Vector Databases

CS 452

Problem Statement

Input:

- A database of news articles

Output:

- We want find similar articles to one we just read
- We want to find entire articles similar to a natural language keyword, i.e., “feel good political stories”

How?

Problem Statement

Input:

- A database of news articles

Output:

- We want find similar articles to one we just read
- We want to find entire articles similar to a natural language keyword, i.e., “feel good political stories”

How?

- Beyond simple regex, neither relational nor non-retational DBs handle this task well

What is semantic search?

- Transform input into numbers
- Similar concepts have similar values

- Save vectors
- Store content with vectors

- Find similar vectors with cosine similarity
- Add rule-based filters using content

https://medium.com/neuml/getting-started-with-semantic-search-a9fd9d8a48cf

What is semantic search?

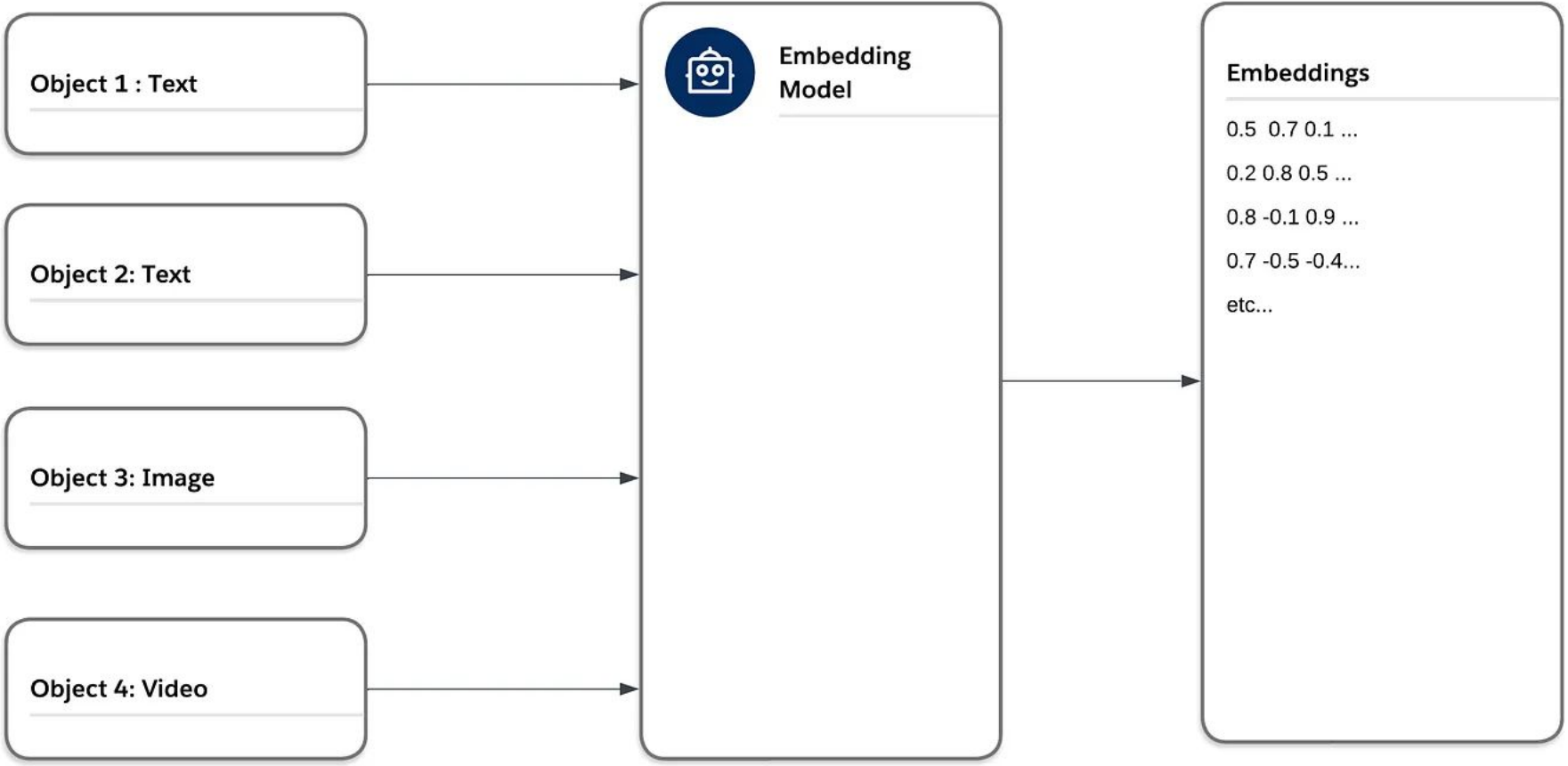
- Transform input into numbers
- Similar concepts have similar values

