گزارش امتیازی پروژه بازیابی اطلاعات

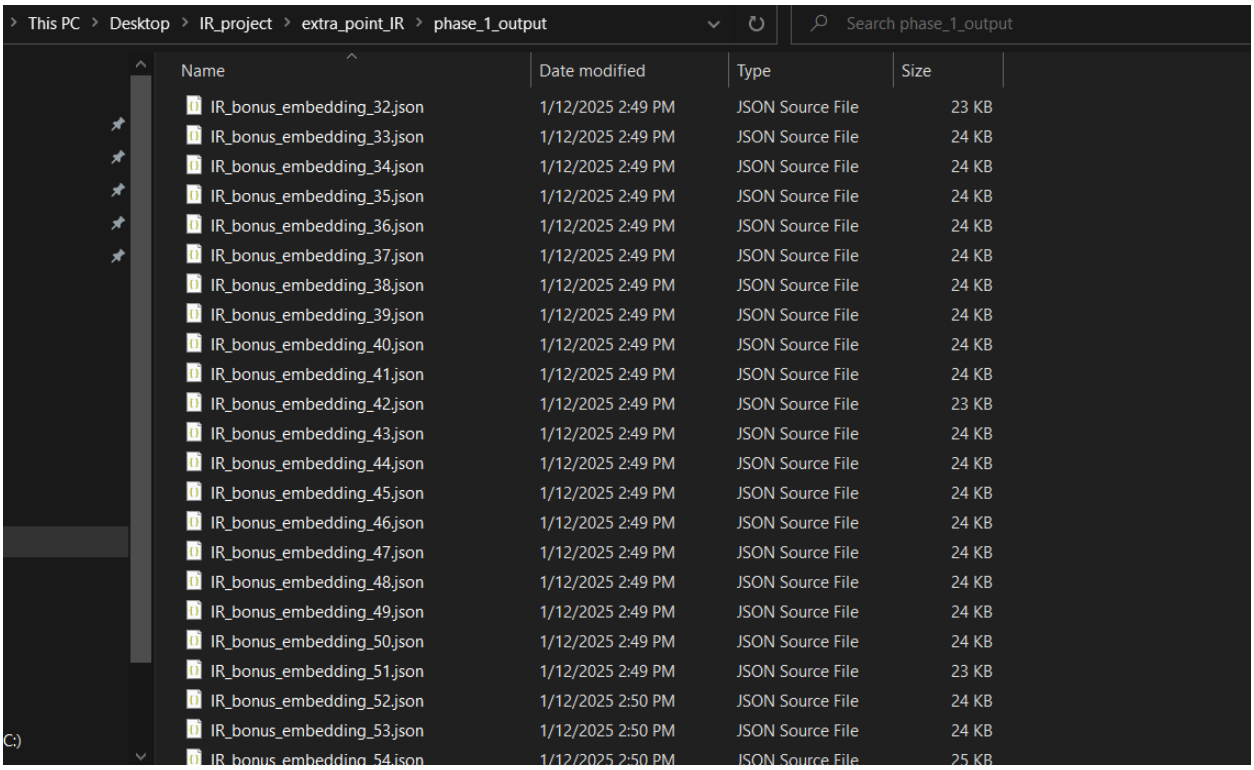
فاز ۱:

```
import json
import torch
from transformers import AutoTokenizer, AutoModel

# Load the ParsBERT model
model_path = "HooshvareLab/bert-fa-zwnj-base"
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModel.from_pretrained(model_path)


# Function to get embedding for a given text
def get_embedding(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)
    with torch.no_grad():
        outputs = model(**inputs)
    embedding = outputs.last_hidden_state[:, 0, :].squeeze().tolist()  # [CLS] token embedding
    return embedding


# Load the JSON dataset
input_file = "IR_bonus_dataset.json"
with open(input_file, "r", encoding="utf-8") as f:
    data = json.load(f)

# Process each document in the dataset and store each separately
for doc_id, doc_info in data.items():
    content = doc_info["content"]
    embedding = get_embedding(content)
    print(doc_id)

    # Prepare the data structure for the document
    output_data = {
        "doc_id": doc_id,
        "content": content,
        "embedding": embedding
    }

    # Save the result to a new JSON file named after the document ID
    output_file = f"IR_bonus_embedding_{doc_id}.json"
    with open(output_file, "w", encoding="utf-8") as f:
        json.dump(output_data, f, ensure_ascii=False, indent=4)

    print(f"Embedding for {doc_id} saved to {output_file}")
```

