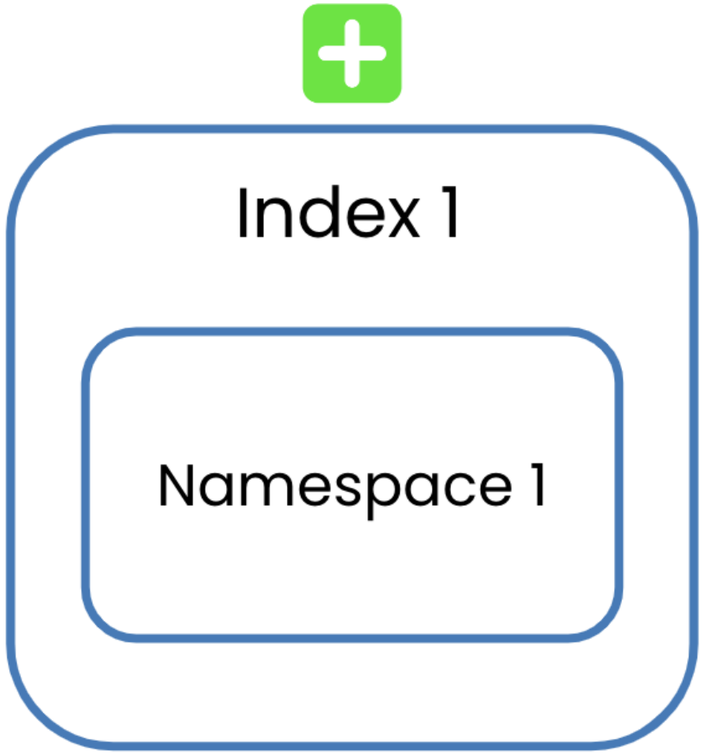
Retrieving vectors

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE

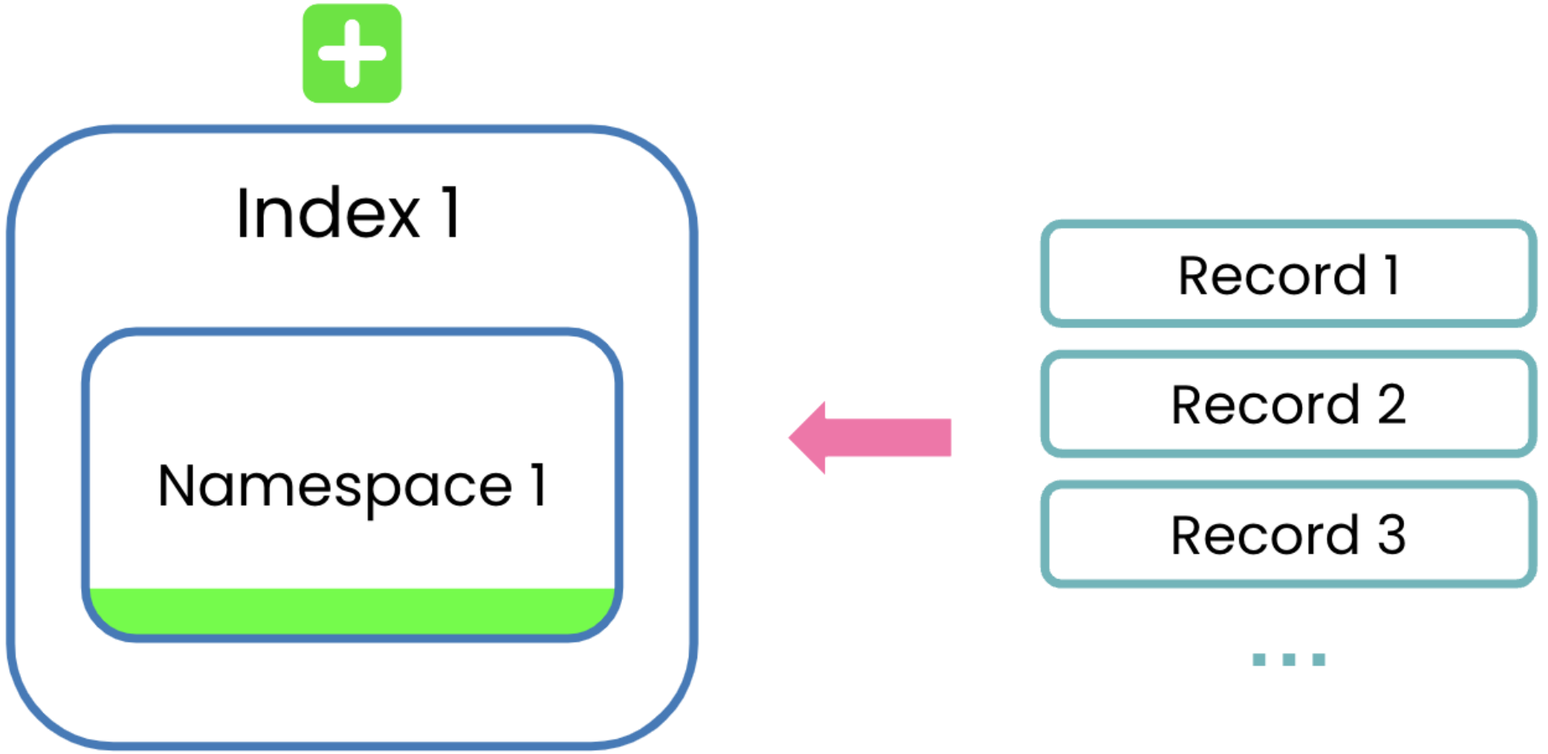
James Chapman
Curriculum Manager, DataCamp

