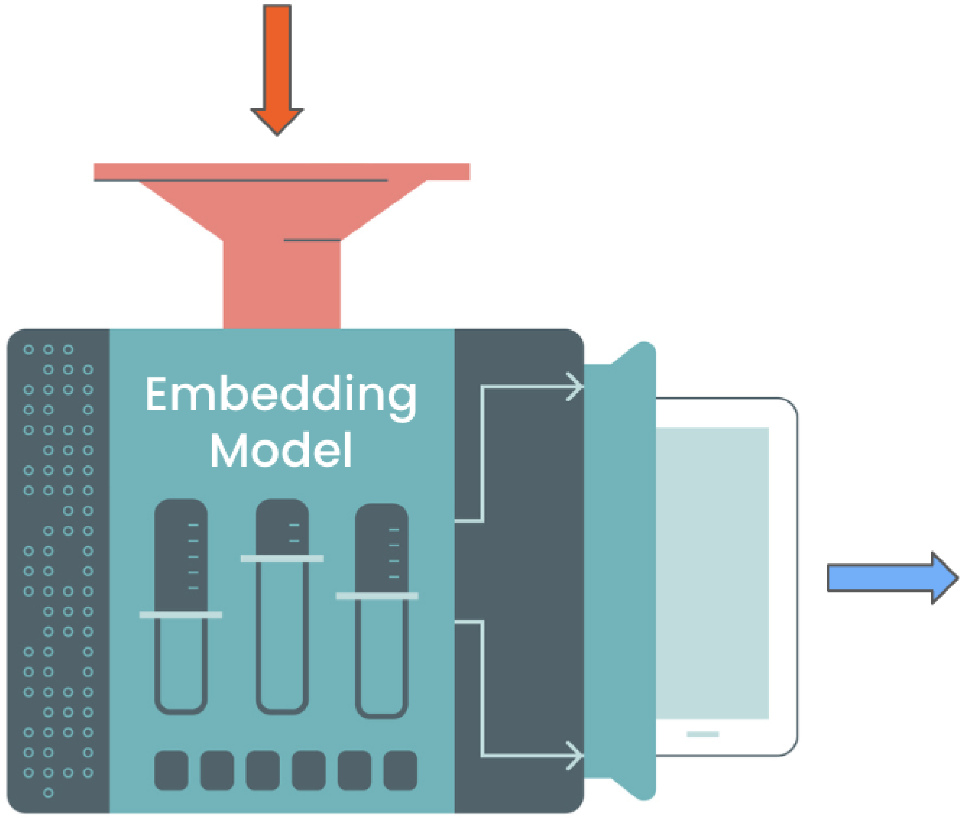
- Use **embeddings** to understand intent and context



