Using SVD-Based Document Representation for Text Classification Using the Reuters Data

Gayathri Pittu
Department of Computer Science
Bowling Green State University
Bowling Green, Ohio
gpittu@bgsu.edu
Advisor: Prof. Dasigi

***Abstract*—Latent Semantic Indexing (LSI), leveraging Singular Value Decomposition (SVD), is explored for its effectiveness in text classification using the Reuters dataset.The study analyzes how dimensionality reduction through LSI affects classification accuracy and processing efficiency. To achieve optimal performance, LSI is integrated with Multi-Layer Perceptrons (MLPs). Experiments are conducted to understand how varying latent dimensions influence classification performance. This analysis aims to identify the ideal balance between dimensionality reduction and preserving semantic relationships within the text data. The results demonstrate that using 100 to 500 latent dimensions significantly improves text classification accuracy, making this approach particularly suitable for large datasets while maintaining computational efficiency.**

***Index Terms*—Latent Semantic Indexing (LSI), Singular Value Decomposition (SVD), Text Classification, Reuters Dataset, Multi-Layer Perceptrons (MLPs), Dimensionality Reduction**

I. Introduction

Automatic text classification is useful for retrieving information and organizing tasks, and it plays an important role in many real-world applications such as sentiment analysis, document categorization, spam detection etc. It enables efficient classification of large document collections, facilitating information access and analysis. The effectiveness of text classification algorithms is highly dependent on the quality of the feature representations used to capture the semantic content of documents. However, dealing with high-dimensional text data leads to significant computational challenges and may affecting classification performance. Hence, there arises a pressing need for effective dimensionality reduction techniques tailored to textual data.

Latent Semantic Indexing (LSI), which is based on Singular Value Decomposition (SVD), is a dimensionality reduction technique widely used in text processing. LSI aims to capture the underlying latent semantic structure of a document collection by identifying a lower-dimensional subspace that preserves the most relevant information [1]. By applying LSI, we can expect a reduction in feature space complexity while maintaining the essential semantic relationships between documents.

This study investigates the effectiveness of LSI for text classification using the Reuters dataset [6], which is known for its varied collection of news articles across many categories , a benchmark collection for text categorization tasks. We explore the impact of varying the number of latent dimensions (K) on the classification performance. This work aims to determine the optimal K value that balances dimensionality reduction and the preservation of semantic information for improved text classification accuracy.

In this work, we leverage Multi-Layer Perceptron's (MLPs) as our text classification model. MLPs are a powerful class of artificial neural networks known for their ability to learn complex non-linear relationships in data. By combining LSI for dimensionality reduction with an MLP classifier, we aim to achieve accurate classification while maintaining efficiency.

The remainder of this paper is structured as follows: Section II presents a review of related work in text classification. Section III details our

Identify applicable funding agency here. If none, delete this.

methodology, including the steps for data collection, preprocessing, and TDM calculation. Section IV describes the dimensionality reduction techniques, including SVD Decomposition and optimal K values. Section V outlines the MLP model architecture, the experiments conducted, and the evaluation metrics used. Section VI presents the results, discusses limitations, and outlines potential future directions.

II. Related study

Textual data often suffers from high dimensionality and sparsity, posing challenges for traditional machine learning algorithms. LSI, a powerful dimensionality reduction technique, addresses this by utilizing Singular Value Decomposition (SVD) to transform text data into a lower-dimensional semantic space (Deerwester et al., 1990) [1]. The process reveals hidden patterns within the data and relationships between words, which were concealed in the original high-dimensional model (Berry et al., 2003) [3]. By capturing semantic meaning, LSI allows for a more detailed understanding of textual content.

The effectiveness of LSI in text classification tasks is well-supported by research. Harrag et al. (2010) compared various dimensionality reduction techniques for Arabic text classification and found LSI, along with Document Frequency (DF) and TFIDF, to be particularly efficient and effective [2]. This highlights LSI's suitability for large-scale text classification problems.

While LSI offers significant benefits, applying it to very large datasets can be computationally expensive. Dasigi et al. (2011) introduced the concept of reference libraries as a solution [4]. These libraries act as condensed representations of the entire dataset, capturing its core themes. By performing LSI on this smaller, pre-defined space, researchers can significantly reduce computational costs while preserving informative content crucial for classification. This approach has been shown to maintain informative content crucial for classification and potentially even improve classification accuracy by focusing the analysis on the most relevant semantic relationships within the document collection.

