TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

REPORT

ON

TOPIC MODELING USING LATENT DIRICHLET ALLOCATION (LDA) , LATENT SEMANTIC ANALYSIS (LSA) AND BERT TRANSFORMER

SUBMITTED BY:

AGRIM PANERU (PUL77BCT008)

BINIT KC (PUL77BCT017)

BISHAL ARYAL (PUL77BCT021)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

NOVEMBER, 2024

1. Acknowledgments

We extend our heartfelt gratitude to the **Department of Electronics and Computer Engineering** at **Institute of Engineering, Pulchowk Campus** for fostering a conducive learning environment and for providing the necessary facilities that have enabled the conceptualization and preparation of this project.

We extend our deep appreciation to **Samsung Innovation Campus** for providing an enriching academic environment. The programs offered, along with the effective classes, comprehensive study materials, and hands-on lab sessions, have greatly contributed to our understanding of the subject matter and the development of this project.

We are deeply thankful to our instructor, Sir **Dr.Suresh Pokhrel**, whose expertise and insightful feedback have been invaluable in shaping the direction of this project.

We also wish to thank our peers for their constructive feedback and encouragement throughout this process. Their collaboration and input have been instrumental in shaping the final outcome of this project.

This project is the result of a collaborative effort, and we are thankful to each individual and entity that has contributed to its formulation. Your support and involvement have been invaluable.

Contents

List of Figures

2. List of Abbreviations

LDA	Latent Dirichlet Allocation
VOC	Voice of Customer
BLSTM	Bi-directional Long Short-Term Memory
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
BERT	Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing
UGC	User Generated Content
ROC	Receiver Operating Characteristic

3. Introduction

In this project, we aim to explore topic modeling techniques, specifically Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA), and BERT-based topic transformer models, to extract and analyze topics from textual data within two distinct domains: smartphone reviews and news articles. By applying these methods, we intend to uncover underlying themes and patterns in customer feedback and media content, enabling a structured understanding of prevalent topics in both datasets. The BERT-based topic transformer model, in particular, leverages pre-trained transformer architectures to capture contextual meanings and semantic relationships, providing more coherent and accurate topic extraction. This advanced approach is especially beneficial for analyzing diverse and nuanced datasets like news articles. Through this work, we aim to gain insights into consumer sentiment and news trends, showcasing the effectiveness of these techniques in processing and interpreting large-scale unstructured text data.

3.1 Background

Topic modeling is an essential technique in natural language processing (NLP) that helps uncover hidden patterns in large collections of unstructured text data. With the exponential growth of user-generated content, such as online reviews and news articles, identifying prevalent themes has become valuable for businesses and researchers alike. Smartphones, being one of the most frequently reviewed products, offer a wealth of insights into consumer preferences and concerns, while news articles reflect real-time public interest and global trends.

Traditional topic modeling methods, such as Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA), have been widely used for this purpose. However, the emergence of transformer-based models, such as BERT, has revolutionized the field by enabling deeper contextual understanding of text. BERT-based topic models excel at capturing nuanced relationships and semantic coherence, making them especially suitable for analyzing diverse datasets like smartphone reviews and news articles. These models enhance the ability to identify and interpret complex patterns, providing richer and more accurate insights into consumer behavior and societal dynamics..
Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) are widely used topic modeling methods. LDA is a probabilistic approach that assumes each document contains a mix of topics, while each topic comprises a distribution of words, enabling a highly inter-

pretable breakdown of topics. In contrast, LSA relies on matrix decomposition to reduce text data dimensions, capturing latent semantic structures. Through these methodologies, we can distill complex content in smartphone reviews and news into organized topics, providing a framework to enhance user understanding and decision-making based on text analysis.

3.2 Problem statement

The increasing volume of unstructured textual data in the form of online reviews and news articles presents challenges in efficiently extracting meaningful insights. In particular, smartphone reviews reflect diverse user experiences, while news articles cover a wide range of current events and topics. Manually analyzing such large datasets is impractical and time-consuming, making it difficult to uncover underlying themes and patterns. This project addresses the need for automated, interpretable topic modeling approaches to categorize and summarize the information contained within these datasets.

By applying Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA), and BERT-based topic transformer models, we aim to develop a structured understanding of the primary topics discussed in smartphone reviews and news content. The BERT-based approach offers a significant advantage by capturing deeper contextual and semantic relationships in the text, enhancing the coherence and accuracy of identified topics. These methodologies collectively facilitate easier interpretation and utilization of the data for decision-making and trend analysis, showcasing the potential of advanced NLP techniques in processing large-scale unstructured text.

3.3 Objectives

1. To evaluate and compare the effectiveness of LDA, LSA, and BERT-based models in uncovering topics, assessing factors such as coherence, interpretability, semantic richness, and the depth of insights generated from both datasets.

2. To develop visualizations to represent the identified topics, enabling easier interpretation and understanding of consumer sentiments in smartphone reviews and trends in news articles.

3. To create an interactive website showcasing the project, allowing users to explore the datasets, view visualizations, and interact with topic modeling outputs for a comprehensive understanding of the analyzed data.

3.4 Scope

This project focuses on the analysis of text-based reviews from major e-commerce platforms. The scope includes:

1. Collect data by gathering a diverse set of customer reviews across various product categories and from various news portals.
2. Preprocess data by cleaning and preparing the review data, handling inconsistencies, removing irrelevant information, and standardizing formats for analysis.
3. Apply Topic Modeling techniques such as Latent Dirichlet Allocation (LDA) and BERT-based topic transformer models to identify and extract key themes and topics from the reviews.

It is to be noted that our project does not involve the development of new NLP models but aim to utilize existing models and tools.

4. Literature Review

4.1 Related work

In recent years, the integration of topic modeling with sentiment analysis has gained significant attention, particularly with models such as Latent Dirichlet Allocation (LDA) due to its efficacy in uncovering latent themes within text data. The paper by Akhmedov et al. (2021) [1] contributes to this growing body of research by introducing an LDA-based Topic/Document/Sentence (TDS) model that extends traditional sentiment analysis beyond the document level to include a finer-grained sentiment assessment at the word level. Building on the foundational work that surveys the applications and development of LDA in various domains as outlined by Jelodar et al. (2019)[2], the study by Akhmedov et al. (2021) applies LDA to develop a Topic/Document/Sentence (TDS) model for sentiment analysis, demonstrating its utility in extracting nuanced sentiment information at multiple levels of text granularity.

Maarif (2022)[3] utilized LDA for topic modeling in the analysis of user reviews on the Amazon platform for the Lenovo K8 Note smartphone. Their study emphasizes the use of text analysis techniques, including sentiment analysis, to efficiently summarize large volumes of user-generated content from social media and e-commerce platforms

Kwon et al. (2021)[4] employed LDA for topic modeling and sentiment analysis on online airline reviews. Their study identified significant topics such as 'seat', 'service', and 'meal', and revealed that delays negatively impacted customer satisfaction, while positive aspects like 'staff service' contributed to higher satisfaction levels.

Wang et al. (2022) [5] proposed a novel approach to sentiment analysis using LDA for topic modeling, applied to customer-generated online videos.

Grootendorst's paper [6] on BERTopic highlights its modular design with BERT embeddings, UMAP, HDBSCAN, and c-TF-IDF for superior topic modeling performance. It demonstrates coherence, diversity metrics, and dynamic topic modeling with time-based c-TF-IDF

Cheddak et al. [7] explore BERTopic's use in automating brainstorming sessions, showing its superiority over traditional models like LDA for semantically coherent topic generation and managing complex, multilingual data

4.2 Related theory

4.2.1 Topic Modeling

Topic Modeling is an unsupervised machine learning technique used to automatically discover the hidden thematic structure (topics) in a collection of texts (documents). It identifies patterns of words that frequently appear together and groups them into coherent clusters or topics.

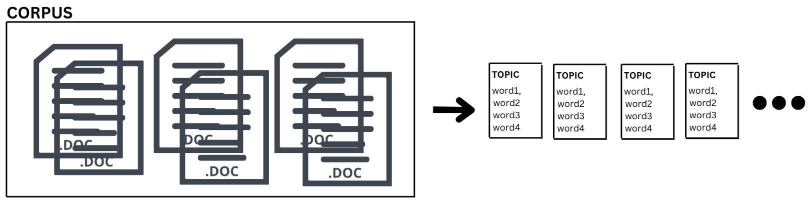


Figure 4.1: Topic Modeling.

4.2.2 Latent Dirichlet Allocation (LDA) and Variational Inference

In Latent Dirichlet Allocation (LDA), we aim to uncover hidden topics within a collection of documents by modeling each document as a mixture of topics and each topic as a mixture of words. The fundamental challenge is that the posterior distribution over topics and word assignments (given the observed words in each document) is intractable to compute directly.

Variational inference is used to approximate this complex posterior with a simpler, tractable distribution. The basic idea is to define an alternate distribution over the hidden variables (topic proportions and topic assignments) and find the best parameters for this alternate distribution by minimizing the divergence from the true posterior.

4.2.3 Step-by-Step Variational Inference Process in LDA

1. **Define a Simplified Variational Distribution**

   In LDA, we approximate the posterior $p(\theta, z \mid w, \alpha, \beta)$, where:

   - θ: topic proportions for each document.
   - z: topic assignment for each word.

- w: observed words in the document.
- α and β: hyperparameters for the Dirichlet priors on topics and words.

We define a variational distribution $q(\theta, z \mid \gamma, \phi)$ to approximate this posterior:

$$q(\theta, z \mid \gamma, \phi) = q(\theta \mid \gamma) \prod_{n=1}^{N} q(z_n \mid \phi_n),$$

where:

- $q(\theta \mid \gamma)$ is a Dirichlet distribution parameterized by γ, representing the approximate topic proportions for the document.
- $q(z_n \mid \phi_n)$ is a multinomial distribution over topics for each word w_n, with parameters ϕ_n representing the probability that word w_n belongs to each topic.

2. **Using Jensen's Inequality to Set Up a Lower Bound**
We need to estimate the log likelihood of the observed words $\log p(w \mid \alpha, \beta)$, but directly computing it is infeasible due to the intractable posterior. Applying Jensen's inequality, we establish a lower bound on $\log p(w \mid \alpha, \beta)$ in terms of our variational distribution q:

$$\log p(w \mid \alpha, \beta) \geq E_q \left[\log p(\theta, z, w \mid \alpha, \beta)\right] - E_q \left[\log q(\theta, z)\right].$$

The right side of this inequality, $L(\gamma, \phi; \alpha, \beta)$, serves as a lower bound on the log likelihood and is what we maximize in variational inference.