با استفاده از این تکه کد، embedding ها را تعبیه میکنیم و در پوشه‌ی پروژه‌ی خود به آن ها دسترسی داریم.


```
{
    "doc_id": "6251",
    "content": "پس از مدتی، اندک خط تولید محصول این شرکت، از طریق مدل کیا اسپورتیج، شرکت گسترش یافت"
    "embedding": [
        -1.6038877964019775,
        -0.775762677192688,
        -1.0039913654327393,
        0.46896013617515564,
        -0.32982879877090454,
        -0.241697758436203,
        -0.5669456124305725,
        -0.35902947187423706,
        -0.05825961008667946,
        0.8547347187995911,
        -0.3960050642490387,
        -0.5634655356407166,
        -1.4946902990341187,
        0.6571413278579712,
        1.022910475730896,
        -1.796049952507019,
        2.4124250411987305,
        0.2235233187675476,
        0.7615768313407898,
        1.2572202682495117,
        -1.0148481130599976,
        0.7515184879302979,
        0.8440330624580383,
```


بخشی از یک امبدینگ



This PC > Desktop > IR_project > extra_point_IR > phase_1_output

Search phase_1_output

Name	Date modified	Type	Size
IR_bonus_embedding_32.json	1/12/2025 2:49 PM	JSON Source File	23 KB
IR_bonus_embedding_33.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_34.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_35.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_36.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_37.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_38.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_39.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_40.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_41.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_42.json	1/12/2025 2:49 PM	JSON Source File	23 KB
IR_bonus_embedding_43.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_44.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_45.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_46.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_47.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_48.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_49.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_50.json	1/12/2025 2:49 PM	JSON Source File	24 KB
IR_bonus_embedding_51.json	1/12/2025 2:49 PM	JSON Source File	23 KB
IR_bonus_embedding_52.json	1/12/2025 2:50 PM	JSON Source File	24 KB
IR_bonus_embedding_53.json	1/12/2025 2:50 PM	JSON Source File	24 KB
IR_bonus_embedding_54.json	1/12/2025 2:50 PM	JSON Source File	25 KB

C:)

تعدادی از این فایل ها در پوشه

فاز ۲:

```
import json
import os
import numpy as np  # For computing the Euclidean norm

# Path to directory containing document embeddings
embedding_dir = "C:\\Users\\Parsa\\Desktop\\IR_project\\extra_point_IR\\phase_1_output"

# Function to check the Euclidean norm of embeddings
def check_embedding_norms(embedding_dir):
    embedding_norms = {}

    # Iterate over document embeddings one by one
    for file_name in os.listdir(embedding_dir):
        file_path = os.path.join(embedding_dir, file_name)

        # Load document embedding from JSON file
        with open(file_path, 'r', encoding='utf-8') as f:
            doc_data = json.load(f)
```

```
        # Assuming embeddings are stored in a list of dictionaries
        if isinstance(doc_data, list):
            # If doc_data is a list, inspect its first element
            doc_embedding = doc_data[0]["embedding"]
        else:
            doc_embedding = doc_data["embedding"]

    # Compute the Euclidean norm (magnitude) of the embedding vector
    embedding_norm = np.linalg.norm(doc_embedding)  # Euclidean norm

    # Print the norm for each document
    print(f"Document: {file_name}, Norm: {embedding_norm:.4f}")

# Run the check
check_embedding_norms(embedding_dir)
```

با استفاده از این تکه کد، اندازه هر بردار را حساب میکنیم و مشاهده میکنیم که همگی آن ها یک اندازه هستند. پس میتوانا‌ز شباهت کسینوسی یا حتی از فاصله‌ی بین دو بردار برای شبیه بودن نیز استفاده کرد. که من بازهمم ترجیح دادم کع از شباهت کسینوسی استفاده کنم. اگر اندازه بردار ها یکسان نباشد، استفاده از فاصله اقلیدسی اشتباه و غلط است!

enter the query: نیروهای مسلح جمهوری اسلامی ایران

Time taken to retrieve top 5 documents: 14.4443 seconds

Top similar documents:

۱. File: IR_bonus_embedding_6094.json - Similarity: 0.6379

Content: تیم ملی فوتبال زنان ایالات متحده‌ٔمریکا (به انگلیسی: United States women's national soccer team) نماینده‌ٔ زنان کشور ایالات متحده‌ٔمریکا در رده‌ٔ ملی است؛ که زیرنظر فدراسیون فوتبال‌ٔمریکا قرار دارد. ...