datacamp

Recap...



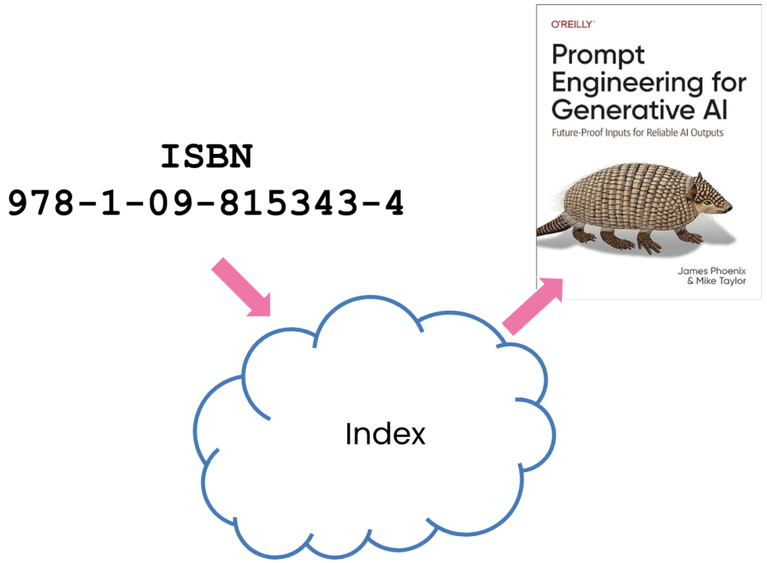
Recap...



Accessing vectors

Fetching

- Retrieve vectors based on their IDs



Querying

- Retrieve similar vectors to an input vector

Fetching vectors

```
index.fetch(
    ids=['0', '1']
)
```

```
{'namespace': '',
 'usage': {'read_units': 1},
 'vectors': {'0': {'id': '0',
                   'metadata': {"genre": "productivity", "year": 2020},
                   'values': [0.0255255547564029694, ...]},
             '1': {'id': '1',
                   'metadata': {"genre": "action", "year": 2023},
                   'values': [-0.0131468913, ...]}}
}
```

[1] https://docs.pinecone.io/guides/data/fetch-data

Read units

- Measure of the resources consumed during read operations:
  - Fetching → **1RU / 10 records**
  - Querying
  - Listing

Serverless

- Up to 2GB storage
  Enough for 300k 1,536-dim vectors
- Up to 2M Write Units per month
- Up to 1M Read Units per month
- 1 Project
- Up to 5 indexes
- Up to 100 namespaces per index

[1] https://www.pinecone.io/pricing/

Fetching vectors from namespaces

```
index.fetch(
    ids=['0', '1']
    namespace='namespace1'
)
```

```
{'namespace': 'namespace1',
 'usage': {'read_units': 1},
 'vectors': {'0': {'id': '0',
                   'metadata': {"genre": "productivity", "year": 2020},
                   'values': [0.0255255547564029694, ...]},
             '1': {'id': '1',
                   'metadata': {"genre": "action", "year": 2023},
                   'values': [-0.0131468913, ...]}}
}
```

Let's practice!