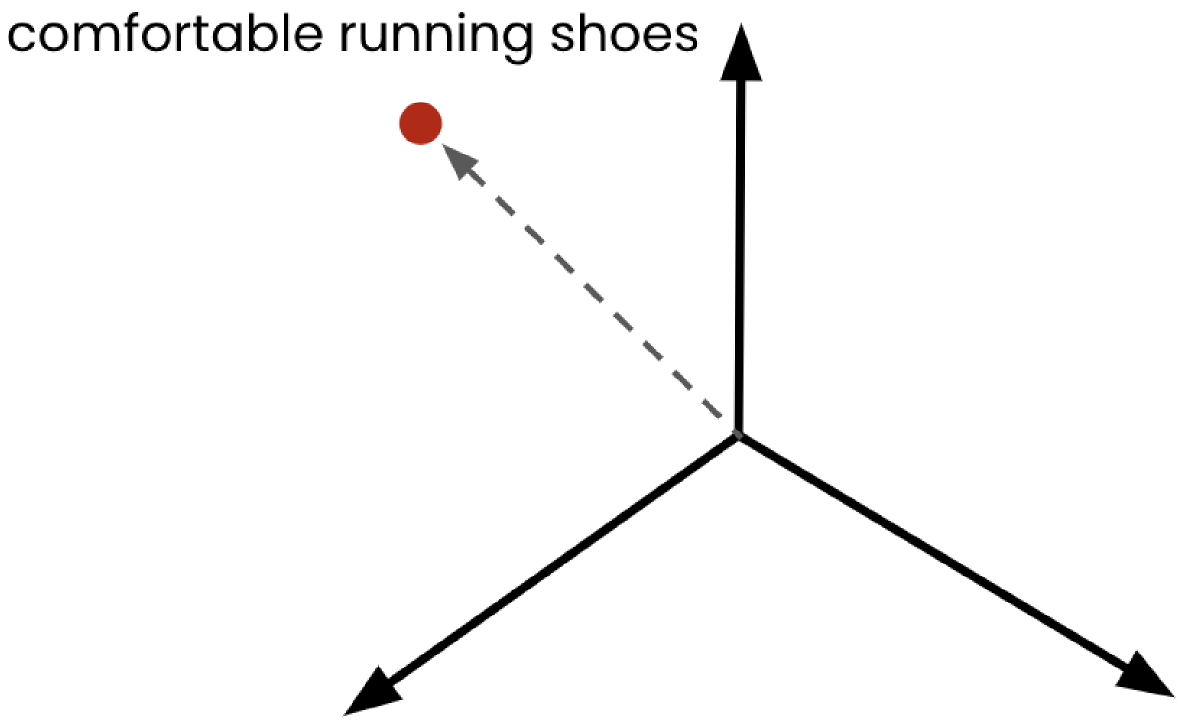
Semantic search engines

- Use **embeddings** to understand intent and context

“comfortable running shoes”