۲. File: IR_bonus_embedding_7573.json - Similarity: 0.6368

Content: عملیات الی بیت‌المقدس(معنا: به‌سوی بیت‌المقدس)، یکی از بزرگترین عملیات های نیروهای مسلح جمهوری اسلامی ایران در جنگ ایران و عراق محسوب می‌شود. نیروهای ایرانی با توان ۱۳۰ هزار سرباز در ساعت ۳۰ دقیقه بام...

۳. File: IR_bonus_embedding_7318.json - Similarity: 0.6137

Content: ستاد اجرایی فرمان امام سازمان حکومتی، شبه‌دولتی ایرانی است که تحت کنترل رهبر جمهوری اسلامی فعالیت می‌کند. ستاد اجرایی در سال ۱۳۶۸ به فرمان روح‌الله خمینی تأسیس شد. در سالهای نخست وظیفه‌ی شناسایی و ...

۴. File: IR_bonus_embedding_2177.json - Similarity: 0.6127

Content: مبارزات مسالمت‌آمیز زنان ایران برای رفع تبعیض و برابری حقوقی با مردان با تکیه بر موازین بین‌المللی حقوق بشر یکی از صفحات درخشان تاریخ مبارزات جهانی زنان برای برابری حقوق با مردان است. ابتکار و نوآور...

۵. File: IR_bonus_embedding_5518.json - Similarity: 0.6123

Content: باشگاه فوتبال اینترناتسیوناله میلانو (ایتالیایی: Football Club Internazionale Milano تلفظ ایتالیایی: [ˌinternattsjoˈnaːle]) که معمولاً با عنوان اینتر میلان یا اینتر شناخته می‌شود، یک باشگاه حرفه‌ای ف...

enter the query: قهرمانی در لیگ باشگاه‌های انگلیس

Time taken to retrieve top 5 documents: 10.6698 seconds

Top similar documents:

1. File: IR_bonus_embedding_3332.json - Similarity: 0.7417

Content: قهرمانی المپیک تنها عنوان بین‌المللی بود که تیم ملی فوتبال زنان آلمان تا سال ۲۰۱۶ نتوانسته بود به‌آن دست پیدا کند. فوتبال زنان برای اولین بار در سال ۱۹۹۶ در بازی‌های المپیک آغاز به کار کرد و بتینا ...

2. File: IR_bonus_embedding_2374.json - Similarity: 0.6628

Content: تیم ملی ایران در بازی‌های آسیایی ۱۹۹۸ بانکوک در اولین مرحله گروهی، با تیم‌های قزاقستان و لائوس در گروه H این مسابقات قرار گرفت. پس از پیروزی ۰–۲ در مقابل قزاقستان، تیم ملی ایران در دومین بازی با نتیج...

3. File: IR_bonus_embedding_600.json - Similarity: 0.6549

Content: گرندپری اروپا رقابتی است که سالانه در یکی از پیست‌های اروپایی فرمول یک برگزار می‌شد.این رقابت از سال ۱۹۸۳ در تقویم فرمول یک جا داشته در سال اول این رقابت در پیست برندز هچ بریتانیا برگزار شد و طی مدت‌ه...

File: IR_bonus_embedding_6244.json - Similarity: 0.6546 .4

Content: جام جهانی فوتبال ۲۰۱۴ بیستمین دورهٔ جام جهانی فوتبال بود که از ۱۲ ژوئن ۲۰۱۴ تا ۱۳ ژوئیه ۲۰۱۴ در کشور برزیل برگزار شد و در نهایت با قهرمانی تیم ملی فوتبال آلمان پایان گرفت. در دیدار پایانی ژرمن‌ها م...