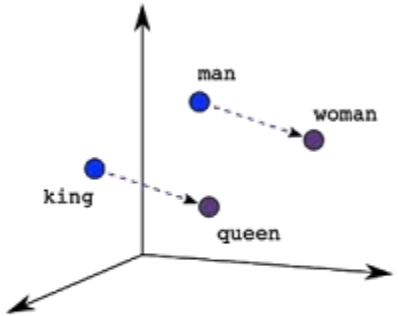
- Save vectors
- Store content with vectors

- Find similar vectors with cosine similarity
- Add rule-based filters using content

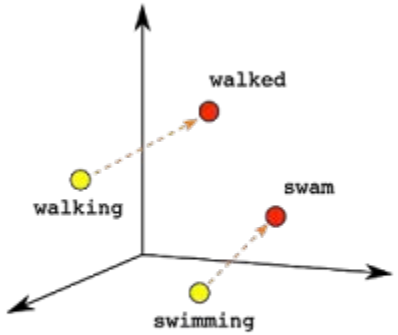
Query	Best Match
feel good story	Maine man wins $1M from $25 lottery ticket
climate change	Canada's last fully intact ice shelf has suddenly collapsed, forming a Manhattan-sized iceberg
public health story	US tops 5 million confirmed virus cases
war	Beijing mobilises invasion craft along coast as Taiwan tensions escalate
wildlife	The National Park Service warns against sacrificing slower friends in a bear attack
asia	Beijing mobilises invasion craft along coast as Taiwan tensions escalate
lucky	Maine man wins $1M from $25 lottery ticket
dishonest junk	Make huge profits without work, earn up to $100,000 a day

https://medium.com/neuml/getting-started-with-semantic-search-a9fd9d8a48cf

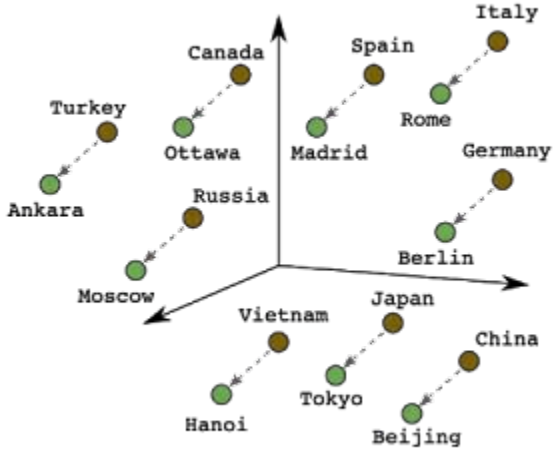

Object 1 : Text
Object 2: Text
Object 3: Image
Object 4: Video
Embedding Model
Embeddings
0.5 0.7 0.1 ...
0.2 0.8 0.5 ...
0.8 -0.1 0.9 ...
0.7 -0.5 -0.4...
etc...