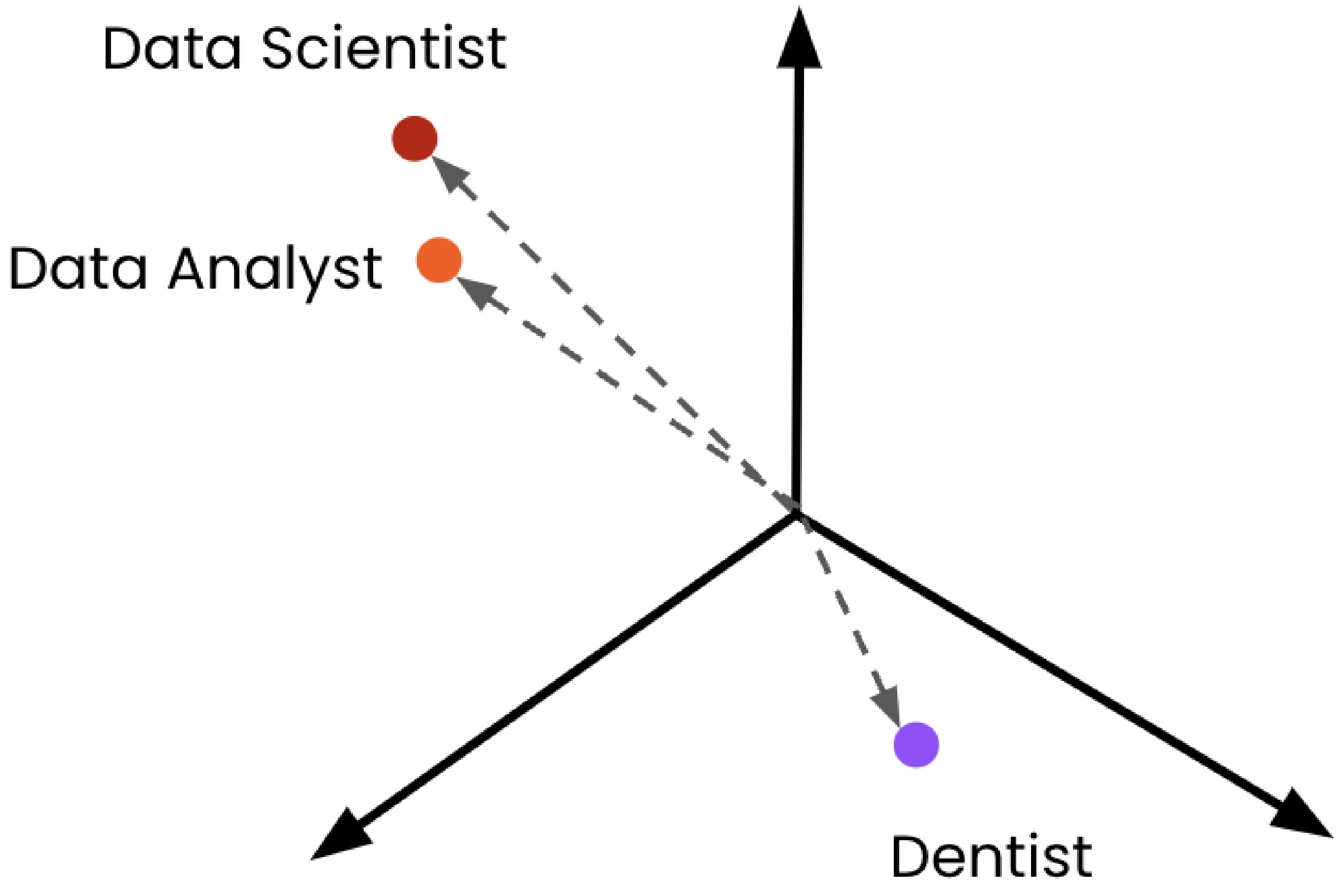


0.6541, ...



Recommendation systems

- **Example:** Job post recommendations
  - Recommend jobs based on descriptions already viewed
  - Mitigates variations in job title



