

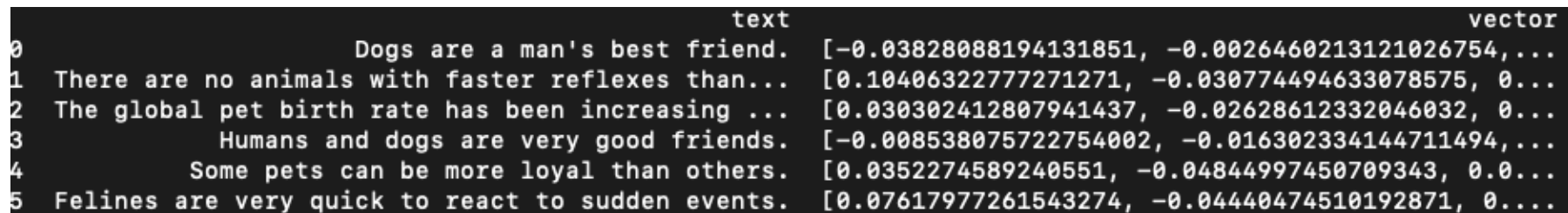
Creating and Querying a Local Vector Database

Now that we have both our data points (which are the sentences) and their corresponding vectors, let's create a local vector “database” using `pandas` as it is a library that can be used to create DataFrames, which are comparable to SQL tables.

Add the following to `demo.py` :

```
1 import pandas as pd
2
3 # Create a dict with the example data
4 data = {
5     'text': examples,
6     'vector': embeddings
7 }
8
9 # Create a DataFrame using the dictionary
10 df = pd.DataFrame.from_dict(data)
11
12 print(df)
```

The output should represent a SQL-like table, and should be similar to the screenshot below:



	text	vector
0	Dogs are a man's best friend.	[-0.03828088194131851, -0.0026460213121026754, ...
1	There are no animals with faster reflexes than...	[0.10406322777271271, -0.030774494633078575, 0...
2	The global pet birth rate has been increasing ...	[0.030302412807941437, -0.02628612332046032, 0...
3	Humans and dogs are very good friends.	[-0.008538075722754002, -0.016302334144711494, ...
4	Some pets can be more loyal than others.	[0.0352274589240551, -0.04844997450709343, 0.0...
5	Felines are very quick to react to sudden events.	[0.07617977261543274, -0.04440474510192871, 0....

Alright! Let's now define a function to query this database!

Example: Article Recommendation

Suppose one evening, a user is reading an article named “Why humans get along so well with dogs”.

Imagine that our example sentences represent article headlines.

How can we generate suggestions for the user? Well, since we have vector embeddings for each of our articles, we can compare the similarity of each headline to the one the user is currently reading by comparing the dot products of the current article’s headline’s vector with every other vector in the database!

Let’s construct this example in our code:

```
1  user_article_headline = 'Why humans get along so well with dogs'
2
3  # Fetch the vector embedding of the headline of the article that the user is currently reading
4  response = requests.post(api_url, headers=headers, json={"inputs": [user_article_headline], "options":{"wait_for_model":True}})
5  user_article_embedding = response.json()[0]
6
7  # Get list of vectors from our database table
8  vectors = df["vector"].values.tolist()
9
10 # Create a list of the dot products of user_article_embedding against each vector in the database
11 similarities = [dot_product(user_article_embedding, vector) for vector in vectors]
12 print("Similarity scores:", similarities)
13
14 # Get the index of the value with the highest similarity score
15 max_similarity_index = similarities.index(max(similarities))
16
17 # Fetch the sentence corresponding to that index from the dataframe
18 sentences = df["text"].values.tolist()
19 most_similar_sentence = sentences[max_similarity_index]
20
21 print(most_similar_sentence)
```

Output:

1 Similarity scores: [0.6367540860345434, 0.25669272034699475, 0.3402974996977122, 0.7539815812800277, 0.6026046867479701, 0.33806413301719257]
2
3 Humans and dogs are very good friends.

As we can see from the output, the most similar sentence was “Humans and dogs are very good friends” with a similarity of around 0.75 - therefore it will be recommended to the user!