Embebidos de Mistral AI

Para proporcionar ejemplos de incrustaciones utilizando Mistral AI, puedes aprovechar su API de incrustaciones para generar representaciones numéricas (incrustaciones) de texto que capturen el significado semántico. Estas incrustaciones son útiles para tareas como la detección de similitudes, agrupamiento, clasificación o búsqueda. A continuación, demostraré cómo generar incrustaciones para un conjunto de oraciones y calcular sus similitudes utilizando el punto final de incrustaciones de Mistral AI. Este ejemplo se basa en la estructura del código proporcionado, pero se adapta específicamente para incrustaciones, según lo descrito en la documentación: Mistral AI Embeddings.

Requisitos Previos

Antes de ejecutar el ejemplo, asegúrate de tener: 1. **Clave API**: Una clave API de Mistral AI válida almacenada en la variable de entorno `MISTRAL_API_KEY`. 2. **Dependencias**: Los paquetes de Python requeridos instalados. Puedes instalarlos con: `bash pip install requests numpy`

Código de Ejemplo

Aquí tienes un script de Python completo que genera incrustaciones para tres oraciones y calcula sus similitudes por pares:

```
import os
import requests
import numpy as np

def call_mistral_embeddings_api(texts, model="mistral-embed"):
    """
    Llama a la API de incrustaciones de Mistral AI para generar incrustaciones para una lista de textos.

    Args:
        texts (list): Lista de cadenas de texto para incrustar.
        model (str): El modelo de incrustación a usar (predeterminado: "mistral-embed").

    Returns:
        list: Lista de vectores de incrustación, o None si la solicitud falla.
    """
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print("Error: La variable de entorno MISTRAL_API_KEY no está configurada.")
        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"Error de API de Mistral Embeddings: Formato de respuesta no válido: {response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Error de API de Mistral Embeddings: {e}")
        if e.response:
            print(f"Código de estado de la respuesta: {e.response.status_code}")
            print(f"Contenido de la respuesta: {e.response.text}")
        return None

def calculate_similarity(emb1, emb2):
    """
    Calcula la similitud entre dos incrustaciones utilizando el producto punto.

    Args:
        emb1 (list): Primer vector de incrustación.
        emb2 (list): Segundo vector de incrustación.

    Returns:
        float: Puntaje de similitud (producto punto, equivalente a la similitud del coseno para vectores norm
    """
```

```
    return np.dot(emb1, emb2)

if __name__ == "__main__":
    # Ejemplo de textos para incrustar
    texts = [
        "Me encanta programar en Python.",
        "Python es un excelente lenguaje de programación.",
        "El clima está soleado hoy."
    ]

    # Generar incrustaciones
    embeddings = call_mistral_embeddings_api(texts)
    if embeddings:
        # Imprimir la dimensión de la incrustación
        print(f"Dimensión de la incrustación: {len(embeddings[0])}")

        # Calcular similitudes por pares
        sim_12 = calculate_similarity(embeddings[0], embeddings[1])
        sim_13 = calculate_similarity(embeddings[0], embeddings[2])
        sim_23 = calculate_similarity(embeddings[1], embeddings[2])

        # Mostrar resultados
        print(f"\nResultados de Similitud:")
        print(f"Texto 1: '{texts[0]}'")
        print(f"Texto 2: '{texts[1]}'")
        print(f"Texto 3: '{texts[2]}'")
        print(f"\nSimilitud entre Texto 1 y Texto 2: {sim_12:.4f}")
        print(f"Similitud entre Texto 1 y Texto 3: {sim_13:.4f}")
        print(f"Similitud entre Texto 2 y Texto 3: {sim_23:.4f}")
```

Cómo Ejecutar

1. **Establecer la Clave API**:

   ```
   export MISTRAL_API_KEY="tu_clave_api_aquí"
   ```

2. **Guardar y Ejecutar**: Guarda el script (por ejemplo, como `embedding_example.py`) y ejecútalo:

   ```
   python embedding_example.py
   ```

Salida Esperada

Supongamos que la llamada a la API tiene éxito, verás una salida como esta (los valores exactos dependen de las incrustaciones devueltas):

```
Dimensión de la incrustación: 1024

Resultados de Similitud:
Texto 1: 'Me encanta programar en Python.'
Texto 2: 'Python es un excelente lenguaje de programación.'
Texto 3: 'El clima está soleado hoy.'

Similitud entre Texto 1 y Texto 2: 0.9200
Similitud entre Texto 1 y Texto 3: 0.6500
Similitud entre Texto 2 y Texto 3: 0.6700
```

Explicación

- **Punto Final de la API**: La función `call_mistral_embeddings_api` envía una solicitud POST a `https://api.mistral.ai/v1/embeddings`, pasando una lista de textos y el modelo `"mistral-embed"`. La API devuelve una respuesta JSON que contiene incrustaciones bajo la clave `"data"`.
- **Incrustaciones**: Cada incrustación es un vector de 1024 dimensiones (según la documentación de Mistral), que representa el contenido semántico del texto de entrada. Las incrustaciones se normalizan a una norma de 1.
- **Cálculo de Similitud**: Dado que las incrustaciones están normalizadas, el producto punto (`np.dot`) entre dos incrustaciones es igual a su similitud del coseno. Los valores más altos indican una mayor similitud semántica:
  - **Texto 1 y Texto 2**: Ambos tratan sobre la programación en Python, por lo que su similitud debería ser alta (por ejemplo, ~0.92).
  - **Texto 1 y Texto 3**: Uno trata sobre programación, el otro sobre el clima, por lo que su similitud debería ser menor (por ejemplo, ~0.65).
  - **Texto 2 y Texto 3**: Patrón similar, menor similitud debido a temas diferentes.

Casos de Uso Adicionales

Las incrustaciones de Mistral AI se pueden aplicar más allá de la detección de similitudes: - **Agrupamiento**: Agrupa textos similares utilizando algoritmos como K-means (requiere `scikit-learn`). `python from sklearn.cluster import KMeans kmeans = KMeans(n_clusters=2).fit(np.array(embeddings)) print(kmeans.labels_)` - **Búsqueda**: Incrusta una consulta y encuentra el texto más similar: `python query`

```
= "Cuéntame sobre 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"Texto más similar: '{most_similar}'")
```
- **Clasificación**: Usa incrustaciones como características en un modelo de aprendizaje automático.

Notas

- **Modelo**: El modelo `"mistral-embed"` está optimizado para incrustaciones. Consulta la documentación para actualizaciones o modelos adicionales.
- **Lote**: La API procesa múltiples textos en una sola llamada, mejorando la eficiencia.
- **Aplicaciones Avanzadas**: Para búsqueda de similitudes a gran escala, integra con bases de datos de vectores como Faiss o Milvus.

Este ejemplo proporciona una introducción práctica al uso de incrustaciones de Mistral AI, adaptable a la estructura del código de finalización de chat proporcionado cambiando el punto final y el formato de datos.