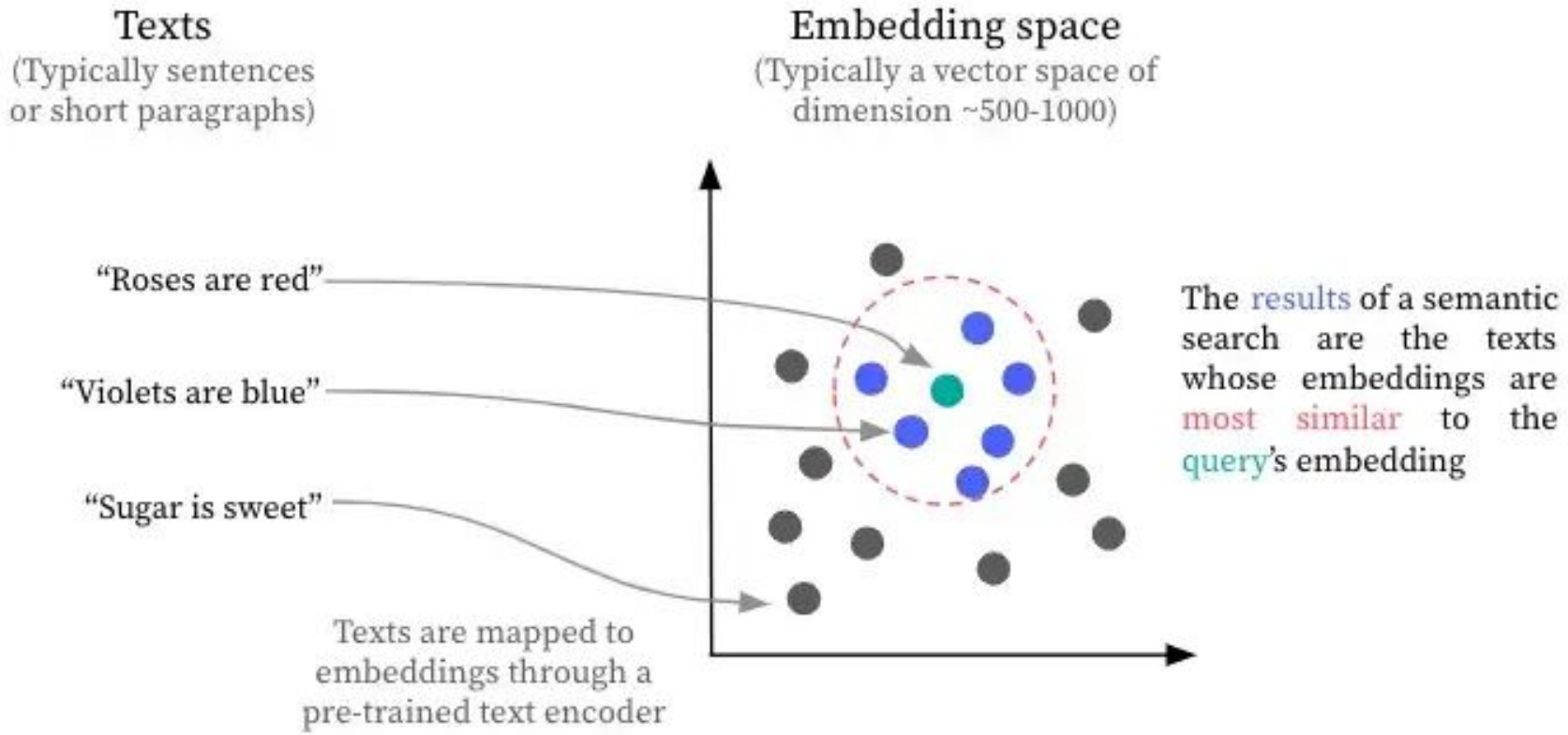
Male-Female



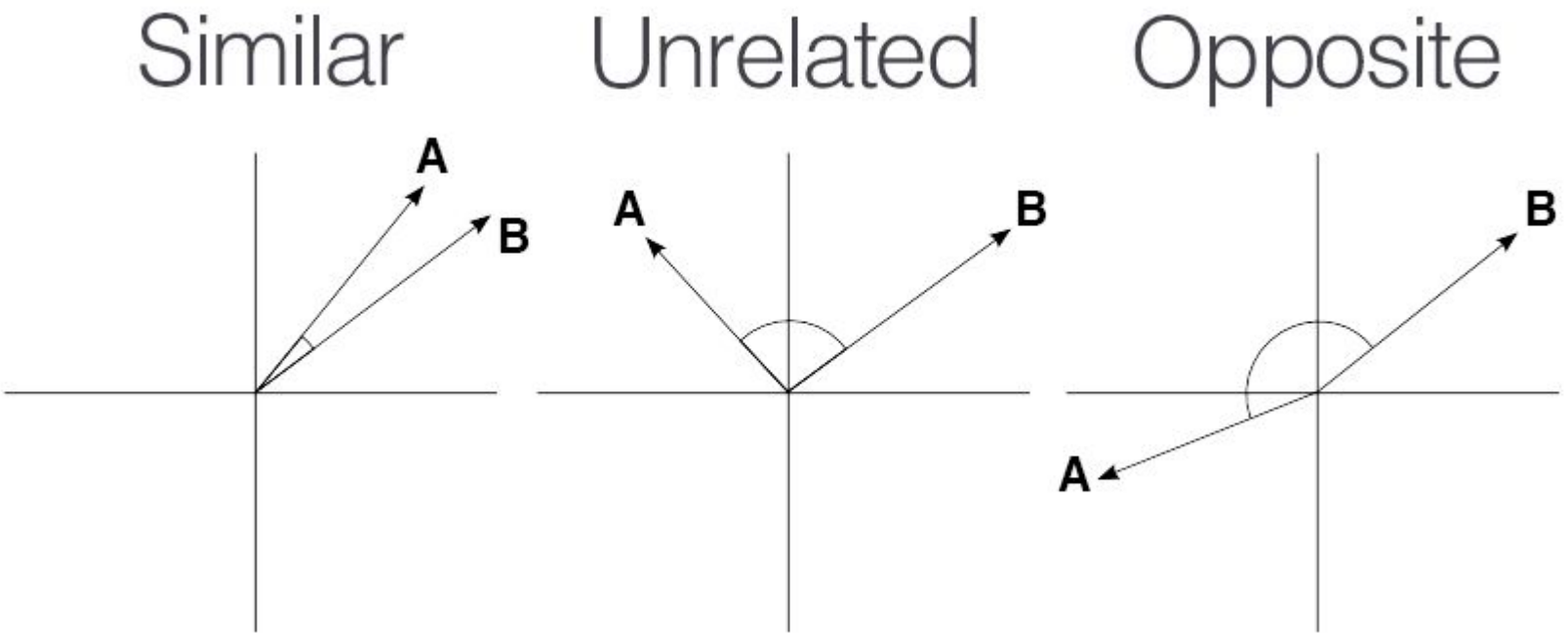
Verb Tense



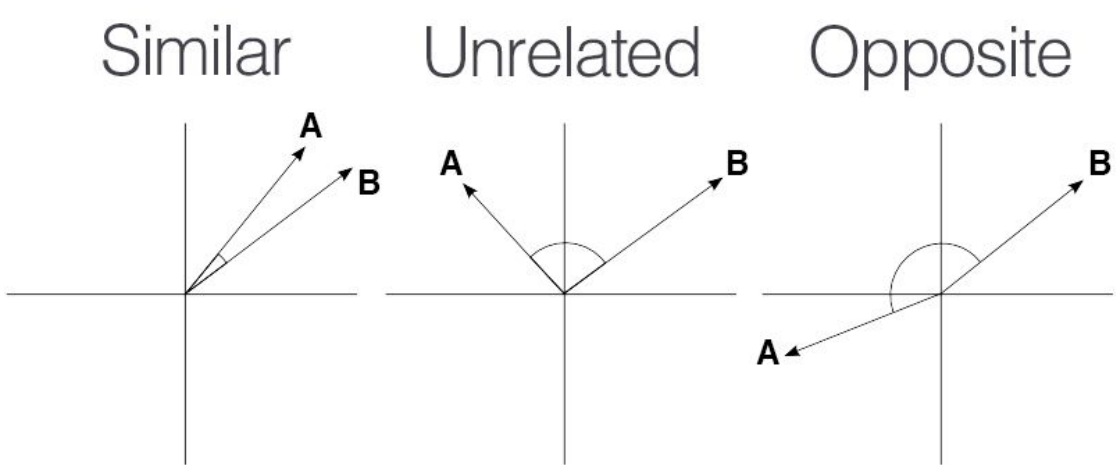
Country-Capital


Texts
(Typically sentences or short paragraphs)
Embedding space
(Typically a vector space of dimension ~500-1000)
"Roses are red"
"Violets are blue"
"Sugar is sweet"
Texts are mapped to embeddings through a pre-trained text encoder
The results of a semantic search are the texts whose embeddings are most similar to the query's embedding