MLPs are a type of artificial neural network architecture that excels at analyzing and categorizing complex data, including text. Quispe et al. (2017) introduced a novel text classification method utilizing MLPs, demonstrating their effectiveness in handling large and intricate documents [5]. MLPs can capture nuanced relationships and complex text features, leading to enhanced accuracy in identifying relevant categories for each document. This makes them particularly well-suited for your project scope if you are dealing with complex textual content.

The studies reviewed in this section highlight the potential of Latent Semantic Indexing (LSI) and Multi-Layer Perceptrons (MLPs) for improved text classification, particularly when dealing with large and complex datasets. The combination of LSI's capability of extracting semantic meaning and reducing dimensionality, together with MLP's capacity for pattern recognition, allows us to develop a robust and efficient text classification system

III. Methodology

A. Dataset

The data originates from the publicly available Reuters Corpus Volume 1 (RCV1) [6]. This study utilizes a subset of RCV1 encompassing news articles from August 20, 1996, to August 19, 1997. This subset comprises 806,791 XML files, each representing a single news story. These XML files contain various metadata fields, such as region codes to identify the geographical location of the news event, industry codes relevant to the news content, and most importantly for this study, topic codes for subject categorization. The dataset contains a total of 130 unique topic categories used to classify the news articles. These classifications can be:

- Single-label: A document is assigned to a single topic class (e.g., C42).
- Multi-label: A document is assigned to two or more topic classes simultaneously (e.g., CCAT, E41, ECAT, GCAT, GJOB).

B. Classification – Methodologies

1) Single-label

The dataset contains 70 unique topic categories under the single-label with a total of 33,226 documents. As shown in table I, the dataset is highly imbalanced, with the first two categories (GCAT and M11) containing a significantly higher number of documents compared to the remaining categories. To address the data imbalance and improve model performance, we focus on a subset of these topics,

Topic Code	Original Count	Subset_1 Count	Subset_2 Count
GCAT	23935	200	500
M11	2674	200	500
C181	676	200	676
C151	647	200	647
CCAT	442	200	442
C152	338	200	338
E512	305	200	305
C11	232	200	232
C31	229	200	229
GD1P	203	200	203

TABLE I: Distribution of Document Counts Across Subsets by Topic Code

specifically the top 10 most populous categories. This selection aims to ensure a manageable number of classes while maintaining a representative sample of the data.

a) Subset 1

This subset aims for a balanced class distribution by selecting 200 documents from each of the top 10 categories. To maintain sufficient data, we select 200 documents from each category because the 10th category contains exactly 200 documents.

b) Subset 2

This subset acknowledges the inherent class imbalance by selecting documents proportionally to their original class distribution. We chose 500 documents each from the top two most populated categories (GCAT and M11) due to their significantly higher document counts (23,935 and 2,674 documents, respectively). The remaining 8 categories are included with their original document counts (ranging from 203 to 676 documents).

2) Multi-label

For multi-label classification, our focus was on three topic labels within the data. These three topic labels are categorized across 102 topics, with a total of over 360,000 documents. To make the data more manageable and focus on multi-label aspects, we implemented a cleaning process.

Initially, the dataset consisted of sequences of labels representing various topics associated with each document. For instance, a sequence like ['C15', 'C151', 'CCAT'] signifies that a document is associated with three distinct topics. However, our initial approach did not consider these sequences; instead, we aimed to extract individual topic counts within sequences and subsequently removed topics

Number of labels	Number of documents
3	357,359
2	4,148
1	538
0	66

TABLE II: Distribution of Documents by Number of Labels in Multi-label Classification

associated with fewer than 500 documents. This resulted in documents having various combinations of remaining labels, ranging from all three labels together to just a single label.

For example, suppose there are 400 documents associated with the label 'CCAT', 200 documents with 'C22', and 600 or more documents with labels 'C15', 'C151', 'M14', 'M143', and 'MCAT'. After filtering out labels with less than 500 documents, we observed instances such as:

- Documents with labels 'C15', 'C151' (2 labels).
- Documents with label 'C11' (1 label).
- Documents with labels 'M14', 'M143', 'MCAT' (3 labels).

As a result of this process, the distribution shown in table II was obtained.

