Mistral AI Embeddings

To provide embedding examples using Mistral AI, you can leverage their embeddings API to generate numerical representations (embeddings) of text that capture semantic meaning. These embeddings are useful for tasks such as similarity detection, clustering, classification, or search. Below, I'll demonstrate how to generate embeddings for a set of sentences and calculate their similarities using the Mistral AI embeddings endpoint. This example builds on the structure of the provided code but adapts it specifically for embeddings, as outlined in the documentation: Mistral AI Embeddings.

Prerequisites

Before running the example, ensure you have: 1. **API Key**: A valid Mistral AI API key stored in the `MISTRAL_API_KEY` environment variable. 2. **Dependencies**: The required Python packages installed. You can install them with: `bash pip install requests numpy`

Example Code

Here's a complete Python script that generates embeddings for three sentences and calculates their pairwise similarities:

```
import os
import requests
import numpy as np

def call_mistral_embeddings_api(texts, model="mistral-embed"):
    """
    Call the Mistral AI embeddings API to generate embeddings for a list of texts.

    Args:
        texts (list): List of strings to embed.
        model (str): The embedding model to use (default: "mistral-embed").

    Returns:
        list: List of embedding vectors, or None if the request fails.
    """
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print("Error: MISTRAL_API_KEY environment variable not set.")
        return None
```

```
    url = "https://api.mistral.ai/v1/embeddings"
    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json",
        "Authorization": f"Bearer {api_key}"
    }
    data = {
        "model": model,
        "input": texts
    }

    try:
        response = requests.post(url, headers=headers, json=data)
        response.raise_for_status()
        response_json = response.json()
        if response_json and "data" in response_json:
            embeddings = [item["embedding"] for item in response_json["data"]]
            return embeddings
        else:
            print(f"Mistral Embeddings API Error: Invalid response format: {response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Mistral Embeddings API Error: {e}")
        if e.response:
            print(f"Response status code: {e.response.status_code}")
            print(f"Response content: {e.response.text}")
        return None

def calculate_similarity(emb1, emb2):
    """
    Calculate the similarity between two embeddings using dot product.

    Args:
        emb1 (list): First embedding vector.
        emb2 (list): Second embedding vector.

    Returns:
        float: Similarity score (dot product, equivalent to cosine similarity for normalized vectors).
    """
    return np.dot(emb1, emb2)
```

```python
if __name__ == "__main__":
    # Example texts to embed
    texts = [
        "I love programming in Python.",
        "Python is a great programming language.",
        "The weather is sunny today."
    ]

    # Generate embeddings
    embeddings = call_mistral_embeddings_api(texts)
    if embeddings:
        # Print embedding dimension
        print(f"Embedding dimension: {len(embeddings[0])}")

        # Calculate pairwise similarities
        sim_12 = calculate_similarity(embeddings[0], embeddings[1])
        sim_13 = calculate_similarity(embeddings[0], embeddings[2])
        sim_23 = calculate_similarity(embeddings[1], embeddings[2])

        # Display results
        print(f"\nSimilarity Results:")
        print(f"Text 1: '{texts[0]}'")
        print(f"Text 2: '{texts[1]}'")
        print(f"Text 3: '{texts[2]}'")
        print(f"\nSimilarity between Text 1 and Text 2: {sim_12:.4f}")
        print(f"Similarity between Text 1 and Text 3: {sim_13:.4f}")
        print(f"Similarity between Text 2 and Text 3: {sim_23:.4f}")
```

How to Run

1. **Set the API Key**:

   ```
   export MISTRAL_API_KEY="your_api_key_here"
   ```

2. **Save and Execute**: Save the script (e.g., as `embedding_example.py`) and run it:

   ```
   python embedding_example.py
   ```

Expected Output

Assuming the API call succeeds, you'll see output like this (exact values depend on the embeddings returned):

```
Embedding dimension: 1024

Similarity Results:
Text 1: 'I love programming in Python.'
Text 2: 'Python is a great programming language.'
Text 3: 'The weather is sunny today.'

Similarity between Text 1 and Text 2: 0.9200
Similarity between Text 1 and Text 3: 0.6500
Similarity between Text 2 and Text 3: 0.6700
```

Explanation

- **API Endpoint**: The `call_mistral_embeddings_api` function sends a POST request to `https://api.mistral.ai/v1/e` passing a list of texts and the `"mistral-embed"` model. The API returns a JSON response containing embeddings under the `"data"` key.
- **Embeddings**: Each embedding is a 1024-dimensional vector (as per Mistral's documentation), representing the semantic content of the input text. The embeddings are normalized to a norm of 1.
- **Similarity Calculation**: Since the embeddings are normalized, the dot product (`np.dot`) between two embeddings equals their cosine similarity. Higher values indicate greater semantic similarity:
  - **Text 1 and Text 2**: Both are about programming in Python, so their similarity should be high (e.g., ~0.92).
  - **Text 1 and Text 3**: One is about programming, the other about weather, so their similarity should be lower (e.g., ~0.65).
  - **Text 2 and Text 3**: Similar pattern, lower similarity due to differing topics.

Additional Use Cases

Embeddings from Mistral AI can be applied beyond similarity detection: - **Clustering**: Group similar texts using algorithms like K-means (requires `scikit-learn`). `python from sklearn.cluster import KMeans kmeans = KMeans(n_clusters=2).fit(np.array(embeddings)) print(kmeans.labels_)` - **Search**: Embed a query and find the most similar text: `python query = "Tell me about Python." query_emb = call_mistral_embeddings_api([query])[0] similarities = [calculate_similarity(query_emb, emb) for emb in embeddings] most_similar = texts[np.argmax(similarities)] print(f"Most similar text: '{most_similar}'")` - **Classification**: Use embeddings as features in a machine learning model.

Notes

- **Model**: The `"mistral-embed"` model is optimized for embeddings. Check the documentation for updates or additional models.
- **Batching**: The API processes multiple texts in one call, improving efficiency.
- **Advanced Applications**: For large-scale similarity search, integrate with vector databases like Faiss or Milvus.

This example provides a practical introduction to using Mistral AI embeddings, adaptable to the structure of the provided chat completion code by changing the endpoint and data format.