Efficient Text Clustering Techniques for Big Datasets

A Thesis

submitted in partial fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering Department

by

Vivek Mehta
(Reg no: 901603023)

Under the supervision of

Dr. Seema Bawa
Professor, Computer Science and Engineering

Dr. Jasmeet Singh
Assistant Professor, Computer Science and Engineering

Thapar Institute of Engineering and Technology
Patiala-147001, Punjab, India
November 2021

Certificate

I hereby certify that the work which is being presented in this thesis entitled "**Efficient Text Clustering Techniques for Big Datasets**", in partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** submitted to the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out during the period August 2016 to March 2021 under the supervision of **Dr. Seema Bawa**, Professor, Computer Science and Engineering and **Dr. Jasmeet Singh**, Assistant Professor, Computer Science and Engineering. I have also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

The matter presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any institution.

Date: Nov 01, 2021

Vivek Mehta
Candidate

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date: Nov 01, 2021

Dr. Seema Bawa
Supervisor

Date: Nov 01, 2021

Dr. Jasmeet Singh
Supervisor

Abstract

Clustering is regarded as one of the most important tools for data analysis, especially when label information is not available. Basically, it segregates a collection of data points into such groups that each group contains as similar data points as possible. A Big dataset in general, is characterized by several complexities including high dimensionality. Specifically, in the case of textual datasets, high dimensionality poses a great challenge for clustering as well as other text mining tasks. In a textual dataset, the number of unique words across the whole corpus (set of documents) becomes the dimensionality of the dataset. Hence, the number of dimensions can reach anywhere from tens of thousands to a few millions, for a dataset containing some thousands of documents. In addition, the matrix representation of such datasets become very sparse (containing a large number of zeros).

These major challenges make traditional clustering techniques such as partitioning-based, hierarchical, and density-based unsuitable for clustering on such high-dimensional and sparse data. In some cases, they even fail to perform clustering. Another important challenge in the case of textual datasets is to include the semantics (meaning) of text while forming clusters. In the literature, several semantic-based text clustering techniques are also defined which consider the semantics and to some extent attempts to reduce the high dimensionality problem. Still, there is a crucial requirement of text clustering techniques that can scale to the high dimensionality of large textual datasets.

In this thesis, such text clustering techniques have been proposed that attempt to simultaneously solve the aforementioned challenges. The first proposed technique is named "Stamantic Clustering" which is based on lexical chains (groups of semantically related words) and WordNet (a lexical database for English). The other proposed technique is named "WEClustering" which is based on word embeddings (a numerical vector that represents a word). Both the techniques have been validated on sufficiently large text datasets having high dimensionality. Based on various performance metrics, a comparative analysis of both techniques has also been performed with some of the existing state-of-the-art text clustering techniques. The analysis shows that the proposed techniques are more efficient in clustering high-dimensional textual datasets. Additionally, the two proposed text clustering techniques are compared among themselves for factors such as accuracy, execution, and scalability.

Keywords: Document clustering, High dimensionality, Unsupervised learning, Text datasets, Big datasets, Word embeddings, Lexical chains, Cluster analysis.

Acknowledgements

First, I would like to express my deep gratitude to my supervisors **Prof. Seema Bawa** and **Dr. Jasmeet Singh** for their invaluable advice and encouragement at every step of my PhD program. Without their unfailing support and belief in me, this thesis would not have been possible. Their contribution to this thesis goes well beyond their role as an academic supervisor and includes constant support on a personal level without which this journey may never have been completed. And for this, I am truly grateful. They are great mentor for my life as well.

I would like to express my gratitude to our Head of the department **Prof. Maninder Singh** for his constant motivation and encouragement. I also wish to thank my research committee members and non-teaching staff of the institute for their help and support. I would like to give special acknowledgement to my fellow Ph.D scholars Dr. Ajay Kumar, Dr. Simar Preet Singh, Dr. Aman Sharma, Mr. Sandeep, and Mr. Vijay for their time to time help.

I would like to express my sincere and deep gratitude to my parents and family member for their love, encouragement, care and support. They are one of the biggest supporting role players in this little tough journey.

Above all, I heartily express a deep sense of gratitude to the one who is the Supreme parent, Supreme guide and Satguru, the *Almighty God father* **SHIVA**. At each minute step, he has lighten my path by showering his unlimited and unconditional help. This moment could not have arrived without the love showered by him upon me.