Since we shifted focus to multi-label classification, documents with only one label or no labels were then excluded. Finally, computational resources limited our ability to analyze the entire dataset. To address this, we balanced the data by selecting a sample of 500 documents from each remaining sequence containing more than 500 documents. We chose 52 out of the initial 58 sequences because only 52 sequences had more than 500 documents after filtering. This process resulted in a final dataset of 26,000 documents spread across 52 unique sequences, each representing a combination of our three chosen labels. This selection process ensures a balanced representation of the various multi-label combinations while remaining manageable for analysis with limited computational resources.

C. Data Preprocessing

Data preparation is an important preliminary step in text classification, laying the foundation for more robust and accurate modeling. Preprocessing improves the quality of the input data by applying a methodical cleaning, standardizing, and structuring process. A series of text preprocessing steps was implemented to clean, normalize, and structure the raw text data, transforming it into a format that is more suitable for further exploration.

- Textual contractions like ”won’t” are expanded to their full forms (”will not”) to ensure consistency and readability. Titles like ”Mr.”, ”Dr.”, ”Prof.” etc. are removed as they often don’t contribute significant meaning to the core content of the text.
- For consistency, all text is converted to lowercase. NLP models are often biased towards case-sensitive words, so lowercasing ensures all words are treated equally.
- The text is split into individual words or meaningful units (tokens) using regular expressions. Tokenized text allows us to understand word usage patterns and build features like term frequency for analysis.
- Common and uninformative words, known as stop words, such as ’a’, ’the’. and ’is’ are removed using a pre-defined stop word list. These words carry little meaning in the context of our analysis, and eliminating them allows us to focus on content-rich words.
- Punctuation marks are removed using regular expressions. Numbers are also removed as they might not be relevant for the analysis.
- Words are reduced to their base form (lemma) using the WordNetLemmatizer. This process helps capture the meaning of words regardless of their grammatical variations and improves model performance by ensuring that words with the same meaning are treated similarly. This can be particularly beneficial for tasks like word counting or building document similarity measures. Finally, any remaining empty strings and underscores are removed to clean up the text further.

D. Document Representation and Normalization

Document representation involves converting text data into a numerical format suitable for machine learning algorithms. This numerical representation allows us to analyze documents based on their word usage patterns and relationships, relying on the concept of vectors. One of the core techniques for text vectorization is the Term Document Matrix (TDM), which is derived from the most common technique: Count Vectorization. Count vectorization is a fundamental technique in NLP (Natural Language Processing) for converting textual data into numerical vectors.

1) Vector Representation (Term Document Matrix - TDM)

Count vectorization involves counting occurrences of terms within each document, producing a raw count vector representing the frequency of each term across each document. These raw counts are then organized into the TDM, where each entry signifies the count of a specific term in a particular document.

The term-document matrix X is constructed such that each entry Xij represents the count of word i in document j. In essence, this TDM transforms each document into a vector in a high-dimensional space, where each dimension corresponds to a term in the vocabulary. The weight (Xij) in the matrix indicates the importance of that term (word I) to that specific document (j).

The size of the TDM is denoted as t x d, where:

- ‘t’ represents the total number of distinct words (terms) across the corpus (collection of documents).
- ‘d’ represents the number of documents in the corpus.

2) Limitations of Raw Term Counts

While the TDM provides a basic representation of documents, the raw counts may not accurately reflect the importance of terms. Frequent but generic words like ”the” or ”a” might have high counts across all documents, skewing the weight distribution. These common words contribute less to understanding the unique content of each document. To address this limitation, normalization techniques are employed to adjust the weights in the TDM.

3) Normalization for Improved Weighting

Normalization aims to place more emphasis on terms that are distinctive to a specific document or a smaller group of documents. This process helps identify terms that are more relevant for understanding the content and differentiating documents from each other.

The Term Document Matrix (TDM) captures basic document information, but it may overemphasize terms that occur frequently. Normalization techniques like ”maximum term frequency normalization" address this. This method identifies the most frequent term within each document and divides all term counts by that value. This reduces the importance of terms that appear often within a single document, even if they are not commonly used across the corpus. As a result, less frequent terms, potentially more informative for understanding the document's unique content, gain relatively more weight. A major benefit of this approach is that it refines the TDM by emphasizing terms that differentiate documents and perhaps reveal hidden connections between them.

Count vectorization has limitations. It doesn't capture word order or semantic relationships between words. Other text representation techniques, such as word embedding models, address these limitations.