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE

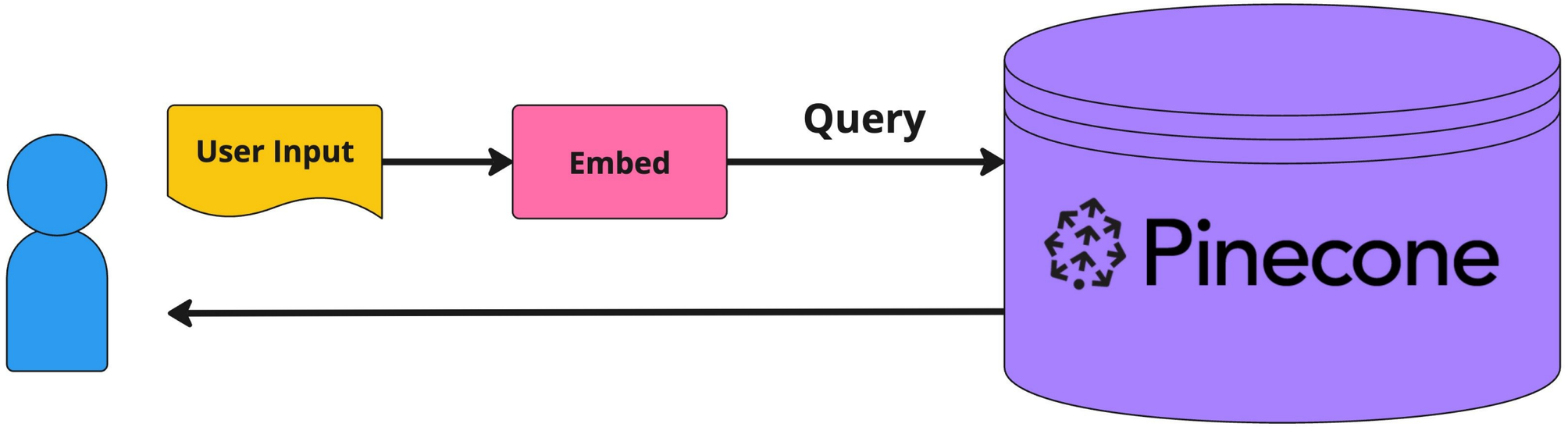
Querying vectors

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE

James Chapman
Curriculum Manager, DataCamp

The power of querying

- **Querying:** receive the most *semantically similar* vectors to an input vector



The .query() method

```
index.query(
  vector=[-0.250919762305275, ...],
  top_k=3
)
```

```
{'matches': [{'id': '1', 'score': 0.0478537641, 'values': []},
             {'id': '2', 'score': 0.046000585, 'values': []},
             {'id': '3', 'score': 0.0458319113, 'values': []}],
 'namespace': '',
 'usage': {'read_units': 5}}
```

The .query() method

```
index.query(
    vector=[-0.250919762305275, ...],
    top_k=3,
    include_values=True
)
```

```
{'matches': [{'id': '1', 'score': 0.0478537641, 'values': [-0.0131468913, ...]},
             {'id': '2', 'score': 0.046000585, 'values': [-0.0120476764, ...]},
             {'id': '3', 'score': 0.0458319113, 'values': [0.00285418332, ...]}],
 'namespace': '',
 'usage': {'read_units': 5}}
```

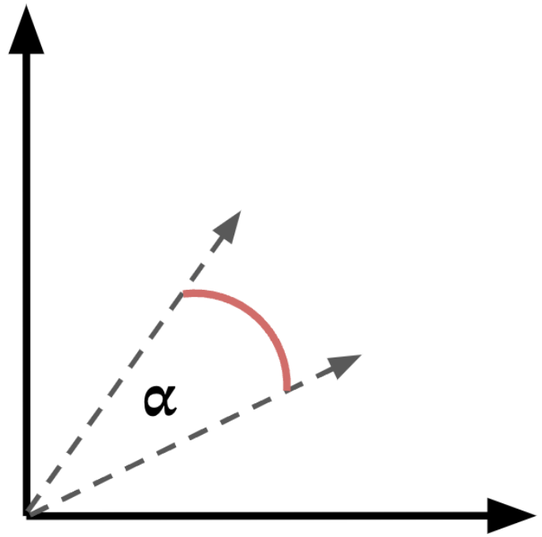
Read units (RUs) for querying

- For querying, RUs is *harder to calculate*
- Dependent on:
  - **No. of records** in the namespace
  - **Size of records**
    - Vector dimensionality
    - Amount of metadata