3. **Relating the Bound to KL Divergence**
The difference between the true log likelihood and the lower bound $L(\gamma, \phi; \alpha, \beta)$ is the KL divergence between the variational distribution q and the true posterior p:

$$\log p(w \mid \alpha, \beta) = L(\gamma, \phi; \alpha, \beta) + D_{KL} \left(q(\theta, z \mid \gamma, \phi) \parallel p(\theta, z \mid w, \alpha, \beta)\right).$$

Goal: Maximize $L(\gamma, \phi; \alpha, \beta)$ by finding optimal parameters γ and ϕ, which minimizes the KL divergence D_{KL} between q and the true posterior.

4. **Expanding the Lower Bound $L(\gamma, \phi; \alpha, \beta)$**
We expand $L(\gamma, \phi; \alpha, \beta)$ by taking expectations over terms in the joint probability $p(\theta, z, w \mid \alpha, \beta)$:

$$L(\gamma, \phi; \alpha, \beta) = E_q \left[\log p(\theta \mid \alpha)\right] + E_q \left[\log p(z \mid \theta)\right] + E_q \left[\log p(w \mid z, \beta)\right] - E_q \left[\log q(\theta)\right] - E_q \left[\log q(z)\right].$$

This equation breaks down the bound into interpretable components:

- Expectations of prior distributions (on θ, z, and w) under the variational distribution q.
- Entropy terms that encourage diversity in q.

5. **Optimization Step 1: Update for ϕ (Variational Multinomial)**
To maximize the lower bound with respect to ϕ, which represents topic assignments for each word, we use the Lagrangian approach due to the normalization constraint $\sum_i \phi_{ni} = 1$. Taking the derivative of the Lagrangian with respect to ϕ_{ni} and setting it to zero, we derive the update for ϕ:

$$\phi_{ni} \propto \beta_{iw_n} \exp\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right),$$

where:

- β_{iw_n}: Probability of word w_n under topic i.
- Ψ: Digamma function, the derivative of the log-gamma function.

This expression allows each word's topic assignment probability ϕ_{ni} to be updated based on the current topic proportions γ.

6. **Optimization Step 2: Update for γ (Variational Dirichlet)**
We next optimize γ, which represents the variational parameters for the Dirichlet distribution over topics in the document. By taking the derivative of $L(\gamma, \phi; \alpha, \beta)$ with respect to γ_i and setting it to zero, we derive the update for γ:

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni}.$$

This update shows that γ_i is influenced by both the prior α_i and the summed topic probabilities across words in the document (through ϕ_{ni}).

7. **Iterative Optimization (Coordinate Ascent)**
Because updates to γ and ϕ depend on each other, variational inference uses an iterative approach. Specifically:

- **E-step:** Update ϕ_{ni} for each word given the current γ.
- **M-step:** Update γ_i for each topic based on the updated ϕ_{ni}.

These updates continue until convergence, where the lower bound $L(\gamma, \phi; \alpha, \beta)$ stabilizes, indicating that the variational distribution q closely approximates the true posterior.

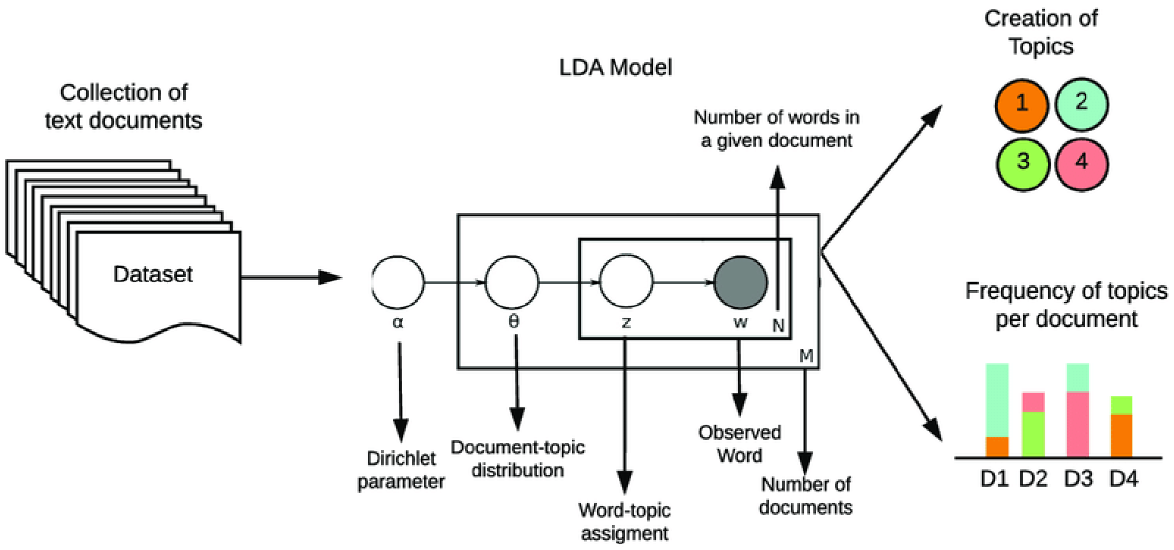


Figure 4.2: Latent Dirichlet Allocation

4.2.4 LSA - Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a powerful statistical technique widely used in the fields of natural language processing (NLP) and information retrieval. Its primary purpose is to uncover hidden relationships between words and documents within a corpus of text data. LSA helps to identify underlying concepts or topics present in a collection of documents.

The strength of LSA lies in its ability to reduce the dimensionality of the data. LSA helps to facilitate extraction of semantic structures that accurately reflect the content of the documents being analyzed by condensing information into a lower-dimensional space. This makes it a valuable tool for understanding large volumes of text.

Vector Space Model

In the vector space model, each document in a collection is represented as a term-document matrix. This matrix is structured as follows:

- Each row corresponds to a unique term (word) from the corpus.
- Each column corresponds to a specific document.
- The entries in the matrix represent the frequency of terms in the documents, often calculated using term frequency (TF) or term frequency-inverse document frequency (TF-IDF) scores.

This representation allows for the mathematical analysis of the relationships between terms and documents.

Dimensionality Reduction

The core of LSA involves a mathematical technique known as **singular value decomposition** (SVD). SVD decomposes the term-document matrix A into three matrices:

$$A = U\Sigma V^T$$

where:

- A is the original term-document matrix.
- U contains the left singular vectors, which can be interpreted as the *term-topic matrix*.
- Σ is a diagonal matrix of singular values, reflecting the strength of each identified topic.
- V^T contains the right singular vectors, representing the *document-topic matrix*.

For a high ranked matrix, we can approximate it by summation of various rank 1 matrices.
The singular values (scaling factors: σ) are ordered in descending order, i.e.,

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r,$$

where r is the rank of the original matrix A. Hence even if we only select two most important σ values i.e., we ignore other σ values, we can get approximate values of original matrix A as a result, dimension is reduced.

Calculating U, Σ, and V^T

1. **Covariance Matrices**:

$$AA^T \quad \text{and} \quad A^TA$$

These matrices are symmetric and help determine the eigenvectors and eigenvalues.

2. **Eigenvalue Decomposition**:

$$AA^T = U\Lambda U^T \quad \text{and} \quad A^TA = V\Lambda V^T$$

Here, Λ contains the eigenvalues.

3. **Singular Values ($\mathbf{\Sigma}$)**: Singular values σ_i are the square roots of the eigenvalues:

$$\sigma_i = \sqrt{\lambda_i}$$

The diagonal entries of Σ are $\sigma_1, \sigma_2, \ldots, \sigma_r$, where $r = \text{rank}(A)$.

4. **Left and Right Singular Vectors (U and V)**: - Columns of U are the eigenvectors of AA^T. - Columns of V are the eigenvectors of $A^T A$.

Final Decomposition

After calculating Σ, U, and V^T, the matrix A can be expressed as:

$$A = U\Sigma V^T$$

By truncating the matrices to keep only the top k singular values, LSA reduces the dimensionality of the data. This truncation captures the most significant semantic structures while filtering out noise, allowing for a clearer representation of the relationships among terms and documents.

Latent Semantic Structure

The reduced matrices produced by SVD provide a representation of the relationships between terms and documents in a lower-dimensional space. In this new space, topics are identified as clusters of words that frequently co-occur across different documents. Words that share similar meanings or contexts tend to be grouped together which helps in the identification of underlying themes within the text. This clustering allows to understand the primary topics present in a collection of documents which provides insights into the overall content and structure of the corpus.

4.2.5 BERTopic

BERTopic is a topic modeling technique designed to extract topics from large corpora of text. It combines transformers, such as BERT, with techniques like UMAP for dimensionality reduction and HDBSCAN for clustering to generate meaningful, interpretable topics. The model leverages embeddings from transformer models for accurate semantic understanding, and uses Class-based TF-IDF (C-TF-IDF) to generate key terms for each topic. This approach is effective for extracting topics that are both coherent and interpretable, and it can be adapted to various text domains such as reviews, news articles, or scientific papers.

The sections highlighted are the models/containers used for the scope of this project.

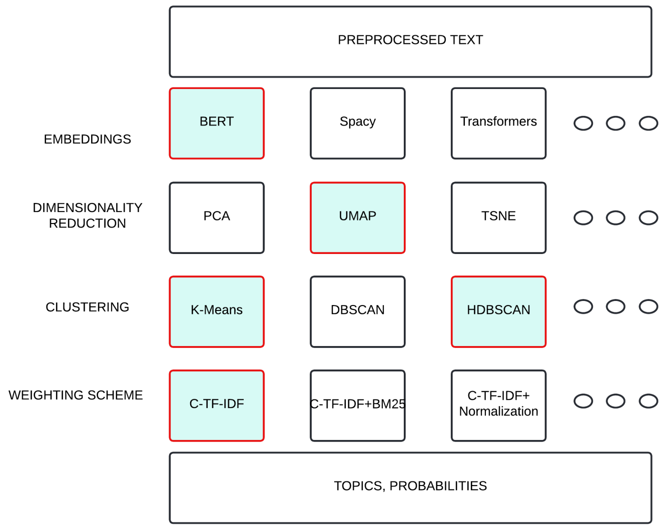


Figure 4.3: Pipeline for BERTopic

4.2.6 Embedding

Embedding is a numerical representation of objects, such as words, sentences, or documents, in a continuous vector space, where semantically similar objects are represented by vectors that are closer together in the space. For the encoding portion, we chose the *paraphrase-MiniLM-L3-v2* model. This model is a compact variant of MiniLM (Miniature Language Model), specifically designed for generating high-quality sentence embeddings suitable for tasks such as paraphrase detection, clustering, and semantic similarity evaluation.