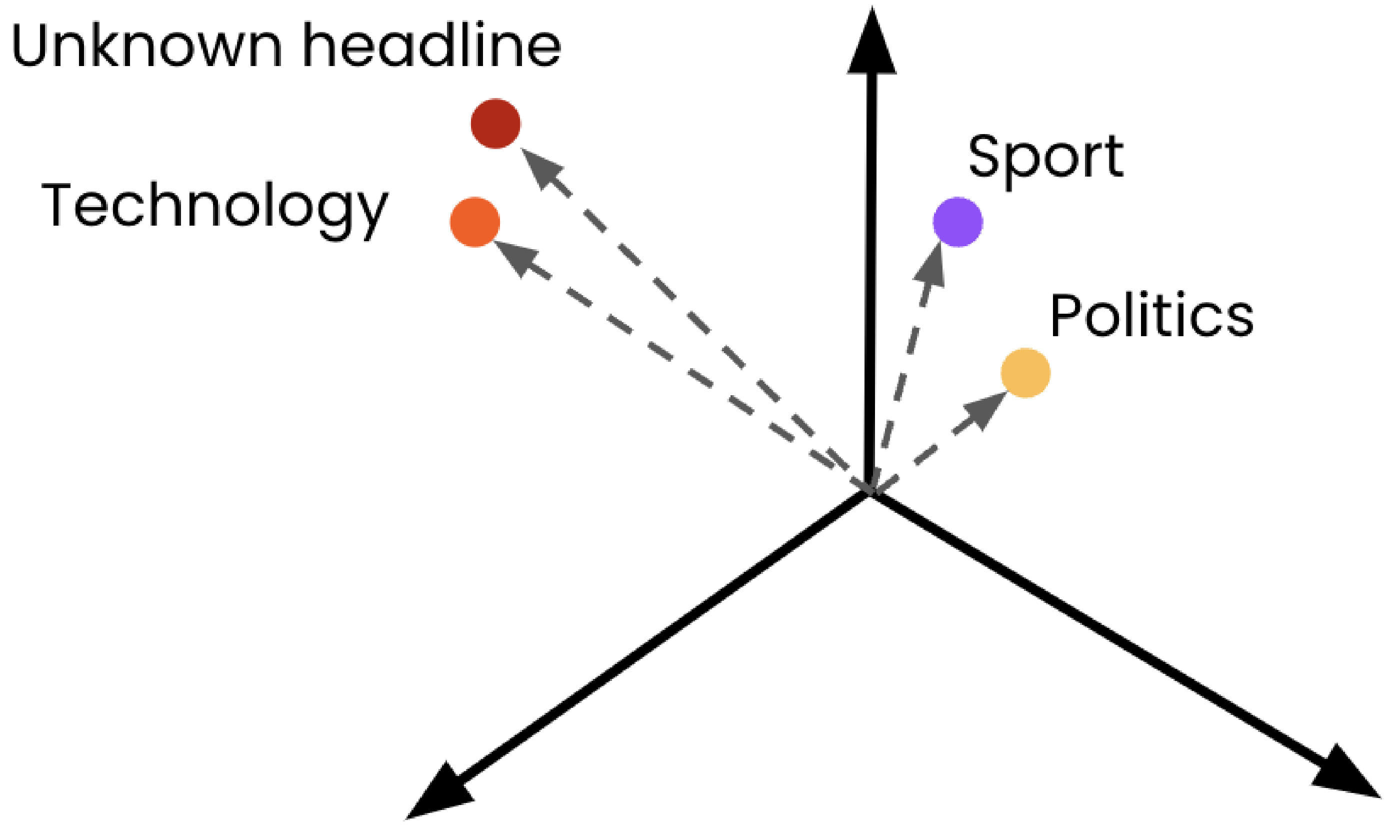
Classification

Classification tasks:

- Classify sentiment
- Cluster observations
- Categorization

Example:

- Classifying news headlines



Creating an Embeddings request

- Embeddings endpoint

```
from openai import OpenAI

client = OpenAI(api_key="<OPENAI_API_KEY>")
response = client.embeddings.create(
  model="text-embedding-3-small",
  input="Embeddings are a numerical representation of text that can be used to
measure the relatedness between two pieces of text."
)

response_dict = response.model_dump()
print(response_dict)
```

[1] https://platform.openai.com/docs/api-reference/embeddings

Embeddings response

```
{'object': 'list',
 'data': [
    {
      "embedding": [0.0023064255, ..., -0.0028842222],
      "index": 0,
      "object": "embedding"
    }
  ],
 'model': 'text-embedding-3-small',
 'usage': {
  "prompt_tokens": 24,
  "total_tokens": 24
  }
}
```

Extracting the embeddings

```
print(response_dict['data'][0]['embedding'])
```

```
[0.0023064255, ...., -0.0028842222]
```

Let's practice!

INTRODUCTION TO EMBEDDINGS WITH THE OPENAI API

Investigating the vector space

INTRODUCTION TO EMBEDDINGS WITH THE OPENAI API

Emmanuel Pire
Senior Software Engineer, DataCamp

Example: Embedding headlines

```
articles = [
    {"headline": "Economic Growth Continues Amid Global Uncertainty", "topic": "Business"},
    {"headline": "Interest rates fall to historic lows", "topic": "Business"},
    {"headline": "Scientists Make Breakthrough Discovery in Renewable Energy", "topic": "Science"},
    {"headline": "India Successfully Lands Near Moon's South Pole", "topic": "Science"},
    {"headline": "New Particle Discovered at CERN", "topic": "Science"},
    {"headline": "Tech Company Launches Innovative Product to Improve Online Accessibility", "topic": "Tech"},
    {"headline": "Tech Giant Buys 49% Stake In AI Startup", "topic": "Tech"},
    {"headline": "New Social Media Platform Has Everyone Talking!", "topic": "Tech"},
    {"headline": "The Blues get promoted on the final day of the season!", "topic": "Sport"},
    {"headline": "1.5 Billion Tune-in to the World Cup Final", "topic": "Sport"}
]
```

Example: Embedding headlines

```
[
  {
      'headline': 'The Blues get promoted on the final day!',
      'topic': 'Sport',
      'embedding': [0.0015748793, ..., -0.0052598542]
  },
  {
      'headline': 'Interest rates fall to historic levels',
      'topic': 'Business',
      'embedding': [-0.030351446, ..., -0.0044289114]
  },
  ...
]
```

Embedding multiple inputs

```
headline_text = [article['headline'] for article in articles]
headline_text
```

```
["Economic Growth Continues Amid Global Uncertainty",
 ...,
 "1.5 Billion Tune-in to the World Cup Final"]
```

```
response = client.embeddings.create(
  model="text-embedding-3-small",
  input=headline_text
)
response_dict = response.model_dump()
```

