NLP project on:

LSA

(Latent Semantic Analysis)

By: Aishwarya Sahay

BTECH/10197/21

CSE – ‘B’

Introduction:

Latent Semantic Analysis (LSA) is a natural language processing (NLP) and information retrieval technique that aims to extract a big text corpus's underlying structure. LSA is a type of dimensionality reduction method that is especially useful for examining the connections between terms and documents in a text collection.

Key concepts :

Vector Space Model:

LSA operates within the Vector Space Model, where documents and terms are represented as vectors in a high-dimensional space. Each term is associated with a unique dimension, and the value of each dimension corresponds to the term's importance in a document.

Term-Document Matrix:

LSA begins by constructing a Term-Document Matrix (TDM) or its variant, the Term-Frequency Inverse Document Frequency (TF-IDF) matrix, where rows correspond to terms, columns correspond to documents, and matrix entries represent the term's frequency or importance in a document.

Singular Value Decomposition (SVD):

The core of LSA involves applying Singular Value Decomposition (SVD) to the Term-Document Matrix. SVD is a mathematical technique that decomposes a matrix into three other matrices: U, Σ, and V^T. This process helps identify latent semantic structures and reduces the dimensionality of the original matrix.

Dimensionality Reduction:

LSA retains only the top-k singular values and their corresponding vectors, effectively reducing the dimensionality of the original matrix. This process helps capture the most important patterns and relationships while discarding noise and less relevant information.

Semantic Space:

The reduced matrices (U, Σ, and V^T) create a semantic space where terms and documents are represented as vectors with fewer dimensions. Similarity measures, such as cosine similarity, can be applied in this space to measure the semantic similarity between terms and documents.

Application of LSA:

Latent Semantic Analysis (LSA) has various applications in natural language processing (NLP), information retrieval, and text analysis. Some of the key applications of LSA include:

Information Retrieval:

LSA is used to improve the accuracy of information retrieval systems by capturing the semantic relationships between terms and documents. It helps in retrieving documents that are semantically relevant to a user's query, even if the specific terms used in the query are not present in the documents.

Document Clustering:

LSA is applied in document clustering tasks to group similar documents together based on

their latent semantic content. By representing documents in a reduced-dimensional semantic space, LSA can identify clusters of documents that share common topics or themes.

Topic Modeling:

LSA is used for uncovering latent topics within a collection of documents. By analyzing the relationships between terms and documents, LSA can identify groups of words that tend to co-occur and represent them as topics. Each document is then associated with a distribution over these topics.

Text Summarization:

LSA is employed in extractive summarization, where the most important sentences or passages are selected to create a concise summary. By identifying the key concepts and

relationships between terms, LSA helps in extracting the most informative content from a document or a set of documents.

Challenges:

While Latent Semantic Analysis (LSA) is a powerful technique for uncovering latent structures in large text corpora, it is not without its challenges. Here are some of the key challenges associated with LSA:

Loss of Interpretability:

The reduced-dimensional semantic space obtained through LSA may lose some interpretability compared to the original high-dimensional space. While it captures latent patterns, the individual components of the

reduced space may not have clear semantic meanings.

Sensitivity to Term Variations:

LSA is sensitive to variations in terms, including synonyms and polysemous words. Synonyms might not be treated as equivalent, and the model may struggle to differentiate between different senses of polysemous words.

Difficulty Handling Polysemy:

Polysemy, where a single word has multiple meanings, poses a challenge for LSA. The technique may not effectively disambiguate between different senses of a word, potentially leading to a loss of precision.