File: IR_bonus_embedding_5.json - Similarity: 0.6370 .5

Content: بازگشت بازیکن سابق آرسنال یعنی جورج گراهام به عنوان مربی در سال ۱۹۸۶ میلادی دوره سوم موفقیت را برای این باشگاه به ارمغان آورد. آرسنال در فصل ۸۷–۱۹۸۶ قهرمان جام اتحادیه شد. همچنین آرسنال قهرمان لیگ...

دو نمونه‌ی ابتدایی از کوئری ها در بالا قابل مشاهده است.

کد و توضیحات مروبط به فاز دوم:

```
# Load the ParsBERT model
model_path = "HooshvareLab/bert-fa-zwnj-base"
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModel.from_pretrained(model_path)


1 usage
def get_embedding(text):
    """Generate embedding for a given text using ParsBERT."""
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)
    with torch.no_grad():
        outputs = model(**inputs)
    # Extract the [CLS] token embedding (the first token's representation)
    embedding = outputs.last_hidden_state[:, 0, :].squeeze().tolist()
    return embedding
```

این تابع همانند تابعی است که برای امبد کردن سندها استفاده کردیم. یک متن (در اینجا کوئری) را دریافت میکند و بردار مربوط به آن را تولید میکند.

```python
def cosine_similarity(a, b):
    """Calculate cosine similarity between two vectors."""
    dot_product = np.dot(a, b)  # Dot product of a and b
    norm_a = np.linalg.norm(a)  # L2 norm (magnitude) of vector a
    norm_b = np.linalg.norm(b)  # L2 norm (magnitude) of vector b
    if norm_a == 0 or norm_b == 0:  # Check if either vector is zero to avoid division by zero
        return 0.0
    return dot_product / (norm_a * norm_b)
```

این تابع نیز برای محاسبه‌ی شباهت کسینوسی بین دو سند استفاده میگردد به این صورت که ضرب داخلی دو بردار را تقسیم بر ضرب اندازه‌ی آن ها میکند.

```python
def get_top_k_similar_docs(query, k, embedding_dir):
    # Generate embedding for the query
    query_embedding = get_embedding(query)

    # Store similarities
    similarities = []

    # Iterate over document embeddings one by one in the specified directory
    for file_name in os.listdir(embedding_dir):
        file_path = os.path.join(embedding_dir, file_name)

        # Ensure it's a JSON file
        if file_name.endswith(".json"):
            # Load document embedding from JSON file
            with open(file_path, 'r', encoding='utf-8') as f:
                doc_data = json.load(f)

            # Handle cases where doc_data is a list or dictionary
            if isinstance(doc_data, list):
                # If the data is a list (containing a single dictionary with embedding)
                doc_embedding = doc_data[0]["embedding"]
                doc_content = doc_data[0].get("content", "No content")
            else:
                # If the data is a single dictionary
```

```python
            # If the data is a list (containing a single dictionary with embeddin
            doc_embedding = doc_data[0]["embedding"]
            doc_content = doc_data[0].get("content", "No content")
        else:
            # If the data is a single dictionary
            doc_embedding = doc_data["embedding"]
            doc_content = doc_data.get("content", "No content")

        # Compute similarity
        similarity = cosine_similarity(query_embedding, doc_embedding)

        # Store result as (file_name, similarity, content)
        similarities.append((file_name, similarity, doc_content))

    # Sort by similarity in descending order
    similarities.sort(key=lambda x: x[1], reverse=True)

    # Return top k results
    return similarities[:k]


# Example usage
query = input("enter the query: ")
k = 5
embedding_dir = "C:\\Users\\Parsa\\Desktop\\IR_project\\extra_point_IR\\phase_1_output"
```

این تابع نیز ابتدا وارد پوشه‌ی شامل بردارهای مربوط به اسناد میشود و سپس با توجه به فرمت فایل شباهت بین فایل‌ها را محاشبه میکند (به دلیل اینکه در ابتدا فرمت دیگری را استفاده کرده بودم این کد را نوشتم و تصمیم گرفتم کد درست را به آن اضافه کنم).