The *paraphrase-MiniLM-L3-v2* model provides embeddings that are particularly efficient for computing pairwise cosine similarity, enabling semantic analysis of sentences with high accuracy. These embeddings are utilized effectively in paraphrase mining tasks to identify semantically similar sentences within large text corpora.

This compact architecture ensures both speed and accuracy, making it a suitable choice for applications requiring high-performance natural language understanding while maintaining computational efficiency.

As per Hugging Face official documentation on Pretrained Models and paper: [8]:

4.2.7 Uniform Manifold Approximation and Projection (UMAP) for Dimensionality Reduction

UMAP is a dimensionality reduction algorithm that effectively captures both local and global structure in high-dimensional data. Below is an outline of UMAP's main steps, from similarity calculations in high dimensions to the optimization process in low-dimensional

Description	All-round model tuned for many use-cases.
Base Model	nreimers/MiniLM-L3-H384-uncased
Max Sequence Length	128
Dimensions	384
Normalized Embeddings	False
Suitable Score Functions	Cosine-similarity (util.cos_sim)
Size	61 MB
Pooling	Mean Pooling
Training Data	AllNLI, sentence-compression, SimpleWiki, altlex, msmarco-triplets, quora_duplicates, coco_captions, flickr30k_captions, yahoo_answers_title_question, S2ORC_citation_pairs, stackexchange_duplicate_questions, wiki-atomic-edits

Table 4.1: Model Details for paraphrase-MiniLM-L3-v2

space.

Step 1: Calculating High-Dimensional Similarity

UMAP starts by constructing a k-nearest neighbor (k-NN) graph to represent the local relationships among points in the high-dimensional space.

Construct the k-NN Graph For each point x_i in the dataset, find its k nearest neighbors based on a chosen distance metric, such as Euclidean distance.

Compute Fuzzy Memberships For each neighboring pair (x_i, x_j), define a probability that captures how strongly x_j belongs to the "neighborhood" of x_i:

$$p_{ij} = \exp\left(-\frac{d_{ij} - \rho_i}{\sigma_i}\right)$$

where:

- d_{ij} is the distance between x_i and x_j.
- ρ_i adjusts for local density by setting a minimum distance to the nearest neighbor.
- σ_i controls the spread of x_i's neighborhood.

Step 2: Creating Symmetric Fuzzy Memberships

Since the fuzzy memberships p_{ij} and p_{ji} may differ, UMAP combines them to create a symmetric similarity score s_{ij} for each pair:

$$s_{ij} = p_{ij} + p_{ji} - p_{ij} \cdot p_{ji}$$

This expression computes the probability that either x_i or x_j considers the other as a neighbor, adjusted for overlap between memberships. This symmetric fuzzy membership matrix represents a balanced similarity structure in the high-dimensional space.

Step 3: Initializing the Low-Dimensional Embedding

To begin the embedding process, UMAP initializes each point y_i in a low-dimensional space (typically 2D or 3D).

Spectral Embedding or Random Initialization UMAP may initialize points using spectral decomposition on the k-NN graph, ensuring that nearby points in the high-dimensional space start close in the low-dimensional space. Alternatively, random initialization can be used for faster results on large datasets.

Step 4: Defining Low-Dimensional Similarity

To match the high-dimensional similarity structure, UMAP defines a similarity measure q_{ij} between each pair of points (y_i, y_j) in the low-dimensional space:

$$q_{ij} = \frac{1}{1 + a\|y_i - y_j\|^{2b}}$$

where:

- a and b are hyperparameters controlling the shape of the similarity distribution in low-dimensional space.
- $\|y_i - y_j\|$ is the Euclidean distance between y_i and y_j.

Step 5: Optimizing the Low-Dimensional Embedding with Stochastic Gradient Descent

UMAP aligns s_{ij} and q_{ij} by minimizing a cross-entropy loss function, effectively creating a low-dimensional representation that preserves high-dimensional relationships.

Define the Cross-Entropy Loss

$$\text{Loss} = \sum_{i \neq j} \left(s_{ij} \log \frac{s_{ij}}{q_{ij}} + (1 - s_{ij}) \log \frac{1 - s_{ij}}{1 - q_{ij}} \right)$$

This function penalizes discrepancies between s_{ij} (high-dimensional similarity) and q_{ij} (low-dimensional similarity), encouraging similar points to stay close and dissimilar points to move apart.

Calculate the Gradient For each point y_i, the gradient of q_{ij} with respect to y_i is:

$$\frac{\partial q_{ij}}{\partial y_i} = -\frac{2ab\|y_i - y_j\|^{2b-2}(y_i - y_j)}{(1 + a\|y_i - y_j\|^{2b})^2}$$

The gradient of the loss function with respect to y_i is then:

$$\frac{\partial \text{Loss}}{\partial y_i} = \sum_{j \neq i} \left(\frac{s_{ij} - q_{ij}}{q_{ij}(1 - q_{ij})} \right) \cdot \frac{\partial q_{ij}}{\partial y_i}$$

Update with Stochastic Gradient Descent UMAP uses stochastic gradient descent (SGD) to iteratively adjust y_i to minimize the loss:

$$y_i \leftarrow y_i - \eta \frac{\partial \text{Loss}}{\partial y_i}$$

where η is the learning rate. This process is repeated for all pairs until the loss converges or a set number of iterations is reached.

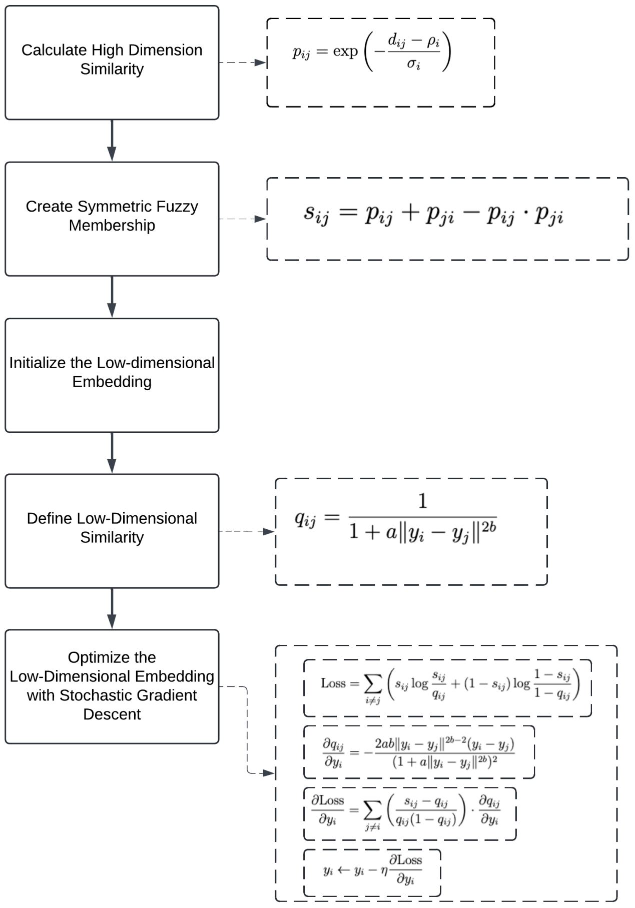

Calculate High Dimension Similarity
$p_{ij} = \exp\left(-\frac{d_{ij} - \rho_i}{\sigma_i}\right)$
Create Symmetric Fuzzy Membership
$s_{ij} = p_{ij} + p_{ji} - p_{ij} \cdot p_{ji}$
Initialize the Low-dimensional Embedding
Define Low-Dimensional Similarity
$q_{ij} = \frac{1}{1 + a\|y_i - y_j\|^{2b}}$
Optimize the Low-Dimensional Embedding with Stochastic Gradient Descent
$\text{Loss} = \sum_{i \neq j} \left(s_{ij} \log \frac{s_{ij}}{q_{ij}} + (1 - s_{ij}) \log \frac{1 - s_{ij}}{1 - q_{ij}} \right)$
$\frac{\partial q_{ij}}{\partial y_i} = -\frac{2ab\|y_i - y_j\|^{2b-2}(y_i - y_j)}{(1 + a\|y_i - y_j\|^{2b})^2}$
$\frac{\partial \text{Loss}}{\partial y_i} = \sum_{j \neq i} \left(\frac{s_{ij} - q_{ij}}{q_{ij}(1 - q_{ij})} \right) \cdot \frac{\partial q_{ij}}{\partial y_i}$
$y_i \leftarrow y_i - \eta \frac{\partial \text{Loss}}{\partial y_i}$

Figure 4.4: UMAP Algorithm FLowchart

4.2.8 K-means Clustering Algorithm

The K-means clustering algorithm is a widely used technique for partitioning a dataset into k clusters. Once the texts are passed through embeddings, and dimensionality reduction through UMAP, we can use K-means for clustering. The algorithm follows these steps:

1. **Initialization:**
   Choose the number of clusters k. Randomly initialize k cluster centroids, denoted as $\mu_1, \mu_2, \ldots, \mu_k$, in the feature space.

2. **Assignment Step:**
   Assign each data point x_i to the nearest cluster. For each data point x_i, compute its distance from each cluster centroid μ_j using Euclidean distance (Euclidean distance is common, but other metrics like Manhattan or cosine distance can be used based on

the problem.):

$$d(x_i, \mu_j) = \|x_i - \mu_j\|_2 = \sqrt{\sum_{m=1}^{M} (x_{i,m} - \mu_{j,m})^2}$$

Assign x_i to the cluster C_j with the closest centroid:

$$\text{Cluster}(x_i) = \arg\min_j d(x_i, \mu_j)$$

3. **Update Step:**
   Recalculate the centroids for each cluster. For each cluster C_j, the centroid μ_j is updated to the mean of all points assigned to that cluster:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

   where $|C_j|$ is the number of data points in cluster C_j.

4. **Convergence Check:**
   Repeat the assignment and update steps until the centroids no longer change significantly or a maximum number of iterations is reached. The convergence criterion is:

$$\|\mu_j^{\text{new}} - \mu_j^{\text{old}}\|_2 < \epsilon \quad \forall j$$

   where ϵ is a small threshold.

5. **Objective Function:**
   The K-means algorithm minimizes the within-cluster sum of squared errors (WCSS):

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|_2^2$$

   At convergence, the value of J is minimized.

6. **Output:**
   The final output consists of:

   - The final cluster assignments for each data point.
   - The centroids of the clusters.

4.2.9 HDBSCAN Algorithm

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is an advanced clustering algorithm that builds upon DBSCAN by providing a hierarchical clustering structure and identifying clusters with varying densities. The steps of the algorithm are outlined below:

1. **Input Parameters:**
   The algorithm takes the following inputs:
   - **Dataset:** A set of n data points in a feature space.
   - **MinPts:** The minimum number of points required to form a dense region. .