Records per namespace	Dimension=384	Dimension=768	Dimension=1536
100,000	5 RUs	5 RUs	6 RUs
1,000,000	16 RUs	10 RUs	18 RUs
10,000,000	18 RUs	32 RUs	59 RUs

[1] https://docs.pinecone.io/guides/organizations/manage-cost/understanding-cost#query

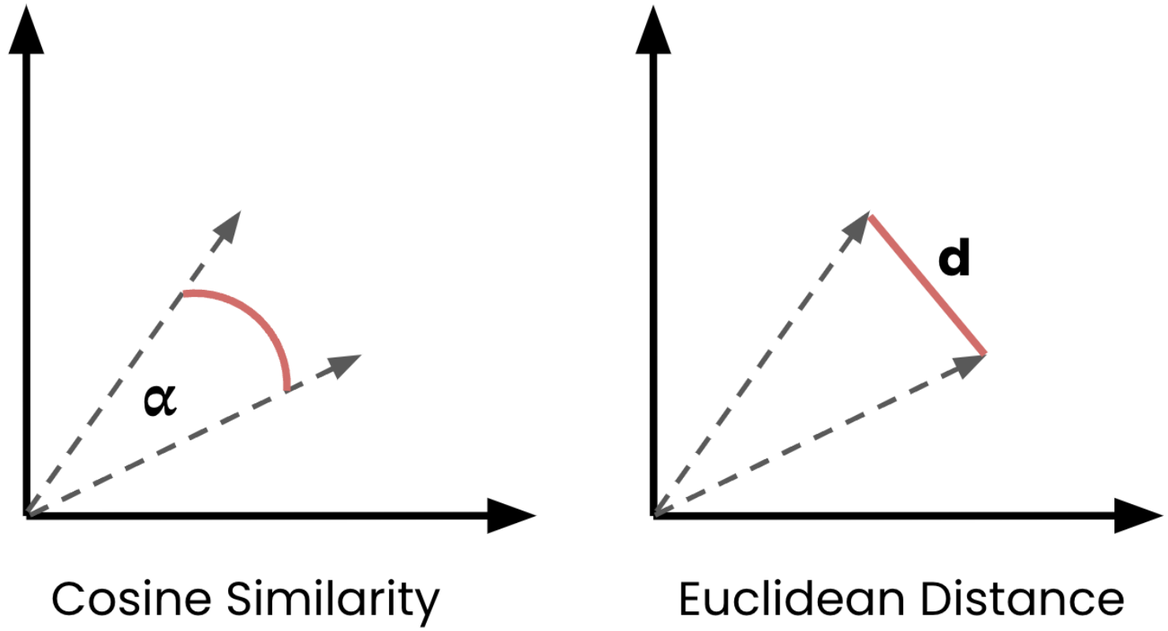
Distance metrics



Cosine Similarity

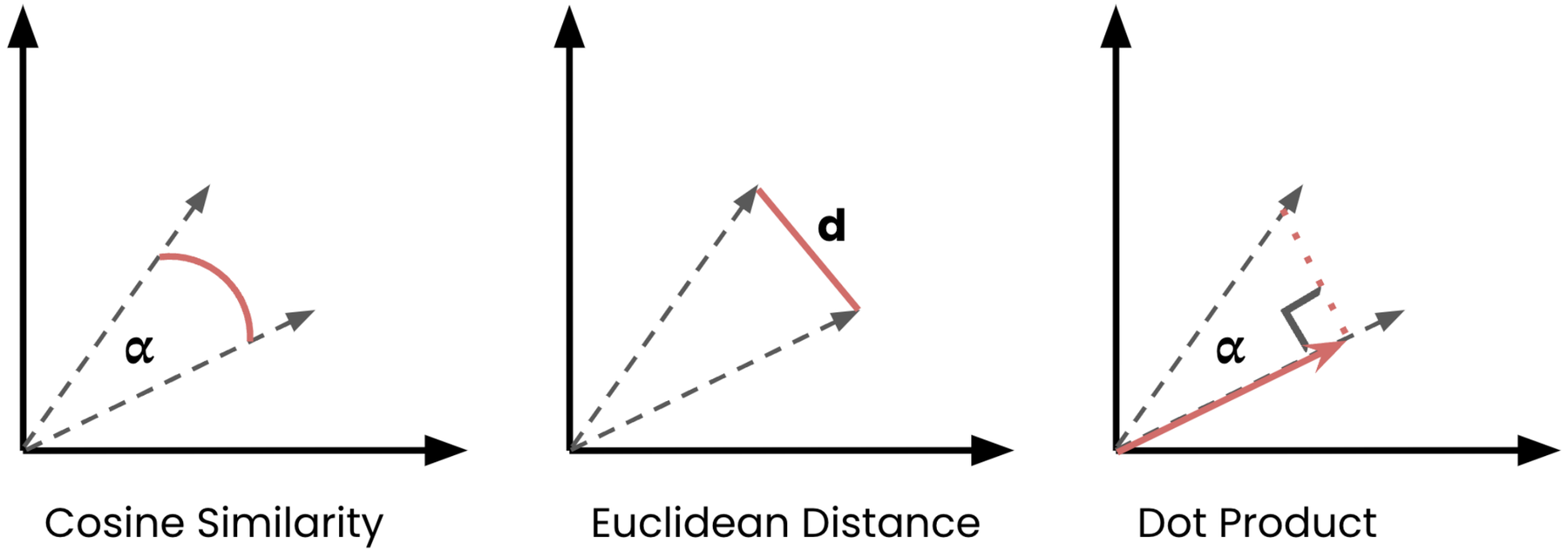
[1] https://docs.pinecone.io/guides/indexes/understanding-indexes#distance-metrics