Cosine Similarity



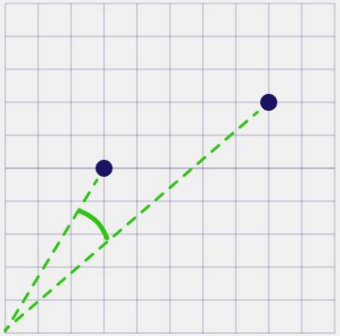
Cosine Similarity



$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}},$$

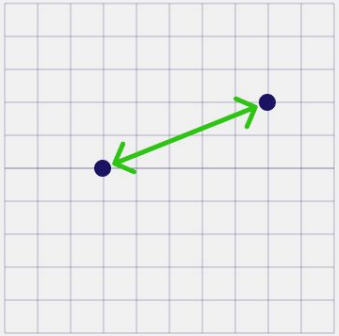
- A measurement of the similarity between two vectors
- Uses angle between vector to assess similarity
- Comprised of dot product (similarity) divided by the total vector length (normalize magnitudes)
- The cosine similarity always falls in the interval [-1, 1]

Distance Metrics in Vector Search



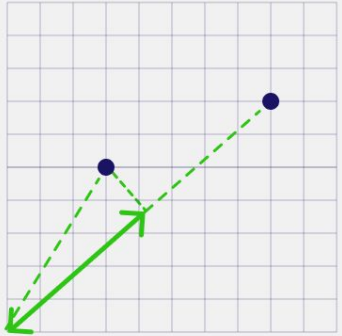
Cosine Distance

$$1 - \frac{A \cdot B}{||A|| \quad ||B||}$$



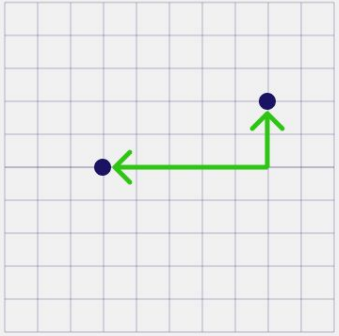
Squared Euclidean (L2 Squared)

$$\sum_{i=1}^{n} (x_i - y_i)^2$$



Dot Product

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

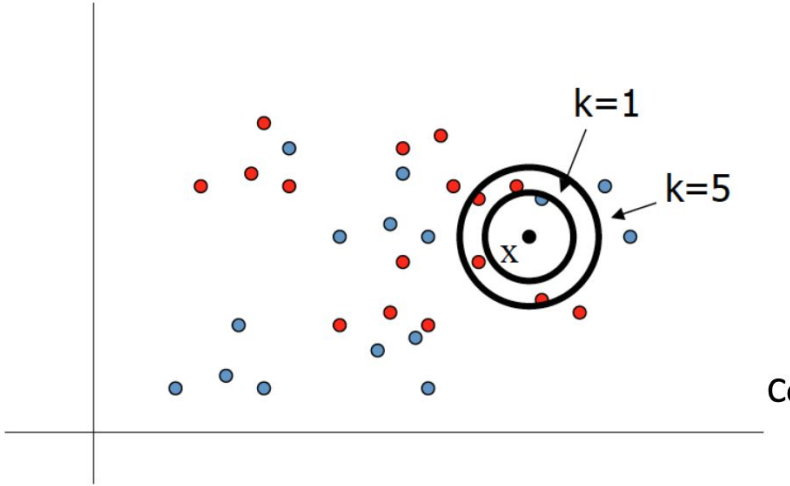


Manhattan (L1)