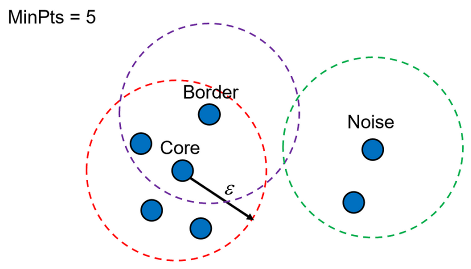


Figure 4.5: Samples in Feature space for HDBSCAN

2. **Compute Core Distances:**
   For each data point p, calculate the core distance, which is the distance to its MinPts-th nearest neighbor:

$$\text{Core Distance}(p) = \text{distance to the MinPts-th nearest neighbor of } p$$

   If a point has fewer than MinPts neighbors, it is considered a sparse point.

3. **Construct Mutual Reachability Distance:**
   The mutual reachability distance between two points p and q is defined as:

$$\text{Mutual Reachability}(p, q) = \max\{\text{Core Distance}(p), \text{Core Distance}(q), \text{Distance}(p, q)\}$$

   This metric ensures that pairwise distances incorporate both density and spatial relationships.

4. **Build the Minimum Spanning Tree (MST):**
   Treat the points as nodes in a graph where edges represent the mutual reachability distances. Construct a minimum spanning tree (MST), which is a tree that connects all points while minimizing the sum of mutual reachability distances.

5. **Generate a Hierarchical Cluster Tree:**

   (a) Start at the root with all points in a single cluster.

   (b) Recursively remove the largest edge (based on mutual reachability distance) in the MST to split clusters.

   (c) Record splits to form a dendrogram representing the hierarchical structure of the clusters.

6. **Extract Stable Clusters:**
   Use *persistence* to identify clusters. Persistence measures how long a cluster remains intact across different levels of the hierarchy. Clusters with higher persistence are considered stable, and less persistent clusters are treated as noise or discarded.

7. **Assign Noise Points:**
   Points that are not part of any stable cluster are labeled as noise. Noise points are retained in the dataset but marked separately.

8. **Output:**
   The algorithm outputs:

   - **Clusters:** A set of clusters identified from the hierarchy.
   - **Noise Points:** Points classified as noise.
   - **Cluster Hierarchy:** The dendrogram showing how clusters are split and merged.

4.2.10 C-TF-IDF: Class-Based Term Frequency-Inverse Document Frequency

C-TF-IDF is an adaptation of the traditional TF-IDF, tailored for identifying representative terms for groups of documents (classes) rather than individual documents. In the context of BERTopic, it is used to extract interpretable topics from clusters of documents.

Traditional TF-IDF Recap

Term Frequency (TF): Measures how often a term t appears in a document d:

$$\mathrm{TF}(t,d) = \frac{\text{Number of occurrences of } t \text{ in } d}{\text{Total number of terms in } d}$$

Inverse Document Frequency (IDF): Measures the importance of a term across the entire corpus D, giving less weight to common terms:

$$\text{IDF}(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

TF-IDF: The product of TF and IDF, providing a weighted measure of a term's importance in a document:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D)$$

Class-Based TF-IDF (C-TF-IDF)

Instead of calculating term importance for individual documents, C-TF-IDF aggregates all documents within a cluster (class) into a single pseudo-document and measures term importance within that class compared to the entire corpus.

Mathematical Representation of C-TF-IDF: Let:

- c: A class (a cluster of documents).
- D: The entire corpus.
- t: A term.

Class-Based Term Frequency (C-TF): The term frequency for a term t in a class c:

$$\text{C-TF}(t, c) = \frac{\text{Number of occurrences of } t \text{ in class } c}{\text{Total number of terms in class } c}$$

Class-Based Inverse Document Frequency (C-IDF): The inverse document frequency for a term t across all classes:

$$\text{C-IDF}(t, D) = \log \frac{|D|}{1 + |\{c \in C : t \in c\}|}$$

Here, $|\{c \in C : t \in c\}|$ counts the number of clusters where t appears.

C-TF-IDF: The product of the two terms:

$$\text{C-TF-IDF}(t, c, D) = \text{C-TF}(t, c) \cdot \text{C-IDF}(t, D)$$

This quantifies the importance of a term t in a cluster c, relative to the entire corpus D.

C-TF-IDF in BERTopic

In BERTopic, C-TF-IDF plays a crucial role in identifying representative terms for each topic (cluster):

Input:

- Each cluster is treated as a single class c, combining the text from all documents assigned to the cluster into a single pseudo-document.
- The algorithm calculates the frequency of each term in these pseudo-documents.

Computation:

- Terms are ranked by their C-TF-IDF values within each cluster.
- The top n terms (e.g., top 10) are selected to represent the topic.

Output:

- A list of keywords for each cluster that define the topic in a human-interpretable manner.
- The keywords are distinctive to the cluster and provide insight into the thematic structure of the data.

4.2.11 Performance Metrics

Coherence Score

The coherence score is used to measure the quality of topics generated by a topic model, such as LDA or BERTopic. A high coherence score indicates that the words in the topic are semantically related and likely to appear together in a document. The Coherence Score (C_v) is calculated based on pairwise similarities of the most frequent words in a topic, and is widely used for evaluating topic models.

Formula for Coherence Score (C_v): Given a topic with the top N words, the coherence score C_v is calculated as the average of pairwise similarities between these words. The similarity between words is measured using Pointwise Mutual Information (PMI) within a sliding window of text.

Pointwise Mutual Information (PMI): PMI measures how much two words co-occur in a given context. The formula for PMI between two words w_i and w_j is:

$$PMI(w_i, w_j) = \log\left(\frac{P(w_i, w_j)}{P(w_i)P(w_j)}\right)$$

Where:

- $P(w_i, w_j)$ is the probability of both words occurring together in the same context window.
- $P(w_i)$ and $P(w_j)$ are the individual probabilities of the words occurring independently.

C_v Coherence Score: The coherence score C_v is calculated as the average PMI for all word pairs in the top N words of a topic:

$$C_v = \frac{1}{N(N-1)} \sum_{i \neq j} PMI(w_i, w_j)$$

Where:

- N is the number of words in the topic.
- The sum is taken over all pairs of words w_i and w_j in the top N words of the topic.

A high coherence score suggests that the words in a topic are semantically related, which makes the topic more interpretable and meaningful.

Topic Diversity

Topic diversity refers to the distinctiveness or variety of topics generated by a topic modeling algorithm. It quantifies how different or unique the topics are from each other. High diversity in topics means that the topics cover a broad range of different themes, while low diversity suggests that the topics are repetitive or redundant.

Measuring topic diversity is important for ensuring that the model does not generate overly similar or redundant topics. One common method for evaluating topic diversity is to compute the average pairwise distance or dissimilarity between the top words of each topic. This distance is typically calculated using cosine similarity or other distance metrics. A higher average dissimilarity between topics indicates greater diversity.

The formula for calculating topic diversity is as follows:

$$\text{Topic Diversity} = \frac{1}{T(T-1)} \sum_{i \neq j} \text{Distance}(T_i, T_j)$$

Where:

- T is the total number of topics.
- T_i and T_j are the sets of top words for topics i and j, respectively.

- Distance(T_i, T_j) is the dissimilarity between the two topics, often computed using cosine distance.

Relation to Coherence: Topic diversity is related to the coherence score in that while increasing the number of topics might improve diversity, it could reduce coherence, making topics less meaningful and more overlapping. According to Mimno et al. (2011) [9] and Blei (2003) [10], this trade-off is common in topic modeling, where optimizing for one metric often results in a reduction in the other.

Silhouette Score

The Silhouette Score is a widely used metric for evaluating the quality of clustering results. It provides a way to assess how well each data point fits into its assigned cluster and how distinct the clusters are from each other. The score ranges from -1 to +1, where:

- +1 indicates that the points are well clustered, with the distance between clusters being large.
- 0 indicates that the points are on or very close to the decision boundary between clusters.
- -1 suggests that the points may have been assigned to the wrong cluster.

Formula

The Silhouette Score for each data point is computed as:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Where:

- $a(i)$ is the average distance between the point i and all other points within the same cluster. This is a measure of how close the point is to other points in the same cluster.
- $b(i)$ is the average distance between the point i and all points in the nearest cluster to which point i does not belong. This measures how close the point is to the next nearest cluster.

Use in Clustering

The Silhouette Score is particularly useful in clustering because it gives both an intra-cluster measure (how close points are within the same cluster) and an inter-cluster measure (how far apart clusters are from each other).

5. Methodology

Data Collection

We have used three datasets whose information is further explained in section (4.1).

Data Preprocessing

Text Cleaning and Normalization: Text cleaning and normalization is a foundational step in this project.To start, a function, expand_contractions, is used to expand common English contractions (e.g., changing ”can't” to ”cannot”) to ensure that similar terms are treated consistently. All text is then converted to lowercase, making it easier to match similar terms without case sensitivity. Next, unwanted elements like punctuation, URLs, and words with digits are removed to focus the analysis on meaningful content. Several specific functions are defined for this purpose: remove_punctuations eliminates punctuation marks and newline characters, while remove_urls uses regular expressions to strip URLs from the text. Additionally, remove_words_with_digits filters out any words containing numeric characters, helping to eliminate irrelevant information such as product codes or other numbers that do not contribute to the analysis

Stopword Removal: Commonly used words (stopwords) that carry minimal contextual information, such as ”the” and ”is,” are removed from the dataset. Additional domain-specific or unnecessary words are added to this list to further refine the content, helping focus the analysis on relevant terms.

Tokenization: The cleaned text is split into individual words, or tokens, which allows for more granular analysis of each word in the text.Tokenization is essential for subsequent steps, such as stemming and lemmatization.

Stemming and Lemmatization: Stemming uses simple rules to cut off word endings, reducing words to a rough ”root” form. For example, it might shorten ”running” and ”runner” to ”run,” but the output can sometimes be incomplete or distorted (e.g., stemming ”better” to ”bett”). Lemmatization, on the other hand, is a more sophisticated process that reduces words to their base or dictionary form, considering the context and part of speech. In our project, we used lemmatization rather than stemming, as lemmatization produces more linguistically accurate and interpretable words. This is advantageous for tasks like topic modeling, where precise word forms enhance the model's ability to recognize core concepts and relationships.

Vectorization: The cleaned and processed text data is transformed into a numerical format using a vectorizer. This vectorization step produces a term-document matrix, where each document is represented as a vector of word counts or term frequencies. This matrix serves as the input for machine learning models, enabling algorithms to analyze the text data in a structured, quantifiable way.

