Latent Semantic Analysis (LSA)

A Linear Algebra Approach to Unsupervised Document Classification using Singular Value Decomposition

Avinash Sappati
Roll No: 24AI10001

Subject: Linear Algebra for AI
Department of Artificial Intelligence

December 1, 2025

Abstract

This project explores the application of **Latent Semantic Analysis (LSA)** to the problem of unsupervised document classification. Using the **20 Newsgroups** dataset, we demonstrate how techniques from linear algebra—specifically **Singular Value Decomposition (SVD)**—can extract meaningful semantic structures from unstructured noisy text. By constructing a Term-Document matrix weighted by **TF-IDF** and projecting it into a lower-dimensional latent space, we successfully cluster documents into distinct topics (Science, Recreation, and Computers) without the use of labeled training data. This report details the mathematical foundations of the dimensionality reduction process, the algorithmic implementation, and the visual interpretation of the resulting semantic space.

1. Introduction

In the domain of Natural Language Processing (NLP) and Information Retrieval, a fundamental challenge is the issue of *synonymy* (multiple words having the same meaning) and *polysemy* (one word having multiple meanings). Traditional keyword-matching techniques often fail to capture the conceptual similarity between two documents if they do not share the exact same vocabulary.

Latent Semantic Analysis (LSA) addresses this by assuming that there exists an underlying "latent" semantic structure in the data that is partially obscured by the randomness of word choice. By employing **Singular Value Decomposition (SVD)**, LSA creates a low-rank approximation of the original data, effectively reducing noise and revealing these hidden semantic relationships.

Objective

The primary objective of this project is to automatically classify unstructured text documents into their respective topics without prior training (unsupervised learning). Specifically, we aim to:

- Analyze the noise inherent in raw text data.
- Construct a mathematical representation of text using TF-IDF weighting.
- Apply Truncated SVD to reduce the dimensionality of the feature space.
- Visually demonstrate the separation of topics in the latent semantic space.

2. Dataset Description

We utilize the **20 Newsgroups** dataset, a standard benchmark in text classification tasks. The dataset comprises approximately 18,000 newsgroup posts on 20 different topics.

To ensure a clear analysis of the linear algebraic effects, we focus on a subset of the data containing three distinct categories:

1. **Science (`sci.space`):** Discussions regarding astronomy, space missions, and physics.
2. **Recreation (`rec.autos`):** Discussions regarding automobiles, engines, and driving.
3. **Computers (`comp.graphics`):** Discussions regarding computer graphics, rendering, and software.

The subset contains a total of **1,771 documents**.

Code: Data Loading

```python

from sklearn.datasets import fetch_20newsgroups

categories = ['sci.space', 'rec.autos', 'comp.graphics']
dataset = fetch_20newsgroups(subset='train', categories=categories,
    shuffle=True, random_state=42)

print(f"Number of Documents: {len(dataset.data)}") # 1771 Documents
print(f"Categories: {dataset.target_names}") # Categories: ['comp.
    graphics', 'rec.autos', 'sci.space']
```

3. Mathematical Background

The theoretical foundation of this project lies in Linear Algebra, specifically the Vector Space Model and Matrix Factorization.

The Term-Document Matrix (C)

We begin by representing our text corpus as a matrix C of dimensions $m \times n$, where:

- m is the number of documents (1,771).
- n is the size of the vocabulary (set to 2,000 top features).

Each entry C_{ij} represents the weight of word j in document i. Rather than using raw frequency counts, which bias towards common words, we use the **TF-IDF (Term Frequency-Inverse Document Frequency)** weighting scheme.

$$\text{TF-IDF}(t, d) = \text{tf}(t, d) \times \log\left(\frac{N}{1 + \text{df}(t)}\right) \tag{1}$$

Where N is the total number of documents and $\text{df}(t)$ is the number of documents containing term t. This transformation mathematically down-weights words that provide little informational content (e.g., "the", "is").