Another approach, which this work focuses on, is Latent Semantic Indexing (LSI). LSI utilizes Singular Value Decomposition (SVD) to analyze the TDM and identify underlying latent semantic structures within the data. By reducing the dimensionality of the TDM while preserving the most important information, LSI can improve the effectiveness of tasks like information retrieval and document clustering.

IV. LSI-BASED DIMENSIONALITY REDUCTION

Latent Semantic Indexing (LSI) offers a powerful approach to information retrieval by addressing the limitations of techniques like count vectorization. While count vectorization provides a foundation, it struggles to capture semantic relationships between words. This can lead to difficulties with synonyms and polysemy. LSI tackles these challenges by uncovering the underlying thematic structure within text data.

This ability to capture semantic relationships builds upon the groundbreaking work of Deerwester et al. (1990) [1]. Their research introduced LSI, highlighting its potential to overcome limitations inherent in keyword-based retrieval systems. Instead of relying solely on keyword matching, LSI delves deeper, identifying hidden connections between terms.

LSI leverages a mathematical technique called Singular Value Decomposition (SVD) to analyze the term-document matrix (TDM). SVD decomposes the TDM into a lower-dimensional space, prioritizing the most important semantic information. This not only enhances retrieval accuracy but also facilitates more efficient processing.

A. Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) offers a powerful matrix factorization technique for analyzing high-dimensional data. It decomposes a matrix into key components, revealing underlying structure and facilitating dimensionality reduction tasks.

SVD is applied to the term-document matrix (X), decomposing it into three key components:

- U (t x k): Left singular vectors, representing relationships between terms.
- $\Sigma(k \times k)$: Diagonal matrix containing singular values, reflecting the significance of each dimension (concept) identified by SVD.
- $V^T(d \times k)$: Transposed right singular vectors, representing relationships between documents based on the identified concepts.

Here's a simplified representation of SVD applied to the normalized TDM (D):

$$D(t \times d) = U(t \times k)\Sigma(k \times k)V^{\intercal}(d \times k) \qquad (1)$$

However, for massive datasets like the Reuters Corpus Volume 1 (RCV1), the computational cost of SVD can become significant. As noted by Papadimitriou et al. [3], the complexity of SVD scales proportionally to the product of the number of documents, number of terms, and the number of dimensions retained. For RCV1, performing SVD on the entire document-term matrix can be computationally expensive and slow down processing

B. Considering Reference Libraries for Efficient SVD Computation

A promising strategy to address the computational cost of SVD for large document collections involves the use of reference libraries. Introduced by Dasigi [4], a reference library is a carefully chosen subset of the entire document collection that effectively captures the essential concepts present within the larger dataset. The selection process for the reference library is crucial, as it should maintain a balance between size and representativeness.

By performing SVD on the smaller reference library matrix instead of the full document-term matrix, significant computational efficiency gains can be achieved. This translates to faster processing times, making SVD a more viable option for analyzing large-scale document collections like RCV1.

As described in dataset section, this work utilizes subsets of RCV1 documents for LSI applications. These chosen subsets serve as your reference libraries.

- Subset 1 (Balanced Single-Label, 2,000 Documents, 20,782 terms)
- Subset 2 (Imbalanced Single-Label, 4072 Documents, 31073 terms)
- Multi-Label Subset (26,000 Documents, 71994 terms)

C. *Selecting the Optimal Dimensionality (k) for Effective LSI*

The next crucial step after SVD is to determine how many dimensions (k) should be retained from the chosen reference library. This choice significantly impacts the effectiveness of LSI:

Selecting the appropriate k value strikes a critical balance between dimensionality reduction and information preservation.

While a lower k offers significant dimensionality reduction, it might discard important semantic information. Conversely, a very high k might not achieve sufficient reduction and could lead to overfitting; higher k values can be computationally expensive to compute due to the nature of SVD.

To find the optimal k for our specific task and chosen reference library, we will experiment with a range of values, such as 50 to 500 in this work. Through assessing the performance of different k values using appropriate metrics (such as the F1 score, particularly useful for tasks like document retrieval), we can find the ideal balance between reducing dimensionality and retaining essential information in our LSI analysis.