Topic Modeling

In this project, we first trained Latent Dirichlet Allocation (LDA), followed by Latent Semantic Analysis (LSA), and finally BERTopic. For LDA, we focused on optimizing the number of topics and passes through hyperparameter tuning to achieve better coherence scores and topic diversity. Next, we trained LSA, where we experimented with the number of components and other parameters to extract meaningful topics. Lastly, we applied BERTopic, which uses transformer-based embeddings and dimensionality reduction techniques such as UMAP to generate more coherent topics. With BERTopic, we also explored hyperparameter tuning, adjusting the number of topics, clustering settings, and other parameters to improve the model's overall performance.

Model Evaluation

To evaluate the performance of the models, we used two primary metrics: coherence score (c_o) and topic diversity (t_d). The coherence score measures how well the words in a topic co-occur in the corpus, with higher values indicating better topic quality. Topic diversity, on the other hand, assesses how different the topics are from each other, with higher diversity representing more distinct topics.

We compared the c_o and t_d values for each model to determine which model produced the best topics. Additionally, we compared the topic modeling results with clustering performance, using the silhouette score to measure how well-separated the clusters are. A higher silhouette score indicates that the clusters are more distinct and well-defined, which is important for topic coherence and separation. By comparing these metrics, we were able to assess the quality of the topics generated by each model and choose the most effective one for our task.

6. Experimental Setup

6.1 Dataset

For this project, we utilized three freely available datasets from Kaggle:

1. A dataset of 1.4 million cell phone reviews.
2. A Global News dataset.
3. Lenovo K8 Review dataset from Amazon.

Initially, our focus was solely on e-commerce reviews. However, after observing the model's strong performance on datasets like news, which can be easily categorized into distinct topics, we decided to include the news dataset as well. For training, we sampled 10,000 documents from each dataset.

6.1.1 1.4 million cell phone reviews Dataset (Dataset I)

The dataset [11] contains 1.4 million user reviews and ratings of various cell phone brands, focusing on performance, features, and design.For our purpose, we have selected the products for Samsung. It offers valuable insights into consumer preferences.

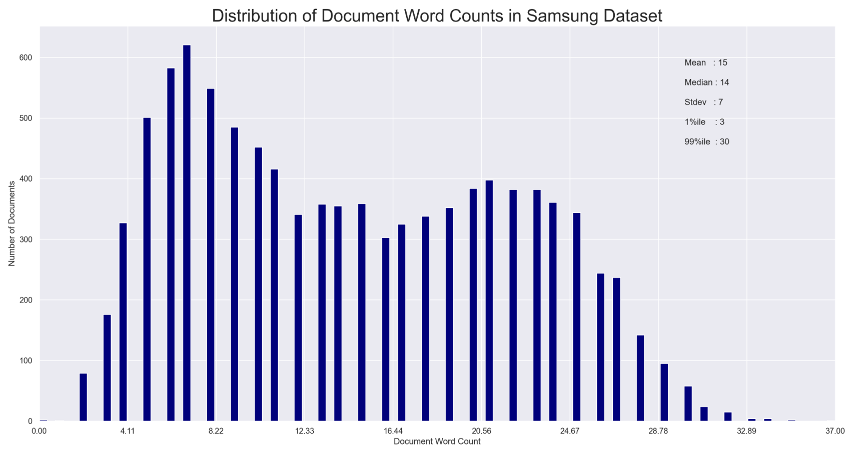


Figure 6.1: Words Distribution for dataset I

6.1.2 Global News Dataset (Dataset II)

The dataset [12] includes the following features:

- **Text of News Articles:** The main content of the news articles.
- **Publication Date and Time:** Timestamps indicating when each article was published.
- **Source Information:** The news source or publication that released the article.
- **Additional Metadata:** Any other metadata available through the NewsAPI.

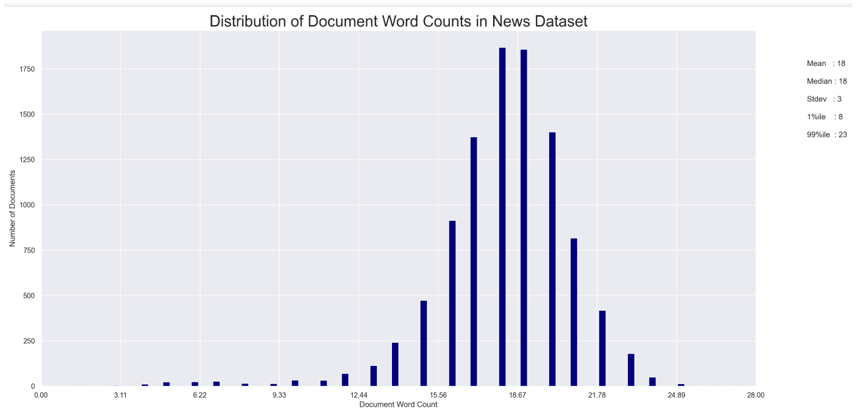


Figure 6.2: Words Distribution for dataset II

6.1.3 Lenove K8 Review amazon (Dataset III)

This dataset contains the reviews/ feedbacks for the lenovo product.

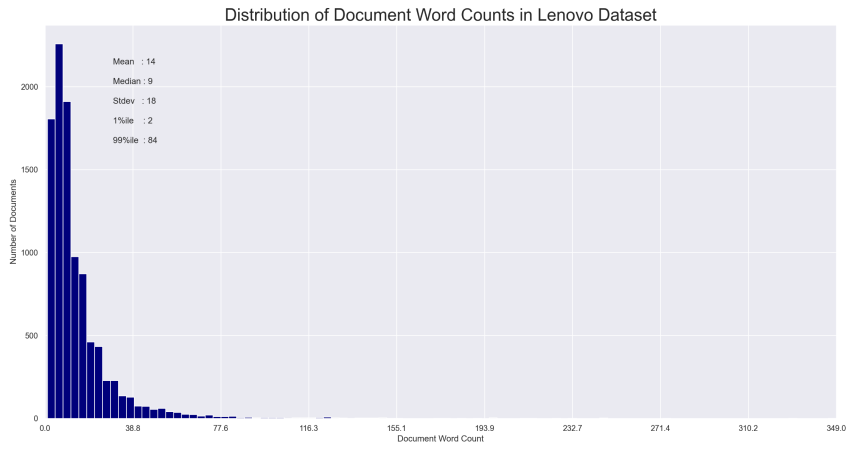


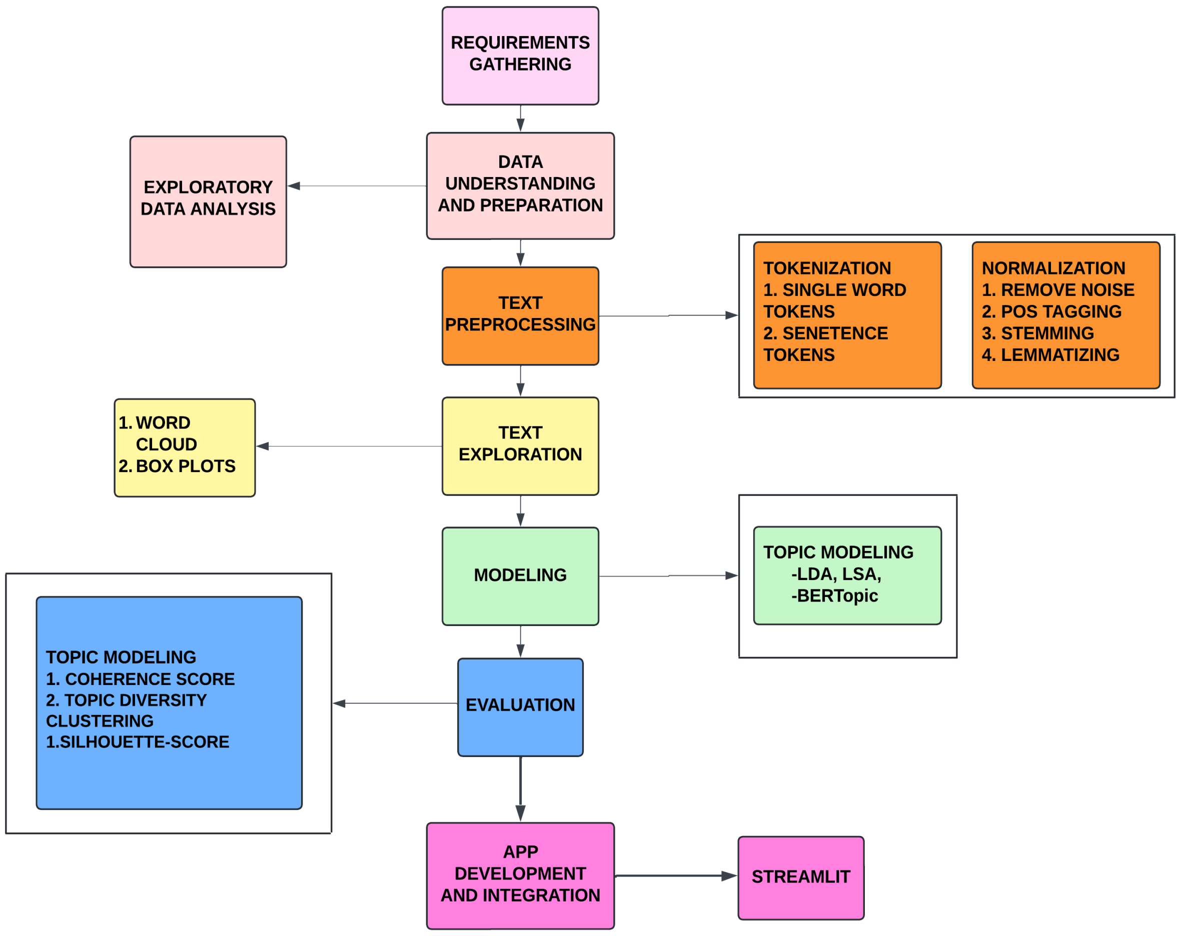
Figure 6.3: Words Distribution for dataset II

6.2 Software and Libraries

The software and libraries used in this project are selected to support the effective processing and analysis of e-commerce reviews. The following components will be utilized:

- **Programming Language:** Python 3.8 or later, due to its rich ecosystem of libraries for natural language processing (NLP) and data analysis.
- **Development Environment:** Jupyter Notebook or Google Colab which offer an interactive environment for coding, testing, and visualization.
- **NLP Libraries:**
  - **NLTK (Natural Language Toolkit):** Useful for basic text processing tasks such as tokenization, stopword removal, and stemming/lemmatization.
  - **spaCy:** Useful for advanced text processing, including named entity recognition and dependency parsing.
  - **Gensim:** Useful for implementation of Topic Modeling techniques, particularly Latent Dirichlet Allocation (LDA).
  - **Scikit-learn:** Useful for machine learning tasks, including model evaluation and feature extraction.
- **Deep Learning Libraries: TensorFlow** has been used for developing and training deep learning models.
- **Data Visualization Tools:**
  - **Matplotlib and Seaborn:** Useful for creating visual representations of topics and sentiments.
  - **WordCloud:** Useful for generating word cloud visualizations to represent the most frequent terms within identified topics.