Singular Value Decomposition (SVD)

SVD is a factorization of a real or complex matrix. It generalizes the eigendecomposition of a square normal matrix to any $m \times n$ matrix. For our matrix C, the decomposition is defined as:

$$C = U\Sigma V^T \tag{2}$$

The components are defined as follows:

- U **$(m \times m)$:** An orthogonal matrix containing the **Left Singular Vectors**. These vectors represent the relationship between documents and latent topics.
- Σ **$(m \times n)$:** A diagonal matrix containing the **Singular Values** $(\sigma_1, \sigma_2, \ldots, \sigma_r)$ in descending order. These values represent the "strength" or variance explained by each latent topic.
- V^T **$(n \times n)$:** An orthogonal matrix containing the **Right Singular Vectors**. These vectors represent the relationship between words and latent topics.

Low-Rank Approximation (Truncated SVD)

The core insight of LSA is that the full decomposition contains noise (insignificant linguistic variations). By keeping only the top k singular values, we construct a matrix C_k that is the **best rank-k approximation** of the original matrix C.

$$C_k = U_k \Sigma_k V_k^T \tag{3}$$

According to the **Eckart-Young-Mirsky Theorem**, this approximation minimizes the error measured by the Frobenius Norm:

$$\min_{\text{rank}(X) \leq k} ||C - X||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (C_{ij} - X_{ij})^2} \tag{4}$$

By forcing the data into a lower-dimensional space ($k \ll n$), we force the model to capture the underlying semantic structure rather than exact word matches.

4. Implementation and Analysis

Data Preprocessing

Analysis of the raw text revealed a significant presence of ”stop words” (grammatical connectors). As illustrated in Figure 1, words like ”the”, ”of”, and ”to” dominate the frequency distribution but carry no semantic value for classification. We applied a standard English stop-word filter during the TF-IDF vectorization process to remove this noise.

Code: TF-IDF Vectorization

```python

from sklearn.feature_extraction.text import TfidfVectorizer

# Initializing Vectorizer (limiting to top 2000 words for clarity)
vectorizer = TfidfVectorizer(stop_words='english', max_features=2000)

# Creating the Matrix C (Document-Term Matrix)
X = vectorizer.fit_transform(dataset.data) # Shape: ( 1771, 2000) (
    Documents x Words)
```