در نهایت هم لیست را مرتب میکنیم و k داکیومنت اول را برمیگردانیم.

```python
# Example usage
query = input("enter the query: ")
k = 5
embedding_dir = "C:\\Users\\Parsa\\Desktop\\IR_project\\extra_point_IR\\phase_1_output"

# Measure the time it takes from getting the query to showing results
start_time = time.time()

top_k_docs = get_top_k_similar_docs(query, k, embedding_dir)

# Calculate the elapsed time
elapsed_time = time.time() - start_time

# Print the results
print(f"Time taken to retrieve top {k} documents: {elapsed_time:.4f} seconds\n")
print("Top similar documents:")
for rank, (file_name, score, content) in enumerate(top_k_docs, start=1):
    print(f"{rank}. File: {file_name} - Similarity: {score:.4f}")
    print(f"   Content: {content[:200]}...")  # Show the first 200 characters of the content
    print()
```

باقی کار تنها صدا زدن توابع و نشان دادن خروجی است.

فاز ۳:

کد زیر را اجرا میکنیم:kابتدا برای پیدا کردن مقدار

```python
import numpy as np
import json
import os
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score


def read_embeddings_from_files(embedding_dir):
    embeddings = []
    for file_name in os.listdir(embedding_dir):
        file_path = os.path.join(embedding_dir, file_name)
        if file_name.endswith(".json"):
            with open(file_path, 'r', encoding='utf-8') as f:
                doc_data = json.load(f)
                if isinstance(doc_data, list):
```

```
            doc_embedding = doc_data[0]["embedding"]
        else:
            doc_embedding = doc_data["embedding"]
        embeddings.append(doc_embedding)
    return np.array(embeddings)



embedding_dir = "C:\\Users\\Parsa\\Desktop\\IR_project\\extra_point_IR\\phase_1_output"

embeddings_matrix = read_embeddings_from_files(embedding_dir)



def print_kmeans_analysis(data, max_k=10):
    print("KMeans Clustering Analysis\n")
    print("{:<8} {:<15}".format("k", "Inertia"))
    print("-" * 38)

    for k in range(2, max_k + 1):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)

        inertia = kmeans.inertia_

        silhouette_avg = silhouette_score(data, kmeans.labels_)

        print("{:<8} {:<15.4f}".format(k, inertia))


print_kmeans_analysis(embeddings_matrix)
```

که خروجی آن به صورت رو به رو میباشد:

```
k        Inertia
--------------------------
2        3319040.0717
3        3177697.3287
4        3057497.7722
5        2995190.2151
6        2954558.7285
7        2910194.0680
8        2877724.3732
9        2844881.9997
10       2815369.4592
```

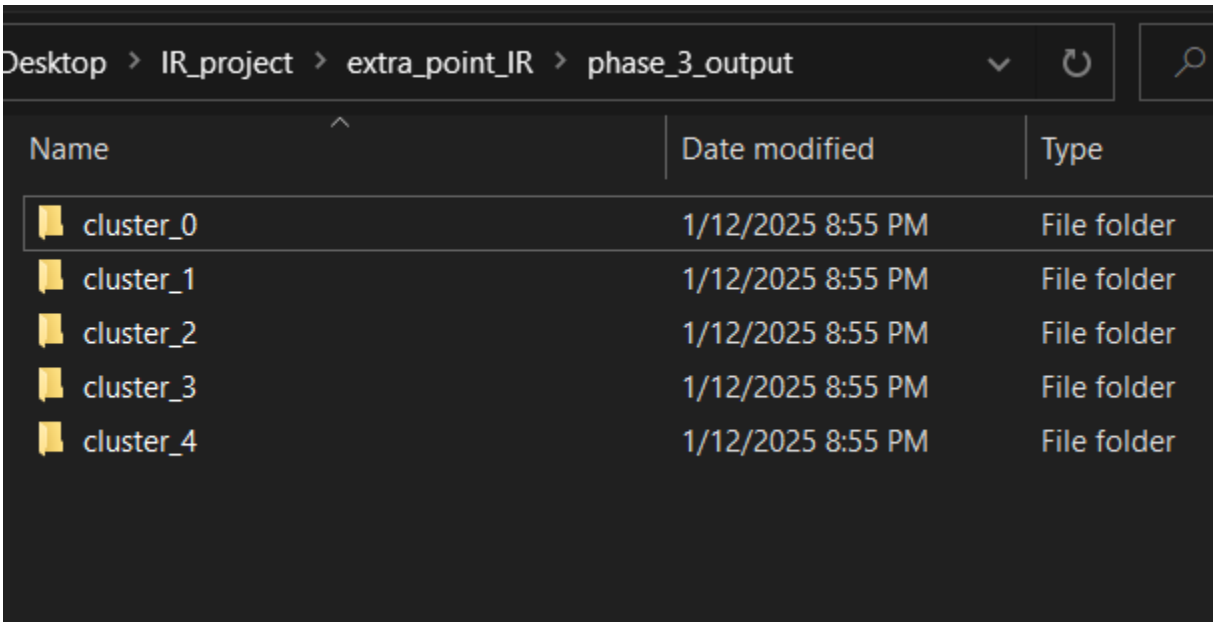
از آنجا که تغییرات، از k = 5 به بعد کمتر میشود و k=5 به اصطلاح elbow است، این مقدار را انتخاب میکنیم.