Vivek Mehta

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AMI	Adjusted Mutual Information
ARI	Adjusted Rand Index
BERT	Bidirectional Encoders Representations using Transformers
BOW	Bag of Words
BOAW	Bag of All Words
BONW	Bag of nouns
BONA	Bag of nouns and adjectives
CD	Concept-Document
DC	Disambiguated concepts
DCS	Disambiguated core semantics
ES	Extensive Similarity
GloVe	Global Vectors
HDBSCAN	Hierarchical Density Based Spatial Clustering of Applications
KLD	Kullback-Leibler Divergence
LCS	Least Common Subsumer
NMI	Normalized Mutual Information
OOV	Out of Vocabulary
SMTP	Similarity Measure for Text Processing
STC	Stamantic Clustering
TF-IDF	Term Frequency-Inverse Document Frequency
VSM	Vector Space Model
WSD	Word sense disambiguation
WEClustering	Word Embeddings Based Clustering

Chapter 1

Introduction

Clustering has emerged as a highly active area of research in the field of data mining and machine learning. The numerous applications of clustering in various domains such as business intelligence, security, medicine, information retrieval, and biology make clustering a very useful tool for data analysis. Alternatively, it also serves as an intermediate step for other data analysis techniques such as classification and outlier analysis. A basic definition of clustering can be given as follows: [7]
"Given a collection of data objects, organize them into different groups such that objects in each group are as similar as possible."

This chapter presents the basic concepts involved in the complete process of clustering a dataset. Section 1.1 provides an introduction to clustering. Section 1.2 defines various proximity measures which are an important component in many clustering techniques. Section 1.3 presents different performance metrics used to measure the accuracy of clustering. Section 1.4 presents various application domains of clustering. Section 1.6 gives an overview of the contributions of this thesis. Lastly, section 1.5 presents the organization of this thesis.

1.1 Clustering

Clustering is the process of segregating a collection of data objects (known as a dataset) into different groups such that the objects in one group are more similar to each other in comparison to the objects of different groups. Hence the underlying principle behind this process is to *maximize the intraclass similarity and minimize the interclass similarity* [8]. Each group or subset resulting out of this process is called a cluster.

The definition of a data object can be different for different applications. For example, a data object can be a patient in a medical dataset, a document in a text dataset (corpus), a house in a real estate dataset, an image in an image dataset, a student in a university dataset, and so on. Other terms that are generally used for data objects are data points, examples, and instances. To process a dataset, each data object is generally represented

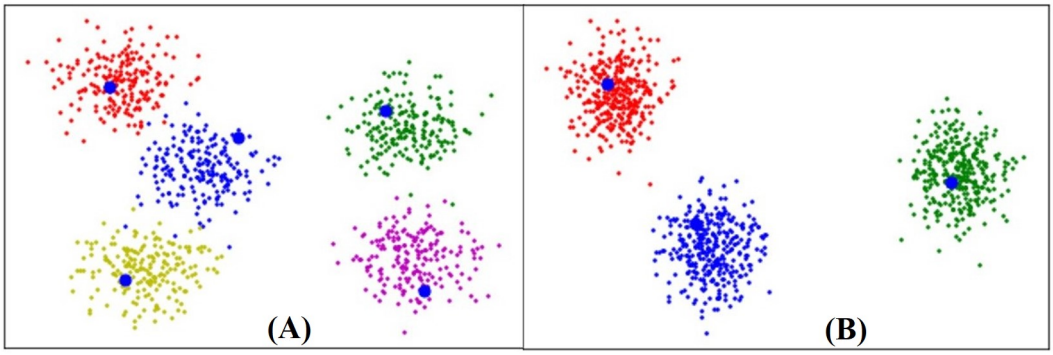

Figure 1.1: Clustering performed on two different datasets [1] each having 1000 data points.

by a set of attributes. For example, a house can be represented by attributes such as its area, the number of bedrooms, year of built, price/sq. feet of land and so on. Terms such as *features, dimensions, attributes, and variable* are used interchangeably in the literature. A simple illustration of clustering performed on two synthetic datasets [1] each containing 1000 data points is shown in Figure 1.1. As shown, 5 clusters are found in the dataset labelled as (A) and 3 clusters are found in the dataset labelled as (B).

There exist many techniques in the literature to perform clustering on a given dataset. Broadly, these techniques can be categorized into five categories [9, 10] namely: Partitioning-based, Hierarchical, Density-based, Grid-based, and Model-based. The definition and details of these categories are provided in chapter 2. In general, a good clustering technique should have the following capabilities [8].

