Mistral AI 向量

要使用 Mistral AI 提供的嵌入示例，可以利用他們的嵌入 API 生成數值表示（嵌入）的文本，這些嵌入捕捉了語義意義。這些嵌入對於類似性檢測、聚類、分類或搜索等任務非常有用。以下，我將展示如何為一組句子生成嵌入並計算它們的相似性，使用 Mistral AI 的嵌入端點。這個示例基於提供的代碼結構，但專門針對嵌入進行了適應，如文檔中所述：Mistral AI 嵌入。

前提條件

在運行示例之前，請確保您擁有：1. **API 密鑰**：有效的 Mistral AI API 密鑰存儲在 `MISTRAL_API_KEY` 環境變量中。2. **依賴項**：安裝所需的 Python 包。您可以使用以下命令安裝它們：`bash pip install requests numpy`

示例代碼

以下是一個完整的 Python 腳本，為三個句子生成嵌入並計算它們的相似性：

```
import os
import requests
import numpy as np

def call_mistral_embeddings_api(texts, model="mistral-embed"):
    """
    請求 Mistral AI 嵌入 API 為一組文本生成嵌入。

    Args:
        texts (list): 要嵌入的字符串列表。
        model (str): 要使用的嵌入模型（默認值："mistral-embed"）。

    Returns:
        list: 嵌入向量列表，如果請求失敗則返回 None。
    """
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print(" 錯誤：MISTRAL_API_KEY 環境變量未設置。")
        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 嵌入 API 錯誤：無效的響應格式：{response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Mistral 嵌入 API 錯誤：{e}")
        if e.response:
            print(f" 響應狀態碼：{e.response.status_code}")
            print(f" 響應內容：{e.response.text}")
        return None

def calculate_similarity(emb1, emb2):
    """
    使用點積計算兩個嵌入之間的相似性。

    Args:
        emb1 (list): 第一個嵌入向量。
        emb2 (list): 第二個嵌入向量。

    Returns:
        float: 相似性得分（點積，等同於標準化向量的餘弦相似性）。
    """
    return np.dot(emb1, emb2)

if __name__ == "__main__":
    # 要嵌入的示例文本
    texts = [
```

```
        " 我喜歡用 Python 編程。",
        "Python 是一種很棒的編程語言。",
        " 今天天氣晴朗。"
    ]

    # 生成嵌入
    embeddings = call_mistral_embeddings_api(texts)
    if embeddings:
        # 打印嵌入維度
        print(f" 嵌入維度：{len(embeddings[0])}")

        # 計算配對相似性
        sim_12 = calculate_similarity(embeddings[0], embeddings[1])
        sim_13 = calculate_similarity(embeddings[0], embeddings[2])
        sim_23 = calculate_similarity(embeddings[1], embeddings[2])

        # 顯示結果
        print(f"\n相似性結果：")
        print(f" 文本 1：'{texts[0]}'")
        print(f" 文本 2：'{texts[1]}'")
        print(f" 文本 3：'{texts[2]}'")
        print(f"\n文本 1 和文本 2 的相似性：{sim_12:.4f}")
        print(f" 文本 1 和文本 3 的相似性：{sim_13:.4f}")
        print(f" 文本 2 和文本 3 的相似性：{sim_23:.4f}")
```

如何運行

1. **設置 API 密鑰**：

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

2. **保存並執行**：將腳本保存（例如，作為 `embedding_example.py`）並運行它：

   ```
   python embedding_example.py
   ```

預期輸出

假設 API 請求成功，您將看到類似以下的輸出（具體值取決於返回的嵌入）：

```
嵌入維度：1024

相似性結果：
```

文本 1：'我喜歡用 Python 編程。'
文本 2：'Python 是一種很棒的編程語言。'
文本 3：'今天天氣晴朗。'

文本 1 和文本 2 的相似性：0.9200
文本 1 和文本 3 的相似性：0.6500
文本 2 和文本 3 的相似性：0.6700

說明

- **API 端點**：call_mistral_embeddings_api 函數向 https://api.mistral.ai/v1/embeddings 發送 POST 請求，傳遞一組文本和 "mistral-embed" 模型。API 返回一個包含嵌入的 JSON 响應，位於 "data" 鍵下。
- **嵌入**：每個嵌入都是一個 1024 維向量（根據 Mistral 的文檔），表示輸入文本的語義內容。嵌入標準化為 1 的範數。
- **相似性計算**：由於嵌入已標準化，兩個嵌入之間的點積（np.dot）等於它們的餘弦相似性。較高的值表示更大的語義相似性：
  - **文本 1 和文本 2**：兩者都涉及 Python 編程，因此它們的相似性應該很高（例如，約 0.92）。
  - **文本 1 和文本 3**：一個涉及編程，另一個涉及天氣，因此它們的相似性應該較低（例如，約 0.65）。
  - **文本 2 和文本 3**：相似的模式，由於主題不同，相似性較低。

額外用例

Mistral AI 的嵌入可以應用於類似性檢測之外：- **聚類**：使用 K-means 等算法對相似文本進行分組（需要 scikit-learn）。python from sklearn.cluster import KMeans kmeans = KMeans(n_clusters=2).fit(np.array(embedd print(kmeans.labels_) - **搜索**：嵌入查詢並找到最相似的文本：python query = " 告訴我有關 Python 的信息。" 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" 最相似的文本：'{most_similar}'") - **分類**：在機器學習模型中使用嵌入作為特徵。

注意事項

- **模型**："mistral-embed" 模型專門針對嵌入進行了優化。請查看文檔以獲取更新或額外模型。
- **批處理**：API 可以在一個請求中處理多個文本，從而提高效率。
- **高級應用**：對於大規模相似性搜索，請與 Faiss 或 Milvus 等向量數據庫集成。

這個示例提供了使用 Mistral AI 嵌入的實用介紹，可以通過更改端點和數據格式來適應提供的聊天完成代碼結構。