Building a Search Applications

Introduction

In this lesson, we will cover:

- Semantic vs Keyword search.
- What are Text Embeddings.
- Creating a Text Embeddings Index.
- Searching a Text Embeddings Index.

Learning Goals

After completing this lesson, you will be able to:

- Tell the difference between semantic and keyword search.
- Explain what Text Embeddings are.
- Create an application using Embeddings to search for data.

What is semantic search?

Now you might be wondering, what is semantic search? Semantic search is a search technique that uses the semantics, or meaning, of the words in a query to return relevant results.

Here is an example of a semantic search. Let's say you were looking to buy a car, you might search for 'my dream car', semantic search understands that you are not `dreaming` about a car, but rather you are looking to buy your `ideal` car. Semantic search understands your intention and returns relevant results. The alternative is `keyword search` which would literally search for dreams about cars and often returns irrelevant results.

What are Text Embeddings?

Text embeddings are a text representation technique used in natural language processing. Text embeddings are semantic numerical representations of text. Embeddings are used to represent data in a way that is easy for a machine to understand. There are many models for building text embeddings, in this lesson, we will focus on generating embeddings using the OpenAI Embedding Model.

Here's an example, imagine the following text is in a transcript from one of the episodes on the AI Show YouTube channel:

```
Today we are going to learn about Azure Machine Learning.
```

We'd pass the text to the OpenAI Embedding API and it would return the following embedding consisting of 1536 numbers aka a vector. Each number in the vector represents a different aspect of the text. For brevity, here are the first 10 numbers in the vector.

```
[-0.006655829958617687, 0.0026128944009542465, 0.008792596869170666, -0.02446001023054123, -0.008540431968867779, 0.022071078419685364, -0.010703742504119873, 0.003311325330287218, -0.011632772162556648, -0.02187200076878071, ...]
```

Vector Databases

For lesson simplicity, the Embedding Index is stored in a JSON file named `embedding_index_3m.json` and loaded into a Pandas DataFrame. However, in production, the Embedding Index would be stored in a vector database such as Azure Cognitive Search, Redis, Pinecone, Weaviate, to name but a few.

Understanding cosine similarity

We've learned about text embeddings, the next step is to learn how to use text embeddings to search for data and in particular find the most similar embeddings to a given query using cosine similarity.

What is cosine similarity?

Cosine similarity is a measure of similarity between two vectors, you'll also hear this referred to as `nearest neighbor search`. To perform a cosine similarity search you need to *vectorize* for *query* text using the OpenAI Embedding API. Then calculate the *cosine similarity* between the query vector and each vector in the Embedding Index. Remember, the Embedding Index has a vector for each YouTube transcript text segment. Finally, sort the results by cosine similarity and the text segments with the highest cosine similarity are the most similar to the query.

From a mathematic perspective, cosine similarity measures the cosine of the angle between two vectors projected in a multidimensional space. This measurement is beneficial, because if two documents are far apart by Euclidean distance because of size, they could still have a smaller angle between them and therefore higher cosine similarity. For more information about cosine similarity equations, see Cosine similarity.