1. **Ability to deal with complex data types**: A large number of clustering techniques are designed for numerical data types. With the advancement of the Internet, datasets of other complex types such as documents, images, and graphs are also being generated on a large scale. Hence, clustering techniques specific to these data types are required.

2. **Ability to handle high-dimensional data**: Traditional clustering techniques are more suitable for clustering datasets with a low number (less than 10) of dimensions. However, in a text dataset, for example, each unique word is considered as a dimension, hence, the number of dimensions may reach several thousand or even more, and that may be sparse too. Clustering techniques suitable to this kind of scenario are required.

3. **Insensitivity to input order**: Some clustering techniques are sensitive to the input order of data objects given to them. Clustering algorithms insensitive to the input order are required.

4. **Fewer input parameters**: Many clustering algorithms require such input parameters that can be hard to determine. For example, the number of clusters can be difficult for users to determine in the initial stage. Hence, clustering techniques that require less and easy to supply parameters are desirable.

5. **Scalability**: Several clustering techniques are suitable for small-sized datasets only, such as those with only a few hundred data objects. With time, datasets have grown very large which may contain up to millions of data objects. Hence, clustering techniques scalable to the large size of datasets are required.

1.2 Proximity Measures

A dataset is a collection of data objects and a data object represents an entity that can be, for example, a patient in a medical database, a document in a document dataset, a student in a university database, and an image in an image dataset. Each data object is represented by some attributes that are also known as features. In data analysis tasks such as clustering, classification, and outlier analysis, the first and very crucial step is to calculate the proximity (similarity/dissimilarity) between two data objects. If important factors such as sparsity in data, the correlation among data features, and feature format are ignored in this step, meaningful patterns may remain obscure [11]. The first and obvious distinction to be made for selecting a proximity measure is the category (or type) to which an attribute belongs. These four basic categories are nominal, binary, ordinal, and numeric [8] as shown in Fig. 1.2. Nominal attributes contain names that represent a category; hence, these are also known as categorical attributes. There is no specific order or sequence to be followed for these names. In the case where nominal attributes contain only two values, attributes are known as binary attributes. Ordinal attributes are the same as nominal but a specific order exists among the labels/names. Lastly, numeric attributes consist of numerical values.

This research work specifically focuses on proximity measures that deal with the numeric type of data attributes. These are discussed in the next subsection.

1.2.1 Proximity measures for numeric data

Various widely used numeric proximity measures are described below.

1. Euclidean distance: Let $p_i = (p_{i1}, p_{i2}, ..., p_{in})$ and $p_j = (p_{j1}, p_{j2}, ..., p_{jn})$ be two data

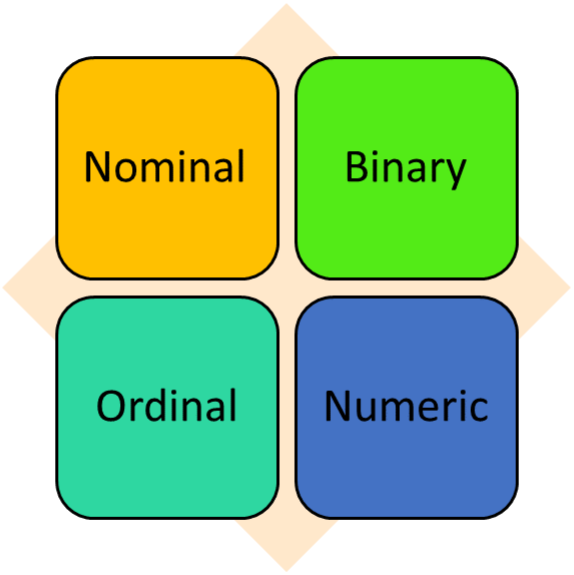

Figure 1.2: Types of attributes

points in some dataset X. The Euclidean distance between p_i and p_j is given as

$$d^2(p_i, p_j) = (p_{i1} - p_{j1})^2 + (p_{i2} - p_{j2})^2 + ... + (p_{in} - p_{jn})^2 \tag{1.1}$$

2. Manhattan distance (or city block distance): This distance is defined as

$$d(p_i, p_j) = |p_{i2} - p_{j2}| + |p_{i2} - p_{j2}| + ... + |p_{in} - p_{jn}| \tag{1.2}$$

That is, it walks along only one of the axis at a time. This is analogous to walking in a city of blocks where one cannot go diagonally between two locations; instead, we have to walk along with either of the two dimensions at a time.

