ChatGPT API and Word Embeddings

ChatGPT API and Word Embeddings

Initial Request: Accessing ChatGPT API and Retrieving Embeddings

Human: Please write code that will send text to ChatGPT 3.5 Turbo and return the vector embedding. Use my ChatGPT API key.

Assistant: I've created code that demonstrates how to use OpenAI's API to both get responses from ChatGPT 3.5 Turbo and retrieve embeddings for text. The code is working successfully:

```
# Import necessary libraries
import os
import openai
import json

# Set up the OpenAI client with API key from environment variables
if 'OPEN_AI_KEY' in os.environ:
    client = openai.OpenAI(api_key=os.environ['OPEN_AI_KEY'])
    print("OpenAI client initialized with API key from environment variables.")
else:
    print("OpenAI API key not found in environment variables. Please provide your API key.")

# Function to get a response from ChatGPT 3.5 Turbo
def get_chatgpt_response(prompt):
    try:
        response = client.chat.completions.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": "You are a helpful assistant."},
                {"role": "user", "content": prompt}
            ]
```

```
        )
        return response.choices[0].message.content
    except Exception as e:
        return f"Error: {str(e)}"

# Function to get embeddings for a text
def get_embedding(text):
    try:
        response = client.embeddings.create(
            model="text-embedding-ada-002",
            input=text
        )
        return response.data[0].embedding
    except Exception as e:
        return f"Error: {str(e)}"

# Example usage
if 'OPEN_AI_KEY' in os.environ:
    # Example text
    example_text = "Hello, world! This is a test for embeddings."

    # Get response from ChatGPT
    print("Getting response from ChatGPT 3.5 Turbo...")
    response = get_chatgpt_response(example_text)
    print("\nChatGPT Response:")
    print(response)

    # Get embedding
    print("\nGetting embedding for the text...")
    embedding = get_embedding(example_text)

    # Print the first 10 values of the embedding vector (to avoid printing the entire vector)
    print("\nEmbedding (first 10 values):")
    if isinstance(embedding, list):
        print(embedding[:10])
        print(f"Total embedding dimensions: {len(embedding)}")
    else:
        print(embedding)  # This will print the error message if there was an error
else:
    print("Cannot demonstrate API calls without an API key.")
```

The code includes two main functions: 1. `get_chatgpt_response()` - Sends text to ChatGPT

and returns its response 2. `get_embedding()` - Gets the embedding vector for a given text

When testing with the example text "Hello, world! This is a test for embeddings.", the embedding is a 1536-dimensional vector that represents the semantic meaning of the text in a way that can be used for various NLP tasks like similarity comparison, clustering, or classification.

Word Embeddings for King, Woman, Man, Queen

Human: Please send the following words to chatgpt one at a time, get the embedding for each, and assign each vector to a column of a dataframe with the word as the column name: "king", "woman", "man", "queen".

Assistant: I've successfully retrieved the embeddings for each of the words "king", "woman", "man", and "queen" and created a dataframe with each word as a column name. Each column contains the 1536-dimensional embedding vector for that word.

```
# Import necessary libraries
import os
import openai
import pandas as pd
import numpy as np
from tqdm import tqdm

# Set up the OpenAI client with API key from environment variables
if 'OPEN_AI_KEY' in os.environ:
    client = openai.OpenAI(api_key=os.environ['OPEN_AI_KEY'])
    print("OpenAI client initialized with API key from environment variables.")
else:
    print("OpenAI API key not found in environment variables. Please provide your API key.")

# Function to get embeddings for a text
def get_embedding(text):
    try:
        response = client.embeddings.create(
            model="text-embedding-ada-002",
            input=text
        )
        return response.data[0].embedding
    except Exception as e:
        return f"Error: {str(e)}"

# Words to get embeddings for
```

```
words = ["king", "woman", "man", "queen"]

# Create a dictionary to store embeddings
embeddings_dict = {}

# Get embeddings for each word
for word in tqdm(words, desc="Getting embeddings"):
    embedding = get_embedding(word)
    if isinstance(embedding, list):
        embeddings_dict[word] = embedding
    else:
        print(f"Error getting embedding for '{word}': {embedding}")

# Check if we have embeddings for all words
if len(embeddings_dict) == len(words):
    # Create a DataFrame with embeddings
    df = pd.DataFrame(embeddings_dict)

    # Display information about the DataFrame
    print(f"\nCreated DataFrame with shape: {df.shape}")
    print(f"Column names: {df.columns.tolist()}")

    # Display the first few rows of the DataFrame
    print("\nFirst 5 rows of the DataFrame:")
    print(df.head())
else:
    print("Could not get embeddings for all words.")
```

I also created a visualization of these word embeddings after reducing them to 2 dimensions using PCA:

```
# Let's visualize the relationships between these word embeddings
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Convert the DataFrame to a format suitable for PCA
# Each row is a word, each column is a dimension of the embedding
embeddings_matrix = df.T.values

# Apply PCA to reduce to 2 dimensions for visualization
pca = PCA(n_components=2)
reduced_embeddings = pca.fit_transform(embeddings_matrix)
```

```
# Create a scatter plot
plt.figure(figsize=(10, 8))
plt.scatter(reduced_embeddings[:, 0], reduced_embeddings[:, 1], c='blue', alpha=0.7)

# Add labels for each point
for i, word in enumerate(words):
    plt.annotate(word, (reduced_embeddings[i, 0], reduced_embeddings[i, 1]),
                 fontsize=12, ha='center', va='bottom')

# Add arrows to show relationships
# For example, the famous king - man + woman = queen relationship
plt.arrow(reduced_embeddings[0, 0], reduced_embeddings[0, 1],  # king
          reduced_embeddings[2, 0] - reduced_embeddings[0, 0],  # to man
          reduced_embeddings[2, 1] - reduced_embeddings[0, 1],
          color='red', alpha=0.4, width=0.01, head_width=0.05)

plt.arrow(reduced_embeddings[1, 0], reduced_embeddings[1, 1],  # woman
          reduced_embeddings[3, 0] - reduced_embeddings[1, 0],  # to queen
          reduced_embeddings[3, 1] - reduced_embeddings[1, 1],
          color='green', alpha=0.4, width=0.01, head_width=0.05)

plt.title('Word Embeddings Reduced to 2D using PCA', fontsize=14)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Let's also calculate some vector arithmetic to test the famous relationship:
# king - man + woman  queen
king_vec = df['king'].values
man_vec = df['man'].values
woman_vec = df['woman'].values
queen_vec = df['queen'].values

# Calculate king - man + woman
result_vec = king_vec - man_vec + woman_vec

# Calculate cosine similarity between result and queen
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Reshape vectors for cosine_similarity function
result_vec_reshaped = result_vec.reshape(1, -1)
```