Sparsity of the Term-Document Matrix:

In large datasets, the term-document matrix can be very sparse, meaning that many entries are zero. This sparsity can affect the quality of the singular value decomposition (SVD) and may require additional preprocessing or regularization techniques.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.datasets import fetch_20newsgroups
import pandas as pd

categories = ['alt.atheism', 'talk.religion.misc', 'comp.graphics', 'sci.space']
dataset = fetch_20newsgroups(subset='all', categories=categories, shuffle=True, random_state=42)

tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
tfidf_matrix = tfidf_vectorizer.fit_transform(dataset.data)


print("\nTF-IDF Matrix:")
print(pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out()))

num_topics = 3
lsa_model = TruncatedSVD(n_components=num_topics)
lsa_topic_matrix = lsa_model.fit_transform(tfidf_matrix)

terms = tfidf_vectorizer.get_feature_names_out()
for i, topic in enumerate(lsa_model.components_):
    top_terms_idx = topic.argsort()[-10:][::-1]
    top_terms = [terms[idx] for idx in top_terms_idx]
    print(f"\nTopic {i + 1}: {', '.join(top_terms)}")

document_term_matrix = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())
print("\nDocument-Term Matrix:")
print(document_term_matrix)


document_topic_matrix = pd.DataFrame(lsa_topic_matrix, columns=[f"Topic {i + 1}" for i in range(num_topics)])
print("\nDocument-Topic Matrix:")
print(document_topic_matrix)


topic_assignments = document_topic_matrix.idxmax(axis=1)
topic_assignments = topic_assignments.map(lambda x: int(x.split()[1]))

doc_topic_df = pd.DataFrame({'Document': dataset.target, 'Topic': topic_assignments})
print("\nDocument Topics:")
print(doc_topic_df)
```

```
TF-IDF Matrix:
            00  000      10  100   11   12       128   13        14        15  \
0     0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.095452  0.000000
1     0.000000  0.0  0.1848  0.0  0.0  0.0  0.145662  0.0  0.000000  0.000000
2     0.039196  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.028473
3     0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
4     0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
...        ...  ...     ...  ...  ...  ...       ...  ...       ...       ...
3382  0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
3383  0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
3384  0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
3385  0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
3386  0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000

          ...     wrote   xv      year     years  yes  york  young  zip  zoo  \
0         ...  0.000000  0.0  0.000000  0.000000  0.0   0.0    0.0  0.0  0.0
1         ...  0.086026  0.0  0.000000  0.000000  0.0   0.0    0.0  0.0  0.0
2         ...  0.000000  0.0  0.031549  0.000000  0.0   0.0    0.0  0.0  0.0
3         ...  0.000000  0.0  0.000000  0.000000  0.0   0.0    0.0  0.0  0.0
4         ...  0.000000  0.0  0.000000  0.000000  0.0   0.0    0.0  0.0  0.0
...       ...       ...  ...       ...       ...  ...   ...    ...  ...  ...
3382      ...  0.076973  0.0  0.000000  0.000000  0.0   0.0    0.0  0.0  0.0
3383      ...  0.000000  0.0  0.000000  0.189449  0.0   0.0    0.0  0.0  0.0
3384      ...  0.000000  0.0  0.190536  0.000000  0.0   0.0    0.0  0.0  0.0
3385      ...  0.000000  0.0  0.000000  0.000000  0.0   0.0    0.0  0.0  0.0
3386      ...  0.000000  0.0  0.000000  0.000000  0.0   0.0    0.0  0.0  0.0

          zoology
0             0.0
1             0.0
2             0.0
3             0.0
4             0.0
...           ...
3382          0.0
```

```
3383      0.0
3384      0.0
3385      0.0
3386      0.0

[3387 rows x 1000 columns]

Topic 1: edu, com, writes, god, space, article, subject, lines, organization, don

Topic 2: god, sandvik, jesus, kent, people, com, apple, christian, newton, sgi

Topic 3: henry, access, space, digex, pat, toronto, nasa, net, sandvik, zoo

Document-Term Matrix:
            00  000      10  100   11   12       128   13        14        15  \
0     0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.095452  0.000000
1     0.000000  0.0  0.1848  0.0  0.0  0.0  0.145662  0.0  0.000000  0.000000
2     0.039196  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.028473
3     0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
4     0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
...        ...  ...     ...  ...  ...  ...       ...  ...       ...       ...
3382  0.000000  0.0  0.0000  0.0  0.0  0.0  0.000000  0.0  0.000000  0.000000
```