- **Batching** is more efficient than using multiple API calls

```
[...]

'data': [
    {
      "embedding": [-0.017142612487077713, ..., -0.0012911480152979493],
      "index": 0,
      "object": "embedding"
    },
    {
      "embedding": [-0.032995883375406265, ..., -0.0028605300467461348],
      "index": 1,
      "object": "embedding"
    },
    ...
  ]

[...]
```

Embedding multiple inputs

```
articles = [
    {"headline": "Economic Growth Continues Amid Global Uncertainty", "topic": "Business"},
     ...
]
```

```
for i, article in enumerate(articles):
    article['embedding'] = response_dict['data'][i]['embedding']

print(articles[:2])
```

```
[{'headline': 'Economic Growth Continues Amid Global Uncertainty',
  'topic': 'Business',
  'embedding': [-0.017142612487077713, ..., -0.0012911480152979493]}
 {'headline': 'Interest rates fall to historic lows',
  'topic': 'Business',
  'embedding': [-0.03295883375406265, ..., -0.0028605300467461348]}]
```

How long is the embeddings vector?

- "Economic Growth Continues Amid Global Uncertainty"

```
len(articles[0]['embedding'])
```

```
1536
```

- "Tech Company Launches Innovative Product to Improve Accessibility"

```
len(articles[5]['embedding'])
```

```
1536
```

- **Always returns 1536 numbers!**

Dimensionality reduction and t-SNE

- Various techniques to *reduce* the number of dimensions
- **t-SNE** (**t**-distributed **S**tochastic **N**eighbor **E**mbedding)

[1] https://www.datacamp.com/tutorial/introduction-t-sne

Implementing t-SNE

```
from sklearn.manifold import TSNE
import numpy as np

embeddings = [article['embedding'] for article in articles]

tsne = TSNE(n_components=2, perplexity=5)
embeddings_2d = tsne.fit_transform(np.array(embeddings))
```

- `n_components` : the resulting number of dimensions
- `perplexity` : used by the algorithm, must be *less than number of data points*
- Will result in **information loss**