Furthermore, previous research by Deerwester et al. [1] provides valuable insights. They suggest a k value around 100 might be suitable as a starting point for exploration. By combining these strategies, we aim to achieve an optimal k that maximizes the effectiveness of our LSI analysis within the context of the chosen reference library

V. Model Overview and Architecture

A. *Input Data Preparation*

LSI model leverages reference libraries to efficiently analyze the vast document collection. The first step involves transforming each document within the chosen reference library into a pseudo-document vector. This allows documents to be represented in a lower-dimensional space while preserving their key semantic relationships.

Following the approach outlined by Deerwester et al. [1], we transform each document within the chosen reference library into a pseudo-document vector.

This enables us to utilize the same comparison formulas employed for document retrieval tasks within the LSI framework.

The transformation for each document (i) in the reference library follows the equation (2):

$$D_i = D' \cdot U \cdot \Sigma^{-1} \tag{2}$$

- D_i $(d \times k)$: Transformed document matrix in the reduced k-dimensional space.
- D' $(d \times t)$: Transposed original term-document matrix (d documents, t terms).
- U $(t \times k)$: Term relationship matrix obtained from SVD on the reference library.
- Σ $(k \times k)$: Diagonal matrix containing singular values from SVD.
- Σ^{-1} $(k \times k)$: Diagonal matrix containing the inverse of singular values.

B. *Model Architecture*

The MLP architecture consists of two sequential hidden layers. The first hidden layer has 512 neurons, followed by a dropout layer (50% drop rate) to prevent overfitting. The second hidden layer has 256 neurons, followed by another dropout layer. This setup effectively captures the relevant relationships within the data.

The MLP model was experimented with the addition of more hidden layers, but these additions did not significantly improve results. This suggests the base model with two hidden layers effectively learns from the reduced-dimensional data.

C. *Model Training*

The MLP model was trained using early stopping to prevent overfitting. Early stopping halts training when the performance on a validation dataset stops improving, thus preventing the model from learning noise in the training data. Additionally, 10-fold cross-validation was employed to assess the model's generalization performance across different subsets of the data.

D. Evaluation Metrics

The selection of appropriate performance metrics ensures a thorough evaluation of the model's performance, considering factors such as class distribution and the specific objectives of the classification task. This is particularly important when analyzing datasets with imbalanced class distributions.

Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is a measure of the accuracy of the positive predictions made by the model.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)}+\text{False Positives (FP)}}$$

Recall

Recall is the ratio of correctly predicted positive observations to all observations in the actual class. It indicates how well the model can find all the relevant cases within a dataset.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)}+\text{False Negatives (FN)}}$$

F1-Score

The F1-score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. It is particularly useful when the classes are imbalanced.

$$\text{F1-Score} = 2 \times \left(\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}\right)$$

Accuracy

The ratio of correctly predicted observations to the total number of observations provides an overall view of a model's performance.

$$\text{Accuracy} = \frac{\text{True Positives (TP) + True Negatives (TN)}}{\text{Total Observations}}$$

For balanced datasets, accuracy serves as a straightforward and effective metric, reflecting the overall model performance by calculating the proportion of correctly classified instances relative to the total number of instances. However, in imbalanced datasets, where certain classes significantly outnumber others, relying solely on accuracy can be misleading, as it may overestimate the model's effectiveness by focusing only on the popular categories.

To mitigate this issue, the adoption of Macro and Micro metrics becomes essential. Macro metrics evaluate the performance across all classes independently, providing insight into the model's ability to perform well across diverse categories. Micro metrics aggregate the performance across all classes, giving more weight to larger classes and offering a more balanced evaluation of the model's overall performance [7]. This ensures that the model's capacity to accurately predict less frequent classes is also considered during performance assessment.

VI. Experiments and Results

The study focuses on finding the best balance between dimensionality reduction and classification accuracy through Latent Semantic Indexing (LSI). A series of experiments were conducted to investigate the impact of varying the number of dimensions (k) retained after applying LSI on the model's performance. This study covers a range of singular values (k) values, specifically from 50 to 500 for each classification approach:

- Single-Label Classification: Analyzed on both balanced (200 documents/category) and representative (original distribution) subsets.
- Multi-Label Classification: Examined on documents with overlapping topic labels.

The effectiveness of LSI was evaluated using a macro average of precision, recall, and F1-score for single-label tasks and macro and micro F1-score for multi-label tasks.

A. Results

LSI demonstrated promising results in single-label and multi-label classifications.