7. System design



REQUIREMENTS GATHERING
DATA UNDERSTANDING AND PREPARATION
EXPLORATORY DATA ANALYSIS
TEXT PREPROCESSING
TOKENIZATION 1. SINGLE WORD TOKENS 2. SENETENCE TOKENS
NORMALIZATION 1. REMOVE NOISE 2. POS TAGGING 3. STEMMING 4. LEMMATIZING
TEXT EXPLORATION
1. WORD CLOUD 2. BOX PLOTS
MODELING
TOPIC MODELING -LDA, LSA, -BERTopic
EVALUATION
TOPIC MODELING 1. COHERENCE SCORE 2. TOPIC DIVERSITY CLUSTERING 1.SILHOUETTE-SCORE
APP DEVELOPMENT AND INTEGRATION
STREAMLIT

Figure 7.1: System Design

8. Results & Discussion

8.1 Traditional Methods

8.1.1 Latent Dirichlet Allocation (LDA)

Effect of Number of Passes and Number of Topics on Coherence Score

In Latent Dirichlet Allocation (LDA), the number of passes and the number of topics are important hyperparameters that significantly impact the coherence score, which is used to evaluate the quality of the topic model. Below, we discuss the effect of each parameter in detail.

Number of Passes (Iterations) **Effect on Convergence:** The number of passes determines how many times the model iterates over the entire corpus to optimize the topics. More passes give the model additional chances to refine the topic distributions for each document, generally improving convergence and stability.

Impact on Coherence Score: Increasing the number of passes can help the model find more consistent and meaningful topics, particularly in cases where the dataset is large or complex. However, after a certain point, increasing the number of passes has diminishing returns on coherence, as the model reaches saturation. Excessive passes may even lead to overfitting, potentially reducing coherence or producing less generalizable topics.

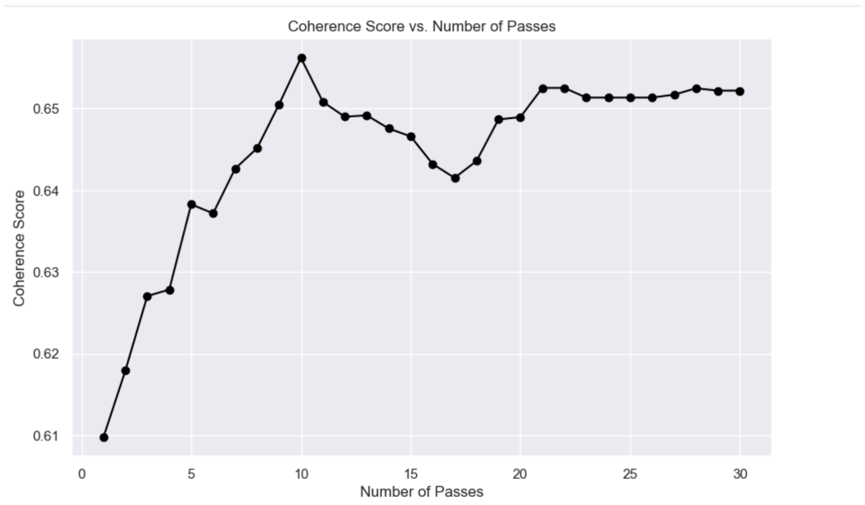


Figure 8.1: Effect of Number of Passes on Coherence Score

Number of Topics **Effect on Granularity:** The number of topics affects the granularity of the topics generated. A higher number of topics typically means that each topic covers a narrower theme, while fewer topics produce broader themes.

Impact on Coherence Score: The coherence score often increases as the model finds an ideal balance in the number of topics. Too few topics may combine distinct themes into a single topic, resulting in low coherence. Conversely, too many topics may produce overly specific or fragmented topics, also lowering coherence. A moderate number of topics often yields the best coherence score by grouping similar terms effectively while maintaining distinct, interpretable topics.

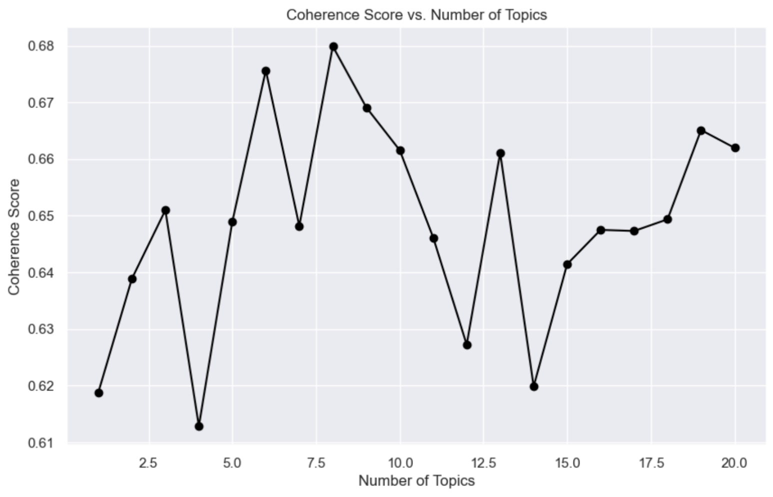


Figure 8.2: Effect of Number of Topics on Coherence Score

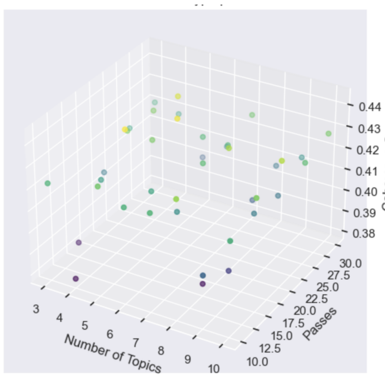
Hyperparameter Tuning

Given that both the number of passes and the number of topics influence the coherence score, we conduct hyperparameter tuning by testing various combinations of these two parameters to identify the optimal settings.

The parameters after hyperparameter tuning for each dataset are summarized in Table 8.1.

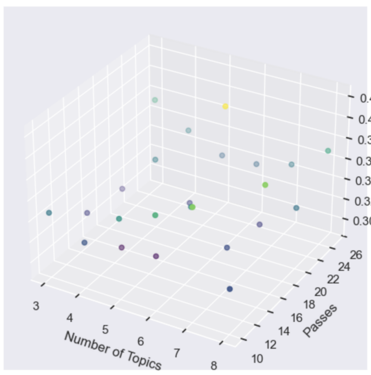
Table 8.1: Parameters After Hyperparameter Tuning

Dataset	**Best Number of Topics**	**Best Number of Passes**
Dataset I	5	15
Dataset II	6	20
Dataset III	6	20



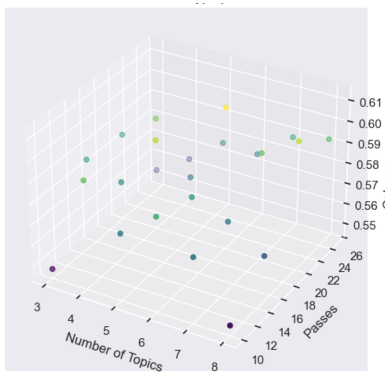
```
Best Number of Topics: 5
Best Number of Passes: 15
Best Coherence Score: 0.4428
```

(a) Hyper-parameter Tuning (Dataset I)



```
Best Number of Topics: 6
Best Number of Passes: 20
Best Coherence Score: 0.4208
```

(b) Hyper-parameter Tuning (Dataset II)



```
Best Number of Topics: 6
Best Number of Passes: 20
Best Coherence Score: 0.6117
```

(c) Hyper-parameter Tuning (Dataset III)

Figure 8.3: Hyper-parameter Tuning for Datasets I, II, and III

Scores

The scores for LDA are presented in Table 8.2. The table contains coherence score (`c_o`) and topic diversity (`t_d`) for each dataset.

Table 8.2: Scores for LDA across datasets

Metric	Dataset I	Dataset II	Dataset III
c_o	0.4428	0.4208	0.6117
t_d	0.6972	0.8142	0.7862

Topics Identified after training.

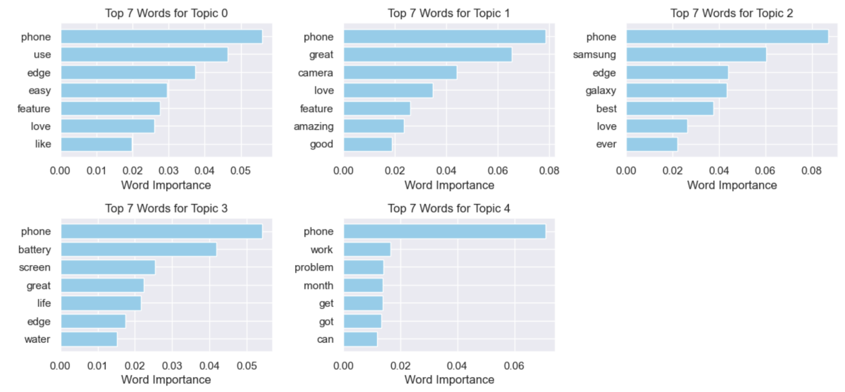


Figure 8.4: Top words per topic (dataset I)

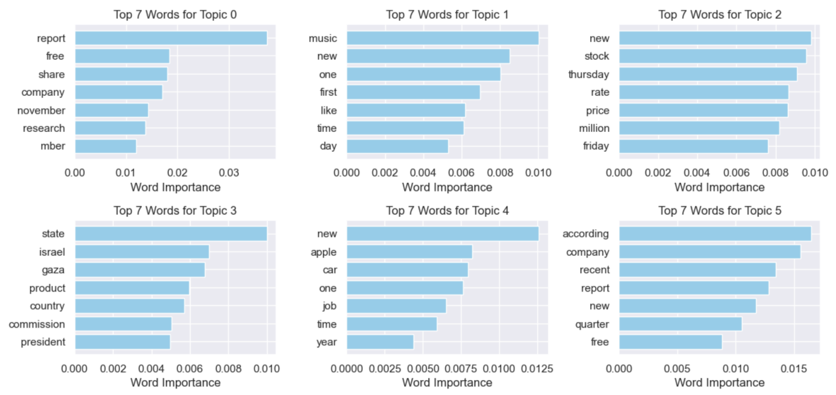


Figure 8.5: Top words per topic (dataset II)

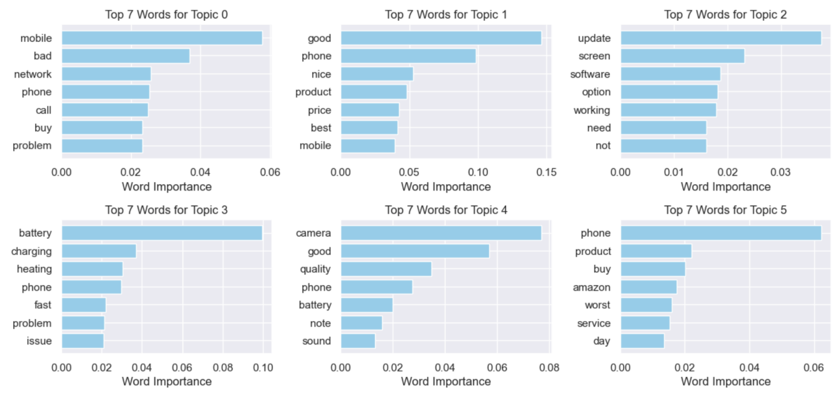


Figure 8.6: Top words per topic (dataset III)

Summary for LDA

The results show that LDA can generate reasonably good topic coherence and diversity when hyperparameters are tuned effectively. However, further improvements could still be made, particularly for Datasets I and II, where a lower coherence score was observed compared to Dataset III.