[1] https://www.datacamp.com/tutorial/introduction-t-sne

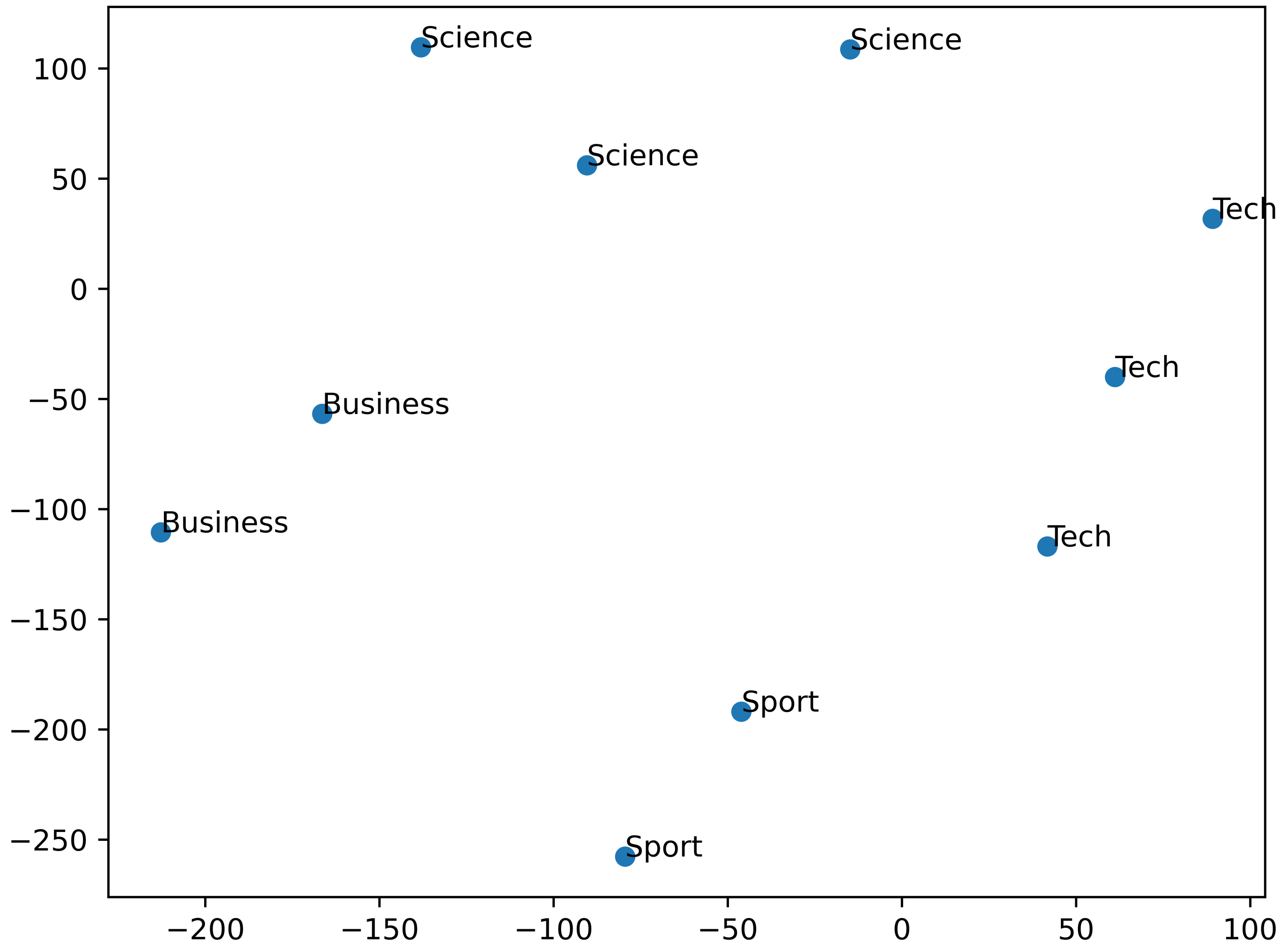
Visualizing the embeddings

```
import matplotlib.pyplot as plt

plt.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1])

topics = [article['topic'] for article in articles]
for i, topic in enumerate(topics):
    plt.annotate(topic, (embeddings_2d[i, 0], embeddings_2d[i, 1]))

plt.show()
```

Visualizing the embeddings

- *Similar* articles are grouped together!
- Model captured the *semantic* meaning

- **Coming up:** Computing similarity



Let's practice!