$$\sum_{i=1}^{n} |x_i - y_i|$$

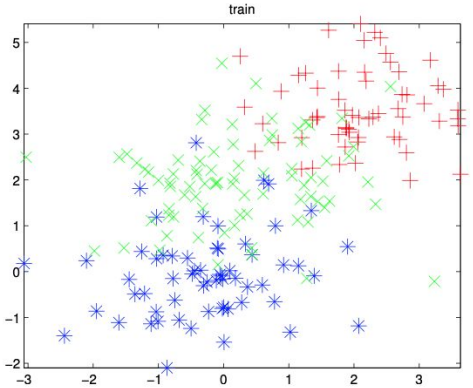
KNN: K-Nearest Neighbor

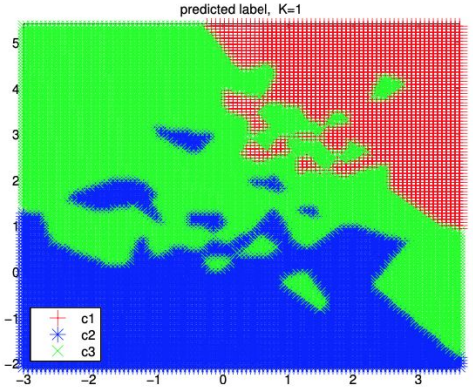
- To classify a new input vector x, examine the k closest training data points to x and assign the object to the most frequently occurring class

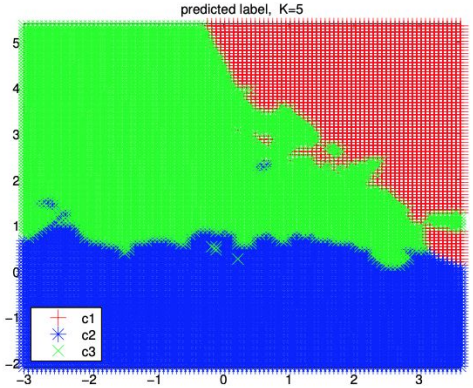


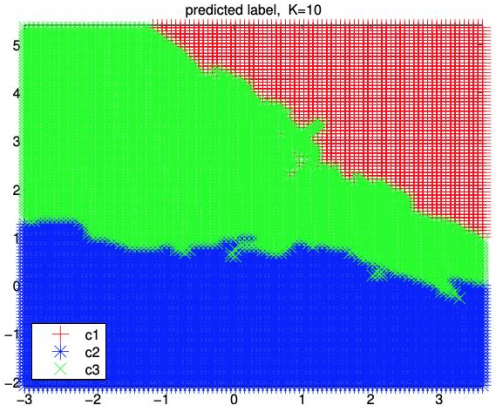
Common values for k: 3, 5

Murphy, Figure 1.15, Figure 1.20









Vector DBs with Postgres (pgvector)

Enable the extension (do this once in each database where you want to use it)

```
CREATE EXTENSION vector;
```

Create a vector column with 3 dimensions

```
CREATE TABLE items (id bigserial PRIMARY KEY, embedding vector(3));
```

Insert vectors

```
INSERT INTO items (embedding) VALUES ('[1,2,3]'), ('[4,5,6]');
```

pgvector Ops

Supported distance functions are:

- `<->` - L2 distance
- `<#>` - (negative) inner product [Also called dot product]
- `<=>` - cosine distance
- `<+>` - L1 distance (added in 0.7.0)
- `<~>` - Hamming distance (binary vectors, added in 0.7.0)
- `<%>` - Jaccard distance (binary vectors, added in 0.7.0)

pgvector Queries

Get the nearest neighbors to a vector

```
SELECT * FROM items ORDER BY embedding <-> '[3,1,2]' LIMIT 5;
```

pgvector Queries

Get the nearest neighbors to a vector

```
SELECT * FROM items ORDER BY embedding <-> '[3,1,2]' LIMIT 5;
```

Get rows within a certain distance

```
SELECT * FROM items WHERE embedding <-> '[3,1,2]' < 5;
```

pgvector Queries

Get the nearest neighbors to a vector

```
SELECT * FROM items ORDER BY embedding <-> '[3,1,2]' LIMIT 5;
```

Get rows within a certain distance

```
SELECT * FROM items WHERE embedding <-> '[3,1,2]' < 5;
```

pgvector Aggregate Queries

Average vectors

```
SELECT AVG(embedding) FROM items;
```

Average groups of vectors

```
SELECT category_id, AVG(embedding) FROM items GROUP BY category_id;
```

Read the full docs!

Many more, very powerful features:

https://github.com/pgvector/pgvector