Semantic Search of Similar User Profiles Using Embeddings

Search Technology

Mykyta Chernenko

Overview of Embeddings

- **Embeddings**:
  - Embeddings are dense vector representations of text that capture the semantic relationships and contextual meanings of words and phrases.
  - They are generated using machine learning models that transform textual data into high-dimensional numerical vectors, enabling computers to perform complex language understanding tasks.
  - Embeddings are typically stored in vector databases. This storage approach facilitates rapid similarity searches and comparisons, which are essential for semantic search and recommendation systems.
- **Cosine Similarity**: Measures the cosine of the angle between two vectors.

```
def cosine_similarity(vec1, vec2):
  return np.dot(vec1, vec2) / (np.linalg.norm(vec1) * np.linalg.norm(vec2))
```

OpenAI Embeddings

- **Features**:
  - Trained on diverse internet data for robust semantic understanding.
  - Optimized for tasks like clustering, classification, and similarity search.
  - Higher accuracy in capturing contextual nuances.
  - Pretty cheap
- **Usage Example**:

```
import openai
openai.api_key = 'sk-proj-****'

def get_embedding(text, model=\"text-embedding-ada-002\"):
  response = openai.Embedding.create(input=text, model=model)
  return response['data'][0]['embedding']
```

Storing and Managing Embeddings

- **Storage Solutions**:
  - **Databases**: SQL or NoSQL databases with support for vector data.
  - **Vector Stores**: Specialized databases like Pinecone or FAISS.
- **Data Structure**: Each user profile has associated metadata and its embedding vector.
- **Example Storage Schema**:

User ID	Job	Bio	Embedding Vector
1	Developer	I love making music...	[0.12, 0.34, ..., 0.56]
2	Freelancer	Lass mal Sommer 24 komplett abreißen!	[0.78, 0.23, ..., 0.89]

Calculating Similarity with Cosine Similarity

- **Function to Calculate Cosine Similarity**:

```
import numpy as np
def cosine_similarity(vec1, vec2):
  return np.dot(vec1, vec2) / (np.linalg.norm(vec1) * np.linalg.norm(vec2))
```

- **Adjusting Similarity Scores**:

```
def get_bio_scores(vec1, vec2):
  similarity = cosine_similarity(vec1, vec2)
  similarity -= 0.3  # Penalize low scores
  if similarity < 0:
    similarity = 0
  return similarity * 30
```

- **Usage Example**:

```
score = get_bio_scores(embedding1, embedding2)
```

Example Workflow with Code

1. **Generate Embeddings**:

```
def get_embedding(text, model=\"text-embedding-ada-002\"):
    response = openai.Embedding.create(input=text, model=model)
    return response['data'][0]['embedding']

user_bio = data.get('job') + ' ' + data.get('bio')
embedding = get_embedding(user_bio)
```

2. **Store Embeddings** in the database/vector store.
3. **Calculate Similarity** between user profiles:

```
  score = get_bio_scores(embedding1, embedding2)
```

4. **Retrieve and Present** similar profiles based on scores.

Project Wooh: Connecting Expat Friends

- **Purpose**: Facilitate connections between likeminded expatriates.
- **Goal**:
  - Analyze user bios to find good matching friends.
  - Enhance social experiences for expats by leveraging semantic search.
- **Approach**:
  - Utilize embeddings to capture the essence of user profiles.
  - Implement similarity calculations to recommend compatible friends.
- **Outcome**:
  - A platform where users can find friends with similar interests and backgrounds.

Demonstration and Examples

- **User Profiles**:
  - **Profile 1**:Developer, I love making music and I also love talking about music. I play guitar and im just very into it. :D Mostly Rock/Metal at the moment but i also love rap and lofi, playing volleyball sometimes...
  - **Profile 2**: Freelancer, I love dancing in general but currently trying to learn bachata (pretty bad at it..haha) . I consider myself open minded, calm, spiritual and empathetic. If you would want to play Ping pong, Volleyball or try dance class, hit me up ✌️😄
  - **Score**: 0.64

Demonstration and Examples

- **Similarity Scores**:
  - High score indicates strong similarity in interests and activities.
  - Lower scores reflect less compatibility.
- **Code Execution**:

```
score = get_bio_scores(embedding1, embedding2)
print(f\"Score: {score}\")
```

- **Result Interpretation**:
  - Profiles with scores above a threshold are recommended as potential friends.