INTRODUCTION TO EMBEDDINGS WITH THE OPENAI API

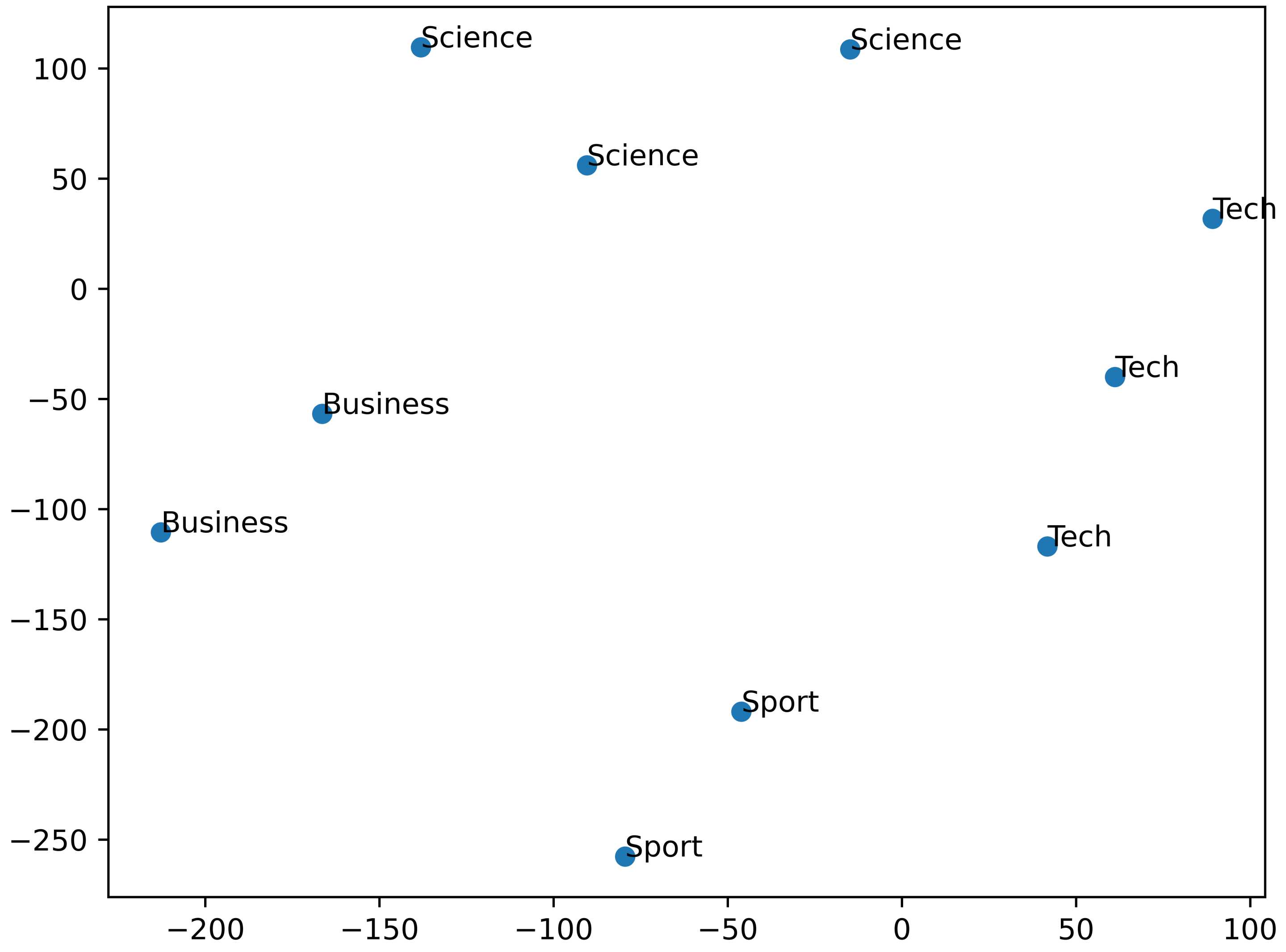
Text similarity

INTRODUCTION TO EMBEDDINGS WITH THE OPENAI API

Emmanuel Pire
Senior Software Engineer, DataCamp

Recap...

- Semantically similar texts are embedded *more closely* in the **vector space**
- Measuring distance allows us to measure similarity
- Enables embeddings applications:
  - Semantic search
  - Recommendations
  - Classification