Distance metrics



[1] https://docs.pinecone.io/guides/indexes/understanding-indexes#distance-metrics

Distance metrics



[1] https://docs.pinecone.io/guides/indexes/understanding-indexes#distance-metrics

Setting the distance metric

```
pc.create_index(
    name="datacamp-index",
    dimension=1536,
    metric='dotproduct',
    spec=ServerlessSpec(
        cloud='aws',
        region='us-east-1'
    )
)
```

- `metric` → `'cosine'`, `'euclidean'`, `'dotproduct'`

Your turn to query!

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE

Metadata filtering

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE

James Chapman
Curriculum Manager, DataCamp

datacamp

Metadata filtering

```
{
    "genre": "action",
    "year": 2020,
    "color": "blue",
    "fit": "straight",
    "price": 29.99,
    "is_jeans": true,
    "areas": ["London", "Kent", "Bath"]
}
```

- Metadata can be *strings*, *numbers*, *Booleans*, and *lists* of strings
- **Metadata filtering**: reduces search space and *query latency*

[1] https://docs.pinecone.io/docs/metadata-filtering

Metadata filtering

```
index.query(
    vector=[-0.250919762305275, ...],
    filter={
        "genre": {"$eq": "documentary"},
        "year": 2019
    },
    top_k=1
)
```

[1] https://docs.pinecone.io/docs/metadata-filtering

Metadata filters

- `$eq` - Equal to (number, string, boolean)
- `$ne` - Not equal to (number, string, boolean)
- `$gt` - Greater than (number)
- `$gte` - Greater than or equal to (number)
- `$lt` - Less than (number)
- `$lte` - Less than or equal to (number)
- `$in` - In array (string or number)
- `$nin` - Not in array (string or number)

[1] https://docs.pinecone.io/docs/metadata-filtering

Metadata filtering - greater than

```
index.query(
    vector=[-0.250919762305275, ...],
    filter={
        "year": {"$gt": 2019},
    },
    top_k=1,
    include_metadatas=True
)
```

```
{'matches': [{'id': '1', 'score': 0.0478537641,
              'values': [],
              'metadata': {'genre': 'action', 'year': 2020}}],
 'namespace': '',
 'usage': {'read_units': 5}}
```

Let's practice!