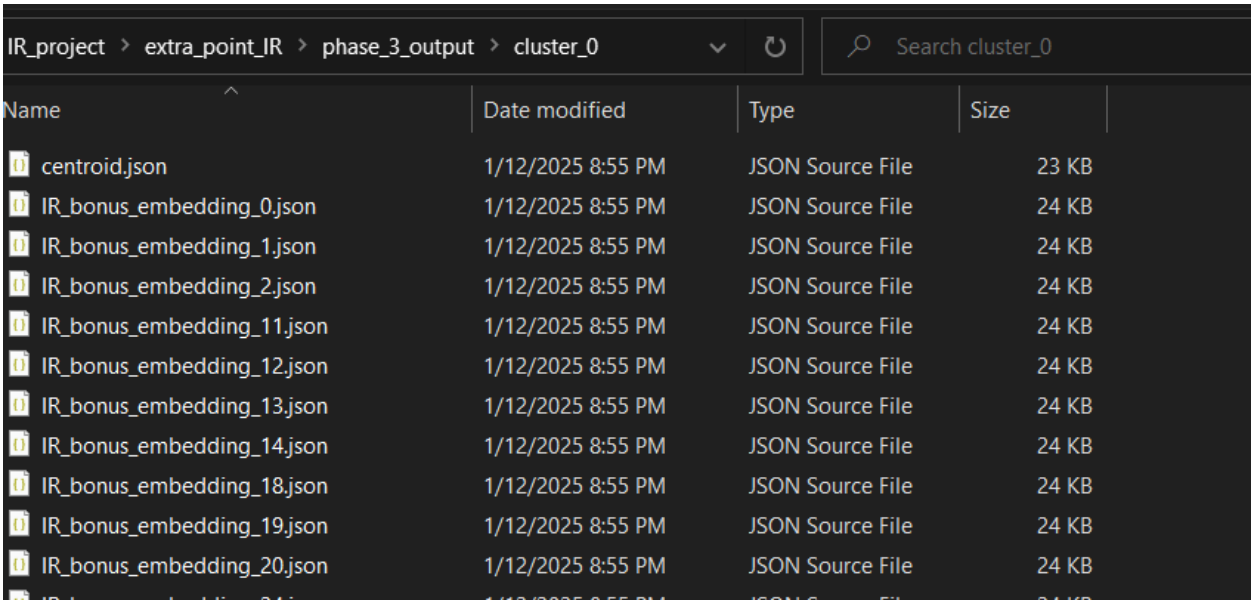
حال با استفاده از کد زیر، اسناد را دسته بندی میکنیم:

```
import numpy as np
import json
from transformers import AutoTokenizer, AutoModel
import torch
import os
import shutil
from sklearn.cluster import KMeans


def read_embeddings_from_files(embedding_dir):
    """
    Read embeddings from JSON files in the specified directory.

    Args:
        embedding_dir (str): Path to the directory containing embedding JSON files.

    Returns:
        list: A list of embeddings.
        list: A list of corresponding file paths.
    """
    embeddings = []
    file_paths = []

    for file_name in os.listdir(embedding_dir):
        file_path = os.path.join(embedding_dir, file_name)

        if file_name.endswith(".json"):
            with open(file_path, 'r', encoding='utf-8') as f:
                doc_data = json.load(f)

            if isinstance(doc_data, list):
                doc_embedding = doc_data[0]["embedding"]
            else:
                doc_embedding = doc_data["embedding"]

            embeddings.append(doc_embedding)
            file_paths.append(file_path)

    return embeddings, file_paths

def save_clusters(kmeans, file_paths, output_dir):
    """
    Save clusters in separate subdirectories along with centroid files.
```

```
    Args:
        kmeans (KMeans): Trained KMeans model.
        file_paths (list): List of file paths corresponding to embeddings.
        output_dir (str): Directory to store the cluster output.
    """
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    for cluster_idx in range(kmeans.n_clusters):
        cluster_dir = os.path.join(output_dir, f"cluster_{cluster_idx}")
        os.makedirs(cluster_dir, exist_ok=True)


        centroid_path = os.path.join(cluster_dir, "centroid.json")
        with open(centroid_path, 'w', encoding='utf-8') as f:
            json.dump({"centroid": kmeans.cluster_centers_[cluster_idx].tolist()}, f, ensure_ascii=False, indent=4)


    for i, file_path in enumerate(file_paths):
        cluster_idx = kmeans.labels_[i]
        cluster_dir = os.path.join(output_dir, f"cluster_{cluster_idx}")
        shutil.copy(file_path, cluster_dir)

def main():
    embedding_dir = "C:\\Users\\Parsa\\Desktop\\IR_project\\extra_point_IR\\phase_1_output"
    output_dir = "phase_3_output"
    num_clusters = 5


    embeddings, file_paths = read_embeddings_from_files(embedding_dir)
    embeddings = np.array(embeddings)


    kmeans = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)
    kmeans.fit(embeddings)


    save_clusters(kmeans, file_paths, output_dir)

    print(f"Clustering completed. Results saved in '{output_dir}'.")

if __name__ == "__main__":
    main()
```