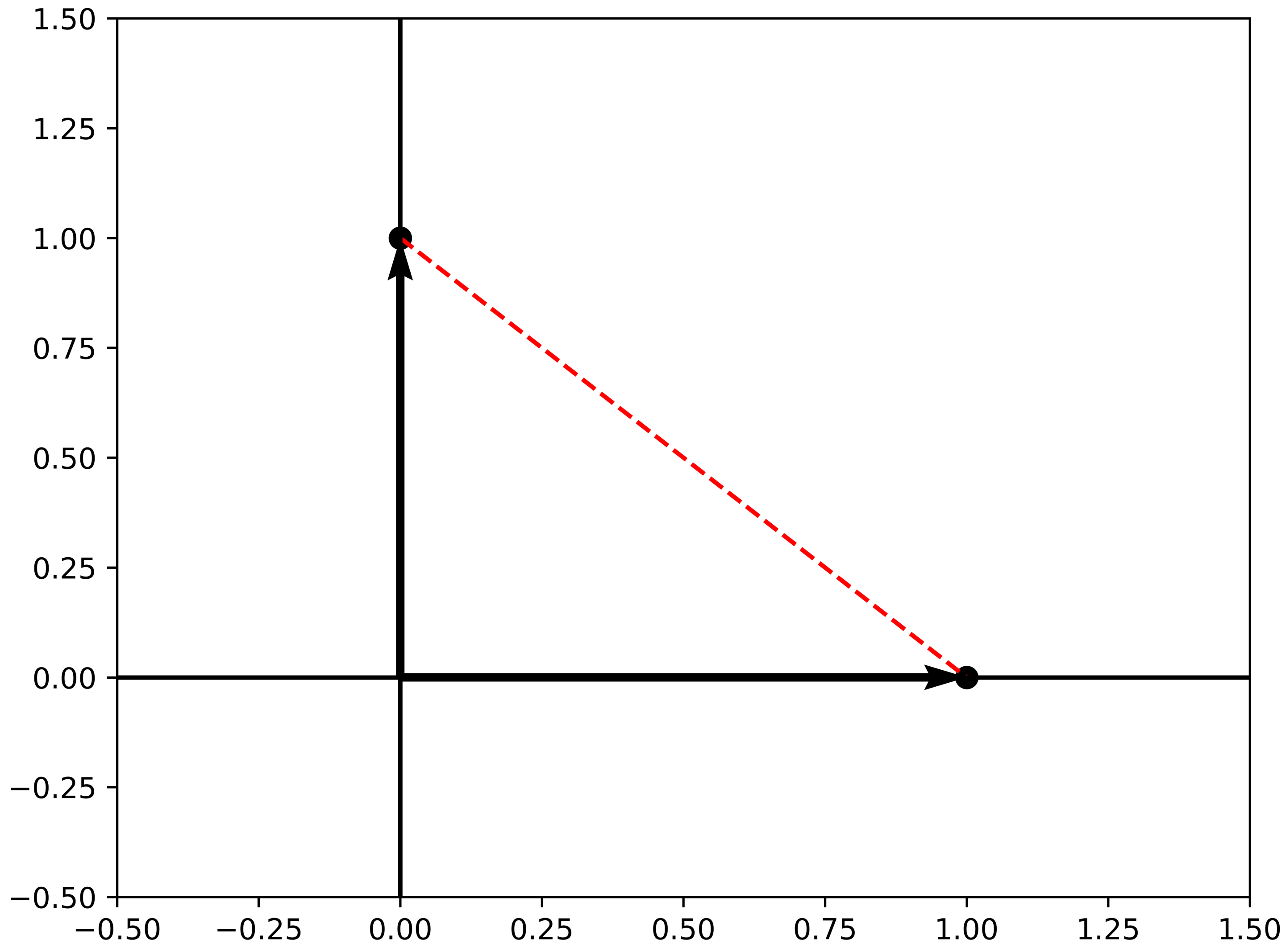
Measuring similarity

Cosine distance

```
from scipy.spatial import distance

distance.cosine([0, 1], [1, 0])
```

```
1.0
```

- Ranges from 0 to 2
- Smaller numbers = *Greater* similarity



Example: Comparing headline similarity

```
[
  {
      'headline': 'The Blues get promoted on the final day!',
      'topic': 'Sport',
      'embedding': [0.0015748793, ..., -0.0052598542]
  },
  {
      'headline': 'Interest rates fall to historic levels',
      'topic': 'Business',
      'embedding': [-0.030351446, ..., -0.0044289114]
  },
  ...
]
```

Example: Comparing headline similarity

```python
def create_embeddings(texts):
  response = client.embeddings.create(
    model="text-embedding-3-small",
    input=texts
  )
  response_dict = response.model_dump()

  return [data['embedding'] for data in response_dict['data']]
```

```python
print(create_embeddings(["Python is the best!", "R is the best!"]))
print(create_embeddings("DataCamp is awesome!")[0])
```

```
[[0.0050565884448587894, ..., , -0.04000323638319969],
 [-0.0018890155479311943, ..., -0.04085670784115791]]
[0.0003701007517520338З, ..., -0.021759100258350372]
```

Example: Comparing headline similarity

```
from scipy.spatial import distance
import numpy as np

search_text = "computer"
search_embedding = create_embeddings(search_text)[0]

distances = []
for article in articles:
  dist = distance.cosine(search_embedding, article["embedding"])
  distances.append(dist)

min_dist_ind = np.argmin(distances)
print(articles[min_dist_ind]['headline'])
```

```
Tech Company Launches Innovative Product to Improve Online Accessibility
```

Let's practice!

INTRODUCTION TO EMBEDDINGS WITH THE OPENAI API