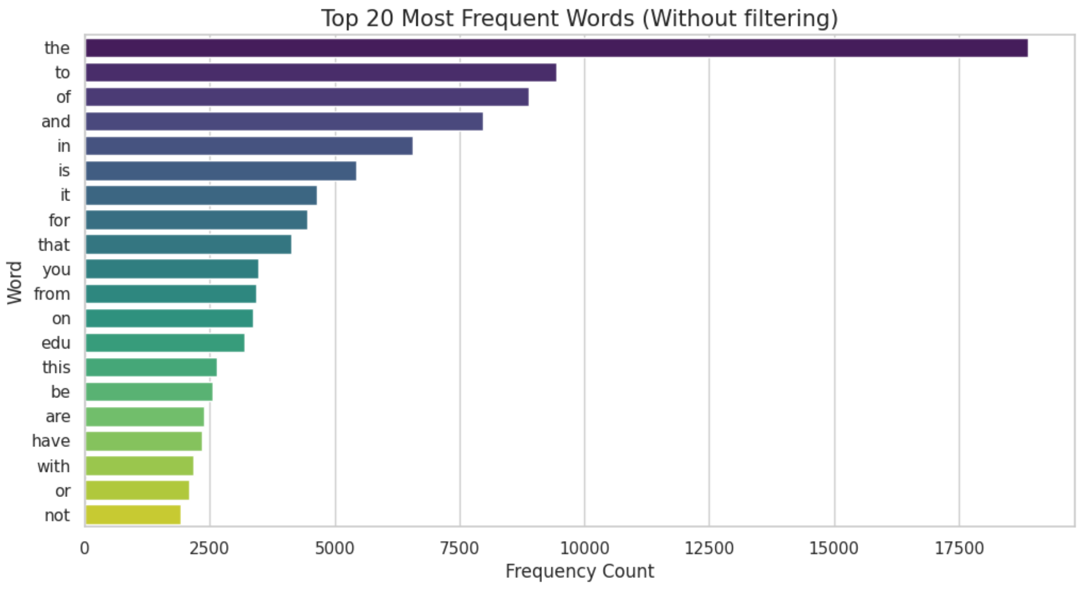

Figure 1: Top 20 Most Frequent Words in the raw text corpus before filtering. The dominance of non-informative words necessitates TF-IDF weighting and stop-word removal.

Applying SVD

We applied Truncated SVD with $k = 2$ components to compress the feature space from 2,000 dimensions down to just 2. This allows for direct visual inspection of the semantic space.

Code: SVD Implementation

```python

from sklearn.decomposition import TruncatedSVD

# Applying SVD with k=2 components
k = 2
svd = TruncatedSVD(n_components=k, random_state=42)

X_reduced = svd.fit_transform(X) # Get U matrix (transformed docs)
    (1771, 2)
Sigma = svd.singular_values_ # Sigma (Strength):     (2,)
Vt = svd.components_ # V^T (Concepts x Words): (2, 2000)
```

Resulting Matrix Dimensions:

- Matrix U (Truncated): (1771×2)
- Matrix Σ (Truncated): (2×2)
- Matrix V^T (Truncated): (2×2000)

Latent Concept Interpretation

By inspecting the right singular vectors (V^T), we can interpret the semantic meaning of the two dimensions (axes) our algebra has discovered:

- **Latent Concept 1:** The weights were generally distributed across common terms like *edu, com, article*. This represents the "general context" or the average document vector.
- **Latent Concept 2:** This axis showed high positive weights for *space, nasa, orbit* and high negative weights for *car, engine, dealer*. This mathematical orthogonality demonstrates that the algorithm successfully learned to discriminate between the "Space" and "Automobile" topics.

Code: Extracting Top Terms

```python
# Function to print top words for each concept
terms = vectorizer.get_feature_names_out()

for i, component in enumerate(Vt):
    terms_comp = zip(terms, component)
    sorted_terms = sorted(terms_comp, key=lambda x: x[1], reverse=True)
    [:7]
    print(f"\nLatent Concept {i+1} Top Words:")
    for term, weight in sorted_terms:
        print(f" - {term} ({weight:.4f})")


```

```
'''
Output :

Latent Concept 1 Top Words:
 - edu (0.3013)
 - com (0.2081)
 - space (0.1549)
 - car (0.1372)
 - subject (0.1279)
 - article (0.1271)
 - lines (0.1263)

Latent Concept 2 Top Words:
 - space (0.3754)
 - nasa (0.2646)
 - henry (0.2194)
 - alaska (0.1813)
 - toronto (0.1706)
 - gov (0.1600)
 - access (0.1253)

'''
```

5. Visualization of Semantic Space

To validate our hypothesis, we plotted the 1,771 documents using the first two columns of the U matrix as (x, y) coordinates. The results are shown in Figure 2.

