WEEK-8

Implement Matrix Decomposition and LSI for a standard dataset.

Matrix Decomposition

Matrix decomposition (or matrix factorization) is a mathematical technique used to break down a large matrix into smaller, simpler matrices.

- In text mining, we typically start with a document-term matrix (DTM) or TF-IDF matrix where:
    - Rows = documents
    - Columns = terms (words)
    - Values = frequency or importance of terms in documents

Latent Semantic Indexing (LSI)

Latent Semantic Indexing (also called Latent Semantic Analysis, LSA) is a technique in information retrieval and natural language processing that uses SVD on the term-document matrix.

Idea behind LSI

- Natural language has synonyms (different words with similar meaning) and polysemy (same word with multiple meanings).
- A raw term-document matrix treats each word independently, which misses semantic relationships.
- LSI reduces the matrix dimensions to capture hidden (latent) semantic structures in text.

Process of LSI

1. Construct a TF-IDF matrix from the dataset.
2. Apply Truncated SVD to decompose into lower-rank matrices.
3. Represent documents and terms in this reduced semantic space.
4. Use cosine similarity or other metrics to find similarity between documents/queries**.**

Singular Value Decomposition (SVD)

- SVD is a mathematical technique that breaks a large matrix into three smaller matrices.
- For a document-term matrix (like TF-IDF), SVD helps find patterns/relationships between terms and documents.
- It identifies the most important concepts (latent topics) in the data

Note: Truncated SVD is a way to compress large text data into fewer hidden topics, keeping the essential meaning but reducing complexity.

CODE

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Step 1: Load dataset (subset for speed)
categories = ['sci.space', 'rec.sport.hockey', 'comp.graphics']
newsgroups = fetch_20newsgroups(subset='train', categories=categories, remove=('headers', 'footers', 'quotes'))

# Step 2: TF-IDF Vectorization
vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
X_tfidf = vectorizer.fit_transform(newsgroups.data)
print(f"Original TF-IDF shape: {X_tfidf.shape}")  # (docs x terms)

# Step 3: SVD for LSI (Latent Semantic Indexing)
k = 100  # number of latent dimensions
svd = TruncatedSVD(n_components=k)
X_lsi = svd.fit_transform(X_tfidf)
print(f"Reduced LSI shape: {X_lsi.shape}")  # (docs x k topics)

# Step 4: Show similarity between some documents
def show_similar_docs(query_idx, top_n=5):
    similarities = cosine_similarity([X_lsi[query_idx]], X_lsi)[0]
    top_indices = similarities.argsort()[::-1][1:top_n+1]
```

```
    print(f"\nQuery Document #{query_idx}:\n{newsgroups.data[query_idx][:300]}...\n")
    print("Top similar documents:")
    for i in top_indices:
        print(f"\nDoc #{i} (Similarity: {similarities[i]:.3f}):\n{newsgroups.data[i][:300]}...")

# Example: Show top 5 similar documents to doc #0
show_similar_docs(query_idx=0, top_n=5)
```