Single-Label

For the balanced single-label classification task, the model's performance improves as the number of singular values (K) increases from 50 to 150. There's a drop in performance after 150 singular values. At K=150, the model achieves a high level of performance with precision, recall, and F1-score all reaching 0.83. This indicates the model is both accurate in its predictions (high precision) and successful at capturing the relevant instances (high recall).

Singular values (K)	Precision	Recall	F1-Score	Accuracy
50	0.74	0.74	0.74	0.74
100	0.79	0.78	0.78	0.78
125	0.80	0.80	0.79	0.79
150	0.83	0.82	0.81	0.81
200	0.82	0.82	0.81	0.81
300	0.80	0.80	0.79	0.80
400	0.81	0.80	0.80	0.80
500	0.80	0.78	0.78	0.79

TABLE III: Single-label Classification: Balanced set results

The model performs consistently across all K values in the representative set, with precision ranging from 0.78 to 0.81, recall ranging from 0.75 to 0.79, F1-score ranging from 0.77 to 0.80, and accuracy ranging from 0.81 to 0.83. However, the model seems to perform best when K is set to 100 or 300, with an F1-score of 0.83 and accuracy of 0.83.

Singular values (K)	Precision	Recall	F1-Score	Accuracy
50	0.80	0.75	0.77	0.81
100	0.81	0.79	0.80	0.83
125	0.79	0.76	0.77	0.81
150	0.81	0.78	0.79	0.83
200	0.78	0.77	0.77	0.81
300	0.81	0.79	0.79	0.83
400	0.80	0.76	0.77	0.82
500	0.79	0.77	0.78	0.83

TABLE IV: Single-label Classification: Representative set results

From tableIII and tableIV, it can be observed that imbalanced dataset with a larger number of documents (over 4,000) generally performed better than the balanced dataset (around 2,000 documents), showcasing the advantage of leveraging larger datasets for training. This improvement can be attributed to the model's ability to learn better from a greater volume of data, as evidenced by the enhanced performance in certain classes with the imbalanced dataset.

Multi-Label

The multi-label classification results show a positive trend in performance as the number of singular values increases. Starting at K=50, the F1 Macro and Micro averages improve steadily, reaching their highest levels at K=500. At this point, the F1 scores peak at 0.86 for macro-average and 0.88 for micro-average, indicating robust and balanced classification performance across Multiple labels.

Singular values (K)	f1_Macro avg	f1_Micro avg
50	0.81	0.84
100	0.83	0.87
125	0.83	0.86
150	0.84	0.86
200	0.85	0.87
300	0.85	0.87
400	0.85	0.88
500	0.86	0.88

TABLE V: Multi-label Classification results (with 0.5 threshold)

Across both single-label and multi-label tasks, the k value range of 100 to 500 yielded similar results. This indicates that within this range, model performance is relatively stable, offering flexibility in choosing k based on factors like computational efficiency (lower k values require less processing power). It's important to note that this optimal k range might be specific to these datasets and could differ for other tasks.

The study examined how the model reacts to different thresholds, but it had to limit the analysis to just 100 singular values due to the large number of experiments needed for each possible value of K. Lower thresholds result in higher F1 scores, indicating a more balanced detection of relevant instances. However, increasing the threshold leads to a significant performance drop, especially after 0.5, with a notable reduction at a threshold of 0.9.

Threshold	F1_Macro avg	F1_Micro avg
0.3	0.83	0.86
0.4	0.83	0.86
0.5	0.83	0.87
0.6	0.81	0.86
0.7	0.79	0.85
0.8	0.74	0.82
0.9	0.68	0.78

TABLE VI: F1 scores at different thresholds for 100 singular values

B. Word Embeddings

Our study focuses on Singular Value Decomposition (SVD) as well as word embeddings to gain a better understanding of text classification techniques. This section presents the findings on word embeddings, explored by Isaac [8], in conjunction with the previously discussed SVD results. We used the same datasets(balanced and imbalanced single-label, and multi-label) and performance metrics for both SVD and word embeddings.

Word embeddings represent words as numerical vectors, capturing their semantic relationships within a high-dimensional space. This allows the model to learn the meaning of words based on their context.

The study investigated the influence of embedding size on a Multilayer Perceptron (MLP) model. Experiments revealed a trend where larger embedding sizes (256 dimensions) led to slight improvements in metrics like accuracy and F1-score. This suggests that capturing richer semantic information through larger embeddings can benefit the model. However, it's crucial to consider the increased computational resources needed for training with larger models.