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE

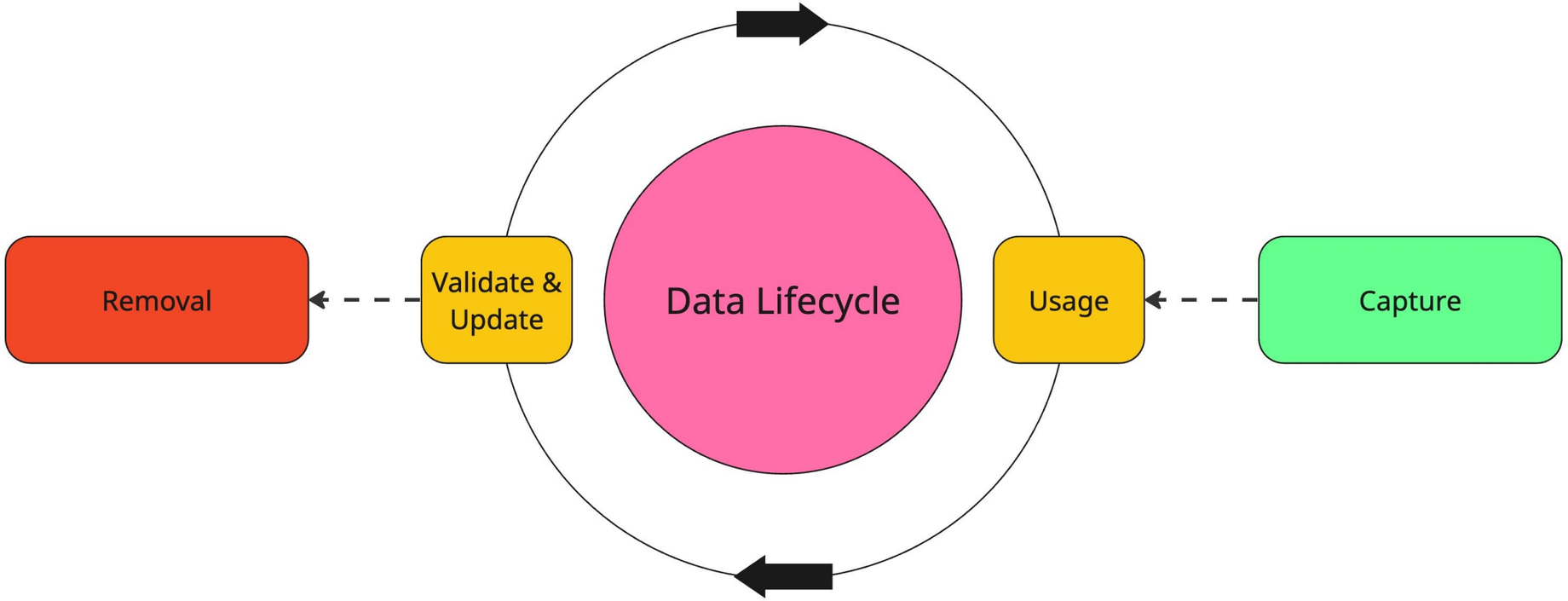
Updating and deleting vectors

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE

James Chapman
Curriculum Manager, DataCamp

Keeping things fresh...

- Keep data *current*
- Optimize query performance
- Maintain **data integrity**



Updating vector values

```
index.fetch(ids=['1'])
```

```
{'namespace': '',
 'usage': {'read_units': 1},
 'vectors': {'1': {'id': '1',
                   'metadata': {"genre": "action", "year": 2023},
                   'values': [-0.0131468913, ...]}
             }
}
```

Updating vector values

```
index.update(
    id="1",
    values=[0.370695321, ...]
)
```

- Ensure: **list length = index dimensionality**

[1] https://docs.pinecone.io/docs/update-data

Updating vector values

```
index.fetch(ids=['1'])
```

```
{'namespace': '',
 'usage': {'read_units': 1},
 'vectors': {'1': {'id': '1',
                   'metadata': {"genre": "action", "year": 2023},
                   'values': [0.370695321, ...]}
             }
}
```

Updating vector metadata

```
index.update(
    id="1",
    set_metadata={"genre": "comedy", "rating": 5}
)
```

[1] https://docs.pinecone.io/docs/update-data

Updating vector metadata

```
index.fetch(ids=['1'])
```

```
{'namespace': '',
 'usage': {'read_units': 1},
 'vectors': {'1': {'id': '1',
                   'metadata': {"genre": "comedy", "year": 2023, "rating": 5},
                   'values': [0.370695321, ...]}
             }
}
```

Updating values and metadata

```
index.update(
    id="1",
    values=[-0.31956, ...],
    set_metadata={"genre": "thriller", "ratings": 4}
)
```

[1] https://docs.pinecone.io/docs/update-data

Deleting vectors

```
index.delete(
    ids=["1", "2"]
)
```

[1] https://docs.pinecone.io/docs/delete-data

Deleting vectors by-metadata

```
index.delete(
    filter={
        "genre": {"$eq": "action"},
    }
)
```

Deleting vectors within a namespace

```
index.delete(delete_all=True, namespace='namespace1')
```

- **Note**: Also removes the namespace!

[1] https://docs.pinecone.io/docs/delete-data

Let's practice!

VECTOR DATABASES FOR EMBEDDINGS WITH PINECONE