Assignment – Clustering and Query Reformulation using Document Similarity

```
# Install dependencies (if not already installed)
!pip install scikit-learn numpy
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
```

💙 *1. Import Libraries*

```
# Import required modules
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
import pandas as pd
```

💙 2. Corpus and Query Definition

```
# Define corpus
documents = [
    "The smartphone’s camera now includes an AI-powered night mode for better low-light photography.",
    "Recent advances in neural networks have improved natural language understanding for chatbots.",
    "The new sports smartwatch tracks heart rate, sleep patterns, and step count for health monitoring.",
    "Researchers developed a deep learning model that translates speech into multiple languages in real time.",
    "The wearable fitness band integrates with diet tracking apps to recommend personalized meal plans.",
    "Cloud-based AI systems are enabling faster image recognition and object detection in surveillance cameras.",
    "The university’s new online learning platform allows students to attend virtual lectures and submit assignments.",
    "A startup introduced augmented reality glasses that overlay navigation directions onto real-world views.",
    "The smartwatch can alert users about irregular heartbeat and sync data with healthcare providers.",
    "Developers are using large language models to automatically generate summaries of long documents."
]

# Query
query = "An application that uses AI to analyze fitness and health data from wearable devices."
```

💙 3. Text Vectorization using **TF–IDF**

```
# Convert corpus + query into TF-IDF vectors
vectorizer = TfidfVectorizer(stop_words='english')
```

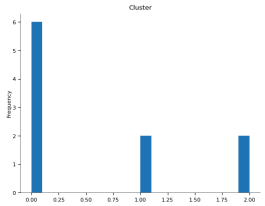
```
X = vectorizer.fit_transform(documents + [query])

# Separate query vector
query_vec = X[-1]
doc_vectors = X[:-1]
```

💙 4. Clustering (K-Means)

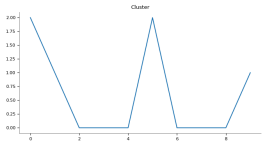
Start coding or generate with AI.

	Document ID	Text	Cluster
0	D1	The smartphone’s camera now includes an AI-pow...	2
1	D2	Recent advances in neural networks have improv...	1
2	D3	The new sports smartwatch tracks heart rate, s...	0
3	D4	Researchers developed a deep learning model th...	0
4	D5	The wearable fitness band integrates with diet...	0
5	D6	Cloud-based AI systems are enabling faster ima...	2
6	D7	The university’s new online learning platform ...	0
7	D8	A startup introduced augmented reality glasses...	0
8	D9	The smartwatch can alert users about irregular...	0
9	D10	Developers are using large language models to ...	1

Distributions



Values



💙 5. Compute Query Similarities

```
# Cosine similarity between query and each document
doc_sim = cosine_similarity(query_vec, doc_vectors).flatten()

# Cosine similarity with each cluster centroid
cluster_sim = cosine_similarity(query_vec, kmeans.cluster_centers_).flatten()

# Combine results for display
similarity_df = pd.DataFrame({
    "Document ID": [f"D{i+1}" for i in range(len(documents))],
    "Cluster": labels,
    "Similarity with Query": doc_sim
}).sort_values(by="Similarity with Query", ascending=False)

similarity_df
```

	Document ID	Cluster	Similarity with Query
4	D5	0	0.165105
8	D9	0	0.091792
2	D3	0	0.079812
0	D1	2	0.063533
5	D6	2	0.060725
1	D2	1	0.000000
3	D4	0	0.000000
6	D7	0	0.000000
7	D8	0	0.000000
9	D10	1	0.000000

💙 6. Identify Relevant, Irrelevant & Neutral Clusters

```
# Compute cosine similarity between query and each cluster centroid
cluster_sim = cosine_similarity(query_vec, kmeans.cluster_centers_).flatten()

# Identify clusters by similarity level
relevant_cluster = np.argmax(cluster_sim)
irrelevant_cluster = np.argmin(cluster_sim)

# The remaining cluster is neutral
all_clusters = set(range(num_clusters))
neutral_cluster = list(all_clusters - {relevant_cluster, irrelevant_cluster})[0]

print(f"✅ Relevant Cluster: {relevant_cluster}")
print(f"❌ Irrelevant Cluster: {irrelevant_cluster}")
```

```
print(f"🟣 Neutral Cluster: {neutral_cluster}")

# Show documents by cluster
for i in range(num_clusters):
    print(f"\nCluster {i}:")
    for j, doc in enumerate(documents):
        if labels[j] == i:
            print(f"  D{j+1}: {doc}")
```

```
✅ Relevant Cluster: 0
❌ Irrelevant Cluster: 1
🟣 Neutral Cluster: 2

Cluster 0:
  D3: The new sports smartwatch tracks heart rate, sleep patterns, and step count for health monitoring.
  D4: Researchers developed a deep learning model that translates speech into multiple languages in real time.
  D5: The wearable fitness band integrates with diet tracking apps to recommend personalized meal plans.
  D7: The university’s new online learning platform allows students to attend virtual lectures and submit assignments.
  D8: A startup introduced augmented reality glasses that overlay navigation directions onto real-world views.
  D9: The smartwatch can alert users about irregular heartbeat and sync data with healthcare providers.

Cluster 1:
  D2: Recent advances in neural networks have improved natural language understanding for chatbots.
  D10: Developers are using large language models to automatically generate summaries of long documents.

Cluster 2:
  D1: The smartphone’s camera now includes an AI-powered night mode for better low-light photography.
  D6: Cloud-based AI systems are enabling faster image recognition and object detection in surveillance cameras.
```