8.1.2 Latent Semantic Analysis (LSA)

Hyperparameter Tuning

The number of topics (`n_components`) in LSA significantly impacts the coherence score (`c_v`). Selecting an optimal number balances specificity and coverage, ensuring meaningful topics. Below is a summary of the best-performing topic numbers for different datasets:

Dataset	Optimal Topics	Coherence Score
Dataset I	3	0.5158
Dataset II	3	0.8909
Dataset III	7	0.4661

Table 8.3: Optimal Number of Topics and Coherence Scores.

Scores

The scores for LSA are presented in Table 8.4. The table contains coherence score (`c_o`) and topic diversity (`t_d`) for each dataset.

Table 8.4: Scores for LSA across datasets

Metric	Dataset I	Dataset II	Dataset III
c_o	0.5158	0.8909	0.4661
t_d	0.7333	0.7333	0.5714

8.2 BERTopic

8.2.1 Embedding and Dimensionality Reduction

For this project, we used the `MiniLM-L3-v2` model. This model is a compact variant of MiniLM (Miniature Language Model), specifically designed for generating high-quality sentence embeddings, which are suitable for tasks such as paraphrase detection, clustering, and semantic similarity evaluation. The reason for choosing this model is its relatively smaller number of parameters and computational efficiency compared to other models. However, for potentially better embeddings, other transformer-based models can be considered.

To reduce the dimensionality of the sentence embeddings and improve the interpretability of the data, we applied UMAP (Uniform Manifold Approximation and Projection). It is particularly well-suited for high-dimensional data like sentence embeddings, as it maintains the relationships between data points while reducing their dimensionality. UMAP was chosen for its computational efficiency and ability to handle large datasets, providing clear, well-separated clusters for downstream analysis.

8.2.2 Clustering, Hyperparameter Tuning, and Silhouette Score

Details on clustering, hyperparameter tuning, and silhouette score for BERTopic are presented. The results are summarized in Tables 8.5, 8.6, and 8.7.

Table 8.5: Clustering Results for Dataset I

Parameter	**Value**
Algorithm	KMeans
n_clusters	7
Silhouette Score	0.37041
min_cluster_size	NaN
min_samples	NaN
cluster_selection_method	NaN

Table 8.6: Clustering Results for Dataset II

Parameter	**Value**
Algorithm	HDBSCAN
n_clusters	NaN
Silhouette Score	0.66701
min_cluster_size	20
min_samples	8
cluster_selection_method	eom

Table 8.7: Clustering Results for Dataset III

Parameter	**Value**
Algorithm	KMeans
n_clusters	5
Silhouette Score	0.345332
min_cluster_size	NaN
min_samples	NaN
cluster_selection_method	NaN

8.2.3 Scores

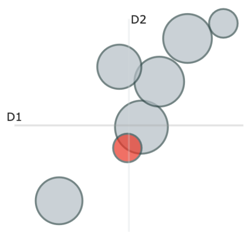
After selecting the best cluster parameters (according to the hyper-parameter turning-can be further improved), the model pipeline (BERTopic) was fitted with these parameters, and the scores for BERTopic are presented in Table 8.8. The table contains coherence score (`c_o`) and topic diversity (`t_d`) for each dataset.

Table 8.8: Scores for BERTopic across datasets

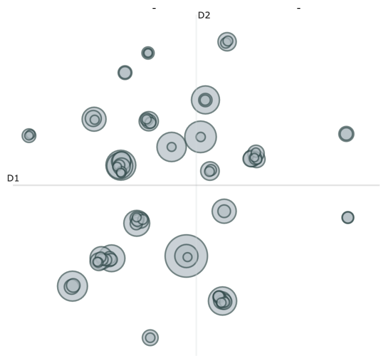
Metric	Dataset I	Dataset II	Dataset III
c_o	0.4729	0.6242	0.4534
t_d	0.5142	0.8662	0.6

8.2.4 Inter-Topic Distance Map in BERTopic

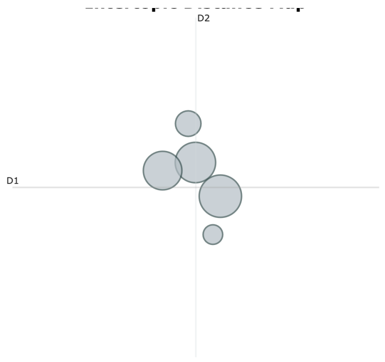
The inter-topic distance map in BERTopic is a visualization tool that shows the relationships and distances between the topics generated by the model. It helps to understand how similar or different the topics are from one another. This map is typically created using a dimensionality reduction technique like UMAP and is displayed in a 2D scatter plot.



(a) Inter-Topic Distance Map (Dataset I)



(b) Inter-Topic Distance Map (Dataset II)



(c) Inter-Topic Distance Map(Dataset III)

Figure 8.7: Inter-Topic Distance Map for Datasets I, II, and III

8.2.5 Topics Identified after training.

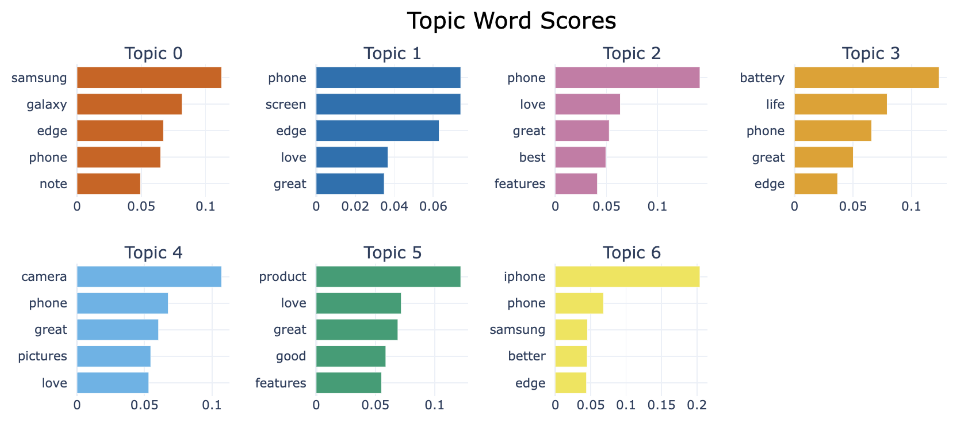


Figure 8.8: Top words per topic (dataset I)

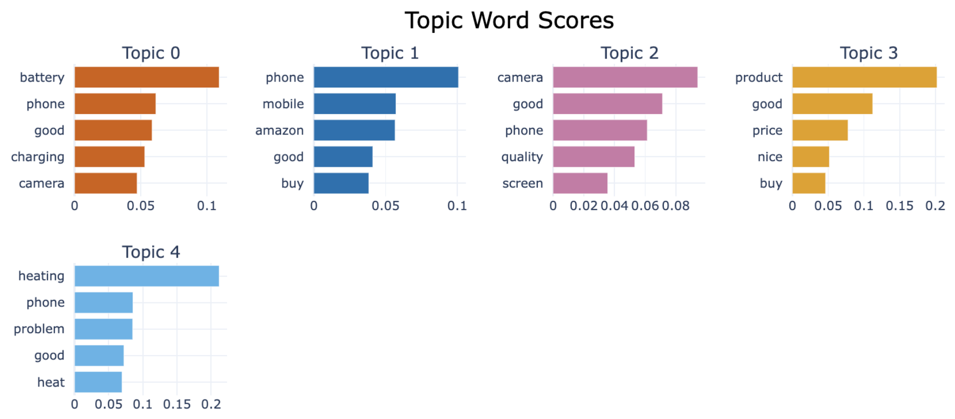


Figure 8.9: Top words per topic (dataset III)

8.2.6 Term Rank Plot

The Term Rank Plot is used to visualize the distribution of term importance within a topic based on their c-TF-IDF scores. This plot provides insights into how the relevance of terms decreases as their rank increases for a given topic.

Axes:

- **X-axis:** Term rank (ordered by importance for a given topic).
- **Y-axis:** c-TF-IDF score (importance of the term to the topic).

Interpretation:

- A **steep decline** in c-TF-IDF scores indicates that only a few terms are highly relevant, making the topic easier to interpret.
- A **shallow decline** suggests that relevance is spread across many terms, which might make the topic harder to summarize.