نتیجه:





حال کافیست هر کوئری را با مرکز دسته مقایسه کنیم، دسته‌ی مربوطه را بیابیم و به عبارتی تکه کد زیر را اجرا کنیم:
این کد خروجی زیر را تولید کرد:

Enter the query: قهرمانی در لیگ باشگاه‌های انگلیس

Time taken to retrieve top 5 documents: 6.7424 seconds

Top similar documents:

1. Document ID: IR_bonus_embedding_3332.json

Similarity Score: 0.7417

Cluster: cluster_0

Content: قهرمانی المپیک تنها عنوان بین‌المللی بود که تیم ملی فوتبال زنان آلمان تا سال ۲۰۱۶ نتوانسته بود به آن دست پیدا کند. فوتبال زنان برای اولین بار در سال ۱۹۹۶ در بازی‌های المپیک اتلانتا به کار کرد و بتینا ...

2. Document ID: IR_bonus_embedding_2374.json

Similarity Score: 0.6628

Cluster: cluster_0

Content: تیم ملی ایران در بازی‌های آسیایی ۱۹۹۸ بانکوک در اولین مرحله گروهی، با تیم‌های قزاقستان و لائوس در گروه H این مسابقات قرار گرفت. پس از پیروزی ۰–۲ در مقابل قزاقستان، تیم ملی ایران در دومین بازی با نتیج...

3. Document ID: IR_bonus_embedding_600.json

Similarity Score: 0.6549

Cluster: cluster_0

Content: گرندپری اروپا رقابتی است که سالانه در یکی از پیست‌های اروپایی فرمول یک برگزار می‌شد.این رقابت از سال ۱۹۸۳ در تقویم فرمول یک جا داشته در سال اول این رقابت در پیست برندز هچ بریتانیا برگزار شد و طی مدت‌ه...

4. Document ID: IR_bonus_embedding_6244.json

Similarity Score: 0.6546

Cluster: cluster_0

Content: جام جهانی فوتبال ۲۰۱۴ بیستمین دورهٔ جام جهانی فوتبال بود که از ۱۲ ژوئن ۲۰۱۴ تا ۱۳ ژوئیه ۲۰۱۴ در کشور برزیل برگزار شد و در نهایت با قهرمانی تیم ملی فوتبال آلمان پایان گرفت. در دیدار پایانی ژرمن‌ها م...