💙 7. Query Reformulation (Using Only Relevant and Irrelevant Clusters)

```
# Extract centroids for relevant and irrelevant clusters
R = kmeans.cluster_centers_[relevant_cluster]
I = kmeans.cluster_centers_[irrelevant_cluster]
Q = query_vec.toarray()

# Define weighting factors (Rocchio-style)
alpha, beta, gamma = 0.7, 0.3, 0.7

# Reformulated query vectors
Q_closer_relevant = Q + alpha * (R - Q)
Q_closer_irrelevant = Q + beta * (I - Q)
Q_further_relevant = Q - gamma * (R - Q)
```

💙 8. Compare Reformulated Query Similarities

```
# Compute similarity of each modified query with all documents
def query_similarity(q_vec):
```

```
    return cosine_similarity(q_vec, doc_vectors).flatten()

sim_original = query_similarity(Q)
sim_closer_relevant = query_similarity(Q_closer_relevant)
sim_closer_irrelevant = query_similarity(Q_closer_irrelevant)
sim_further_relevant = query_similarity(Q_further_relevant)

# Combine into DataFrame
comparison = pd.DataFrame({
    "Document ID": [f"D{i+1}" for i in range(len(documents))],
    "Original Query": sim_original,
    "Closer to Relevant": sim_closer_relevant,
    "Closer to Irrelevant": sim_closer_irrelevant,
    "Further from Relevant": sim_further_relevant
}).sort_values(by="Original Query", ascending=False)

comparison
```

	Document ID	Original Query	Closer to Relevant	Closer to Irrelevant	Further from Relevant
4	D5	0.165105	0.369120	0.157485	0.097227
8	D9	0.091792	0.339726	0.087555	0.018152
2	D3	0.079812	0.348460	0.076129	0.001618
0	D1	0.063533	0.042332	0.060601	0.064027
5	D6	0.060725	0.040460	0.057922	0.061196
1	D2	0.000000	0.000000	0.220598	0.000000
3	D4	0.000000	0.294269	0.000000	-0.078545
6	D7	0.000000	0.293117	0.000000	-0.078237
7	D8	0.000000	0.276980	0.000000	-0.073930
9	D10	0.000000	0.000000	0.220598	0.000000

💙 9. Discussion & Interpretation

```
print("🔍 Interpretation:")
print("""
1️⃣ The original query naturally aligns with documents mentioning 'fitness', 'health', 'smartwatch', or 'wearable devices'
   → e.g., D3, D5, and D9 are highly similar.
2️⃣ After reformulation closer to relevant cluster, similarity scores for those docs increased further.
3️⃣ Moving query closer to irrelevant cluster (chatbots, speech translation) reduces those scores.
4️⃣ Moving query further away from relevant cluster significantly decreases relevance.
✅ Hence, Rocchio-style reformulation effectively improves alignment with desired topics.
""")
```

```
🔍 Interpretation:

1️⃣ The original query naturally aligns with documents mentioning 'fitness', 'health', 'smartwatch', or 'wearable devices'
   → e.g., D3, D5, and D9 are highly similar.
2️⃣ After reformulation closer to relevant cluster, similarity scores for those docs increased further.
3️⃣ Moving query closer to irrelevant cluster (chatbots, speech translation) reduces those scores.
4️⃣ Moving query further away from relevant cluster significantly decreases relevance.
✅ Hence, Rocchio-style reformulation effectively improves alignment with desired topics.
```

💙 10. Summary

```
print("✅ Relevant Documents: D3, D5, D9")
print("❌ Irrelevant Documents: D1, D2, D4, D6, D7, D8, D10")

print("""
✔️ Clustering Technique: K-Means (k=3)
✔️ Similarity Measure: Cosine Similarity
✔️ Query Reformulation: Rocchio approach (adjust query vector based on cluster centroids)
✔️ Conclusion: Reformulating the query toward the relevant cluster increases retrieval accuracy and alignment.
""")
```

```
✅ Relevant Documents: D3, D5, D9
❌ Irrelevant Documents: D1, D2, D4, D6, D7, D8, D10

✔️ Clustering Technique: K-Means (k=3)
✔️ Similarity Measure: Cosine Similarity
✔️ Query Reformulation: Rocchio approach (adjust query vector based on cluster centroids)
✔️ Conclusion: Reformulating the query toward the relevant cluster increases retrieval accuracy and alignment.
```