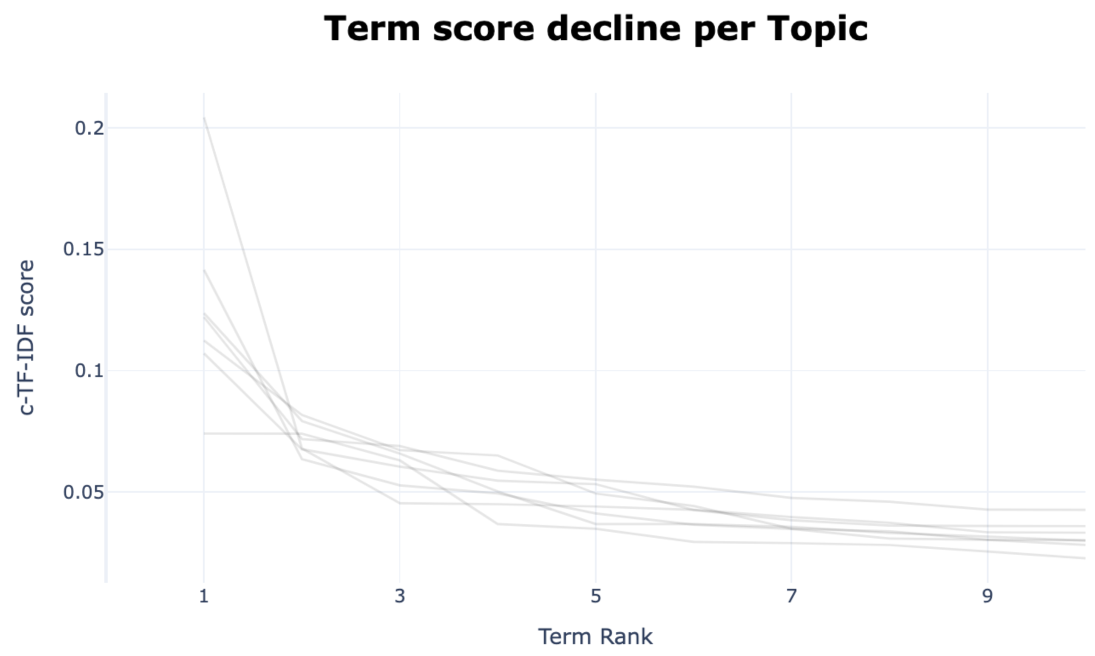


Figure 8.10: Term Rank Plot (dataset I)

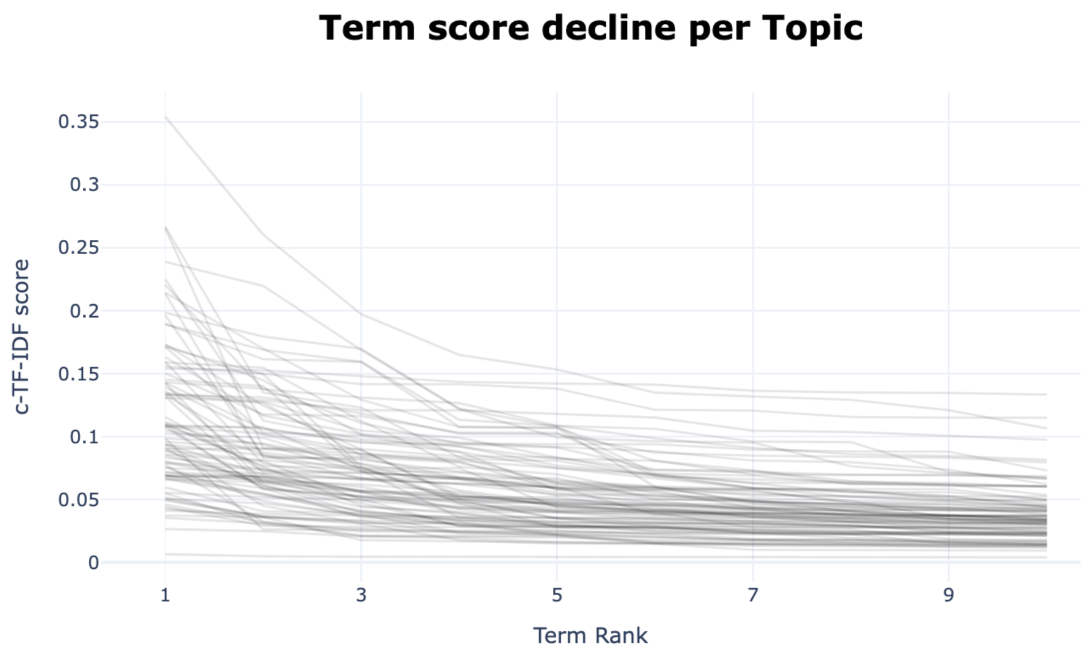


Figure 8.11: Term Rank Plot (dataset II)

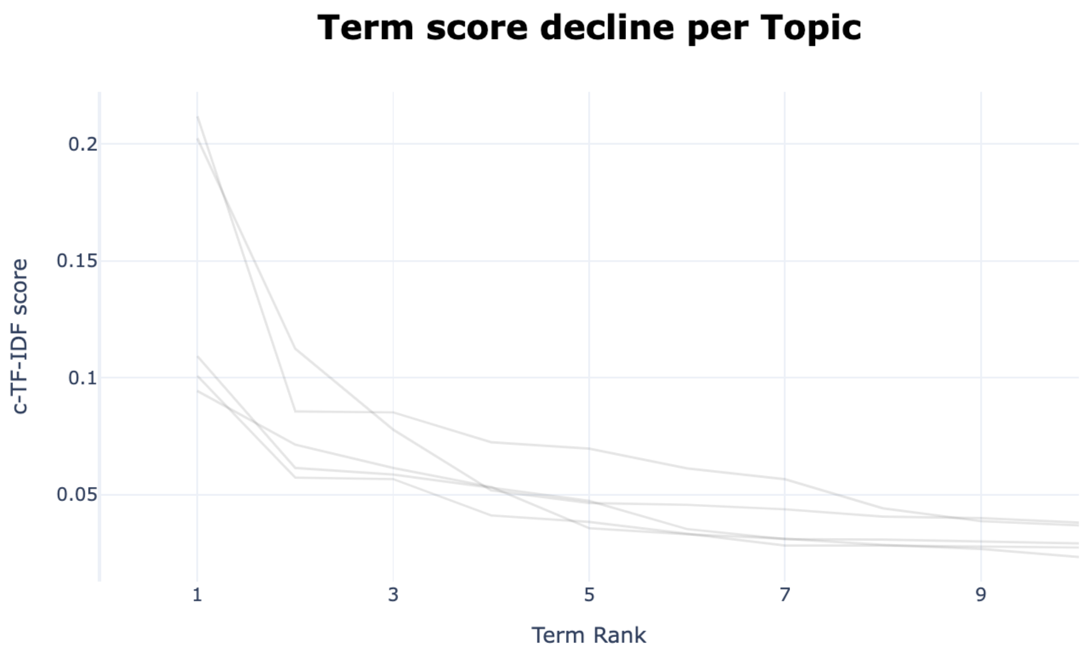


Figure 8.12: Term Rank Plot (dataset III)

8.2.7 Summary for BERTopic

The clustering and hyperparameter tuning process for BERTopic significantly influenced the performance of the model, particularly in terms of coherence score (`c_o`) and topic diversity (`t_d`). For Dataset I, the KMeans algorithm with 7 clusters achieved a silhouette score of 0.37041, which led to a coherence score of 0.4729 and a topic diversity of 0.5142. Although this result was decent, the clusters could still benefit from further tuning to improve coherence and diversity.

The performance across all datasets suggests that fine-tuning the clustering algorithm and hyperparameters, particularly for Dataset I and III, can further improve the model's coherence and topic diversity. The final results are summarized in Table 8.8, where the best combination of parameters led to improvements in both coherence and diversity, as indicated by the higher scores in Dataset II.

9. Conclusion

In this study, we evaluated the effectiveness of Latent Dirichlet Allocation (LDA) and BERTopic for topic modeling across three distinct datasets. The results highlight the importance of hyperparameter tuning in optimizing model performance, particularly in terms of coherence score and topic diversity.

For LDA, the number of passes and topics played a critical role in determining the quality of the topics generated. A higher number of passes generally improved the coherence score, but diminishing returns were observed beyond a certain threshold. Similarly, selecting an appropriate number of topics was essential for achieving a balance between granularity and coherence. After tuning these parameters, the best results were achieved with 5 topics and 15 passes for Dataset I, 6 topics and 20 passes for Dataset II, and 6 topics and 20 passes for Dataset III. However, further improvements in coherence could still be made, especially for Datasets I and II, where the coherence scores were lower compared to Dataset III.

For LSA, the number of topics (n-components) significantly influenced the coherence score (c_v) and the interpretability of the topics. Selecting an optimal number of topics ensures meaningful and coherent results. The optimal topic numbers identified for each dataset were 3 for Dataset I, 3 for Dataset II, and 7 for Dataset III, with coherence scores of 0.5158, 0.8909, and 0.4661, respectively. While the coherence score for Dataset II was notably higher, improvements in topic diversity were also observed, especially for Dataset I ($t_d = 0.7333$) and Dataset II ($t_d = 0.7333$). However, further refinement in parameter tuning for Dataset III ($t_d = 0.5714$) could enhance its performance.

In the case of BERTopic, the choice of clustering algorithm and hyperparameters significantly influenced the coherence and diversity of topics. By applying UMAP for dimensionality reduction and tuning the clustering parameters, we observed notable improvements in topic coherence and diversity, particularly for Dataset II. The use of KMeans clustering with 7 clusters for Dataset I, HDBSCAN clustering with 20 minimum cluster size for Dataset II, and KMeans clustering with 5 clusters for Dataset III produced varying results. Although the best configuration for Dataset II resulted in higher coherence and diversity, further tuning for Datasets I and III could enhance model performance.

References

[1] Akhmedov Farkhod, Akmalbek Abdusalomov, Fazliddin Makhmudov, and Young Im Cho. Lda-based topic modeling sentiment analysis using topic/document/sentence (tds) model. *Applied Sciences*, 11(23):11091, 2021.

[2] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia tools and applications*, 78:15169–15211, 2019.

[3] Muhammad Maarif. Summarizing online customer review using topic modeling and sentiment analysis. *JISKA (Jurnal Informatika Sunan Kalijaga)*, 7:177–191, 09 2022.

[4] Hye-Jin Kwon, Hyun-Jeong Ban, Jae-Kyoon Jun, and Hak-Seon Kim. Topic modeling and sentiment analysis of online review for airlines. *Information*, 12(2):78, 2021.

[5] Zheng Wang, Peng Gao, and Xuening Chu. Sentiment analysis from customer-generated online videos on product review using topic modeling and multi-attention blstm. *Advanced Engineering Informatics*, 52:101588, 2022.

[6] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[7] Asma Cheddak, Tarek Ait Baha, Youssef Es-Saady, Mohamed El Hajji, and Mohamed Baslam. Bertopic for enhanced idea management and topic generation in brainstorming sessions. *Information*, 15(6):365, 2024.

[8] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[9] D. Mimno, B. Wallach, E. Talley, M. Griffiths, and D. Steyvers. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics, 2011.

[10] D. M. Blei, L. E. Carin, and A. Y. Ng. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[11] masaladata. 14 million cell phone reviews, 2023. Accessed: 2024-11-05.

[12] Kumar Saksham. Global news dataset, 2023.