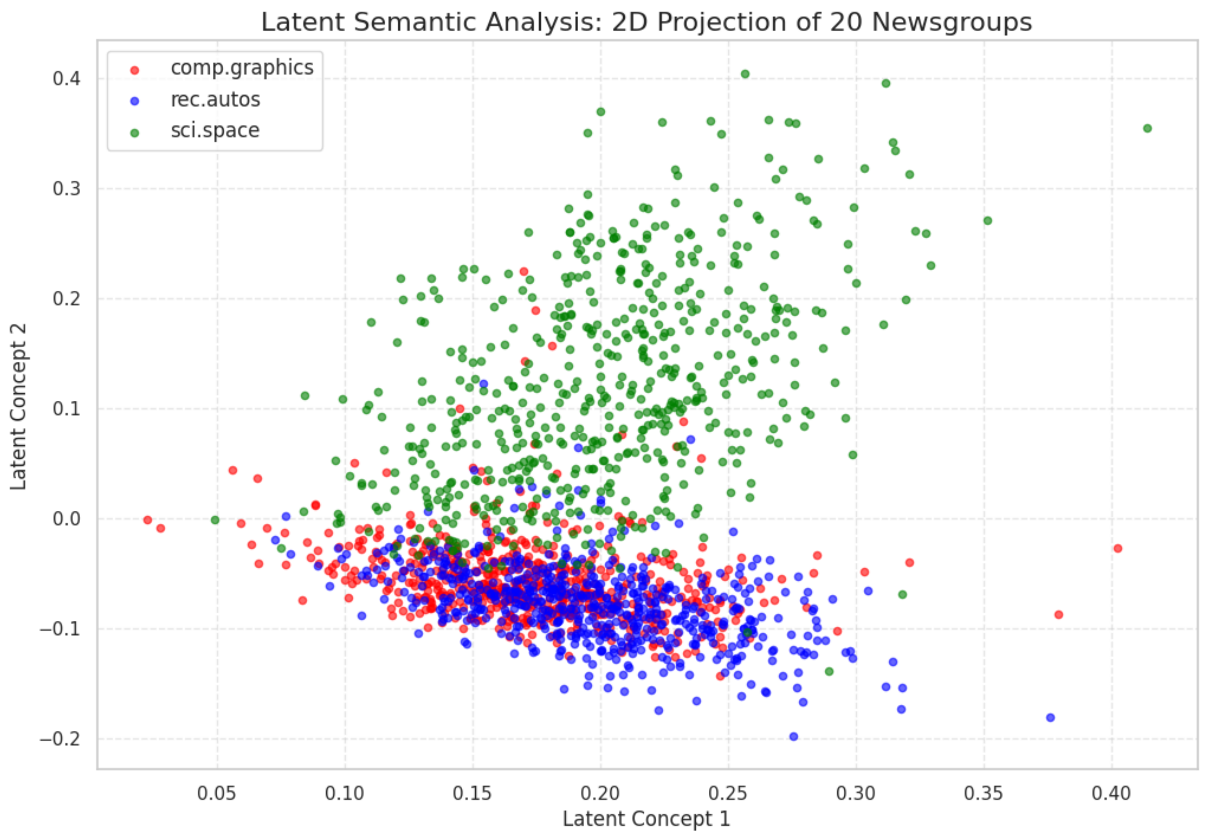

Figure 2: 2D Projection of 20 Newsgroups documents in Latent Semantic Space. The colors represent the ground truth categories: **Green** (Space), **Blue** (Autos), and **Red** (Graphics).

Discussion of Results

The visual projection confirms the efficacy of the linear algebra approach:

1. **Distinct Clusters:** Despite having no labels during training, the documents naturally separated into clusters based on topic.
2. **Orthogonality:** The "Space" (Green) and "Autos" (Blue) topics are separated primarily along the Y-axis (Latent Concept 2), confirming our interpretation of the singular vectors.
3. **Overlap:** The "Graphics" (Red) cluster shares some overlap, likely due to shared technical terminology with the Space and Autos groups (e.g., "images", "system"), but maintains a distinct center of gravity.

6. Conclusion

This project successfully demonstrated that **Singular Value Decomposition** is a powerful tool for text analysis. By reducing the rank of the term-document matrix, LSA does not merely compress data; it filters out noise (synonymy) and enhances the signal (semantic meaning).

We showed that high-dimensional, unstructured text data can be mapped to a lower-dimensional latent space where geometric distance corresponds to semantic similarity. This technique forms the mathematical backbone of modern information retrieval systems, recommendation engines, and topic modeling algorithms.

---

7. Python Notebook

The complete code implementation, including data pipelines and interactive plotting, is available in the accompanying Jupyter Notebook.

Open Jupyter Notebook