5. Document ID: IR_bonus_embedding_3327.json

Similarity Score: 0.6363

Cluster: cluster_0

Content: تیم ملی فوتبال زنان□لمان در جام جهانی فوتبال زنان ۲۰۰۳ در کشور ایالات متحده□مریکا با کانادا، ژاپن و□رژانتین هم‌گروه شد و پس از پیروزی در هر سه بازی گروهی خود تیم روسیه را در یک‌چهارم نهایی با نت...

همانطور که مشاهده میگردد مرتبه زمانی تقریبا نصف (۳/۴) شده است که دلیلش همان پردازش نکردن تمامی سندها و انتخاب یک زیر مجموعه از آن ها و پردازش آن است.

توضیحات مربوط به کد:

```
import numpy as np
import json
from transformers import AutoTokenizer, AutoModel
import torch
import os
import time

model_path = "HooshvareLab/bert-fa-zwnj-base"
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModel.from_pretrained(model_path)

1 usage
def get_embedding(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)
    with torch.no_grad():
        outputs = model(**inputs)
    embedding = outputs.last_hidden_state[:, 0, :].squeeze().tolist()
    return embedding

2 usages
def cosine_similarity(a, b):
    dot_product = np.dot(a, b)
    norm_a = np.linalg.norm(a)
    norm_b = np.linalg.norm(b)
    if norm_a == 0 or norm_b == 0:
        return 0.0
    return dot_product / (norm_a * norm_b)
```

این قسمت، سراسر مانند قسمت قبلی است که در آن امبدینگ یک متن به دست می آید و یک تابع برای محاسبه‌ی شباهت کسینوسی تعریف میگردد.

```python
def get_top_similar_docs(query_embedding, cluster_dir, k=5):
    cluster_similarities = []

    for cluster_name in os.listdir(cluster_dir):
        cluster_path = os.path.join(cluster_dir, cluster_name)
        if os.path.isdir(cluster_path):
            centroid_path = os.path.join(cluster_path, "centroid.json")

            if os.path.exists(centroid_path):
                with open(centroid_path, 'r', encoding='utf-8') as f:
                    centroid_data = json.load(f)
                    centroid_embedding = np.array(centroid_data["centroid"])

                similarity_to_centroid = cosine_similarity(query_embedding, centroid_embedding)

                cluster_similarities.append((cluster_name, similarity_to_centroid, cluster_path))

    cluster_similarities.sort(key=lambda x: x[1], reverse=True)

    best_cluster_name, best_similarity, best_cluster_path = cluster_similarities[0]

    top_similar_docs = []
    cluster_files = [f for f in os.listdir(best_cluster_path) if f.endswith(".json")]

    for file_name in cluster_files:
        file_path = os.path.join(best_cluster_path, file_name)

        if file_name == "centroid.json":
            continue
```

```python
        with open(file_path, 'r', encoding='utf-8') as f:
            try:
                doc_data = json.load(f)

                if isinstance(doc_data, list):
                    doc_data = doc_data[0]

                doc_embedding = doc_data["embedding"]
            except KeyError:
                continue

            similarity_to_doc = cosine_similarity(query_embedding, doc_embedding)
            doc_content = doc_data.get("content", "No content")
            top_similar_docs.append((file_name, similarity_to_doc, doc_content, best_cluster_name))

    top_similar_docs.sort(key=lambda x: x[1], reverse=True)
    return top_similar_docs[:k]
```

این تابع هم تا حدود زیادی شبیه به قسمت قبل است، تنها تفاوت آن این است که ابتدا، کوئری را با هر مرکز دسته مقایسه میکند و نتیجه را در cluster_similarities ذخیره میکند. حال با سورت کردن این پنج عدد و انتخاب بزرگترین، به خوشه‌ی مربوط به کوئری دست پیدا میکنیم.

حال در ادامه تنها امتیاز را با اعضای خوشه محاسبه میکنیم و نتیجه را مانند قبل گزارش میکنیم.