Model	Accuracy	Precision	Recall	F1 Score
64 embed dim	0.79	0.80	0.79	0.79
128 embed dim	0.79	0.82	0.79	0.80
256 embed dim	0.79	0.82	0.79	0.79

TABLE VII: Word Embedding Single-label Results

Threshold	F1_macro	F1_Micro
0.3	0.76	0.80
0.4	0.76	0.81
0.5	0.75	0.81
0.6	0.73	0.80
0.7	0.71	0.78
0.8	0.67	0.76
0.9	0.59	0.72

TABLE VIII: Performance Evaluation at Different Thresholds: 26k Multi-Label Subset

Threshold	F1_macro	F1_Micro
0.3	0.76	0.93
0.4	0.76	0.94
0.5	0.76	0.94
0.6	0.74	0.94
0.7	0.72	0.94
0.8	0.69	0.93
0.9	0.64	0.92

TABLE IX: Performance Evaluation at Different Thresholds: 322k Multi-Label Subset

Multi-label classification performance was investigated using datasets of different sizes, revealing distinct patterns depending on threshold settings. For the smaller dataset 26k, both macro and micro F1-scores steadily decreased with increasing thresholds. This indicates a classic trade-off between precision (fewer false positives) and recall (fewer missed true positives). In contrast, the larger dataset 322k exhibited remarkable stability in micro F1-scores and a minor decline in macro F1-scores at higher thresholds.This suggests that the system worked well, even with more data.

Both Singular Value Decomposition (SVD) and word embeddings did consistently well in single-label classification. SVD, in particular, performed well on this dataset. However, in multi-label classification, SVD performed better than word embeddings, and the approaches showed reasonable results when using a threshold of 0.5.

Overall, SVD showed stronger performance on this dataset, especially in handling multiple labels, and stayed consistent across different settings for dimensions. However, it's essential to note that these results are only relevant to this dataset and task.

VII. Limitations of the Study

Data imbalance and computational limitations were the primary constraints of this study. To address the class imbalance, single-label classification was performed on smaller subsets (2,000 and 4,000 documents) ensuring a manageable and representative data scope. This approach, while practical, restricts the study's insights to smaller samples and may not fully represent the performance across the entire dataset.

Furthermore, the computational complexity associated with multi-label classification presented significant challenges. While the SVD approach I utilized was constrained to a 26,000-document subset due to these limitations, the word embeddings method was able to operate on the entire 322,000-document corpus. This difference highlights a significant limitation of the SVD method: its scalability and resource demands when processing larger datasets. Consequently, this restricts the ability to directly assess the SVD approach's effectiveness on a scale comparable to that of word embeddings, which demonstrated a superior capacity for handling extensive data volumes.

VIII. Conclusion and Future work

This study explored the effectiveness of Latent Semantic Indexing (LSI) for text classification using Multi-Layer Perceptrons (MLPs). The experiments demonstrated that LSI improves performance in both single-label and multi-label classification tasks on the Reuters Corpus by capturing underlying semantic relationships within the text data. We identified a range of k values between 100 and 500 as a practical choice that balances computational efficiency with information preservation in the reduced dimensionality space.

While this study focused on MLPs, future work should explore how different classification algorithms can benefit from the reduced dimensionality features provided by LSI to enhance performance. It would also be helpful to investigate LSI's performance for real-world text classification tasks across a variety of datasets and domains as well as compare alternative dimensionality reduction techniques.

References

[1] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inform. Sci. 41 (6) (1990) 391407.

[2] Harrag, Fouzi et al. "Comparing Dimension Reduction Techniques for Arabic Text Classification Using BPNN Algorithm." 2010 First International Conference on Integrated Intelligent Computing (2010): 6-11.

[3] Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. SIAM review, 37(4), 573-595.

[4] Dasigi, V., Mann, R. C., & Protopopescu, V. A. (2002). Information fusion for text classification: an experimental comparison. Information Fusion, 3(3), 163-175.

[5] Quispe, Oscar et al. "Latent semantic indexing and convolutional neural network for multi-label and multi-class text classification." 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI) (2017): 1-6.

[6] Reuters. (1997). Reuters Corpus, Volume 1. Retrieved from https://trec.nist.gov/data/reuters/reuters.html

[7] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[8] I. K. Anni and V. Dasigi, "Using WordEmbedding-Based Document Representation for Text Classification using the Reuters Data," bgsu-spring-project, 2024.