3. Minkowski distance: This distance measure is given by the function:

$$d(p_i, p_j) = \sum_{k=1}^{n} (|p_{i,k} - p_{j,k}|)^x)^{1/x} \tag{1.3}$$

Note that, for $x = 1$, it becomes Manhattan distance and for $x = 2$, it becomes Euclidean distance. Hence, it is a general form of both of them.

4. Chebyshev distance: For two vectors, this metric is defined as the maximum of the difference across any of the dimensions of vectors; hence it is also known as the maximum metric. Thus, for two data points $p_i = (p_{i1}, p_{i2}, ..., p_{in})$ and $p_j = (p_{j1}, p_{j2}, ..., p_{jn})$, Chebyshev distance is calculated as

$$d(p_i, p_j) = \max_n(|p_{in} - p_{jn}|) \tag{1.4}$$

5. Cosine similarity: For sparse datasets (those in which a significant number of zeros are present), the aforementioned traditional measures often do not work well. For example, in document clustering, the representation of a document often consists of a large number of zeros, making the dataset sparse. In such cases, cosine similarity is often used for measuring the similarity between two documents [8]. It is calculated as the cosine value of the angle between vectors that represent two documents.

$$sim(d_a, d_b) = \frac{d_a.d_b}{||d_a||.||d_b||} \tag{1.5}$$

   Here, $d_a.d_b$ is the dot product between the vectors d_a and d_b. $||d_a||$ and $||d_b||$ denote the length of the vectors d_a and d_b respectively.

6. Pearson distance: Based on the Pearson correlation, the Pearson distance is defined as

$$1 - Corr(a, b) \tag{1.6}$$

   Here, $Corr(a, b)$ is the Pearson correlation of two variables a and b. It is defined as

$$Corr(a, b) = \frac{Cov(a.b)}{\sigma_a.\sigma_b} \tag{1.7}$$

   where $Cov(p, q)$ is the covariance between a and b. σ_a and σ_b are the standard deviations of a and b respectively.
   The Pearson distance lies in [0, 2]. This measure has been shown as sensitive to outliers [12] by Anscombe who highlights the importance of studying graphs.

7. Kullback-Leibler Divergence (KLD): If a dataset is assumed to be following some

probability distribution, then a measure proposed by Kullback and Leibler [13] calculates the distance between two probability distributions P_a and P_b as

$$D_{KL}(P_a||P_b) = \sum_i P_a(i) \log \frac{P_a(i)}{P_b(i)} \tag{1.8}$$

This measure lies in the category of non-metric because it is not the symmetric measure and secondly it does not satisfy the triangle inequality [14]. Huang [15] compared the effectiveness of the average KLD measure with other measures such as Euclidean, Cosine, Jaccard, and Pearson in the domain of text document clustering.

8. Canberra distance metric: This distance metric is used when data vectors contain all non-negative elements [16]. For the n-dimensional vectors p and q, it is formulated as

$$d(p_i, q_i) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i + q_i|} \tag{1.9}$$

9. Bray-Curtis: This dissimilarity measure is specifically used in the field of ecology and biology to calculate the difference between the counts of species existing on two different sites. It is formulated as

$$BC_{ab} = 1 - \frac{2C_{ab}}{S_a + S_b} \tag{1.10}$$

where S_a and S_b are the counts at two sites a and b, and C_{ab} is the sum total of smaller counts for each species on both the sites [17].

10. Jaccard similarity coefficient: Between two finite sample sets, it measures the similarity as the ratio of the intersection and the union of two sets (say S_1 and S_2) [18].

$$J(S_1, S_2) = \frac{|S_1 \bigcap S_2|}{|S_1 \bigcup S_2|} \tag{1.11}$$

In generalized form, the Jaccard index between two vectors $P = (p_1, p_2, p_3, ..., p_n)$ and $Q = (q_1, q_2, q_3, ..., q_n)$ is calculated as

$$J(P, Q) = \frac{\sum_i \min(p_i, q_i)}{\sum_i \max(p_j, q_j)} \tag{1.12}$$

Table 1.1: Capability of different proximity measures.

Proximity measures	Capability factors			
	Metric	Affected by variable's scale	Considers sparsity	Considers correlation
Euclidean Distance	Yes	Yes	No	No
Manhattan Distance	Yes	Yes	No	No
Minkowski Distance	No	No	No	No
Chebyshev Distance	Yes	Yes	No	No
Cosine similarity	No	No	Yes	No
Pearson Correlation	No	No	No	Yes
KLD	No	Yes	Yes	No
Canberra distance metric	Yes	No	No	No
Bray-Curtis	No	No*	No	No
Jaccard Coefficient	Yes	Yes	No	No
Dice coefficient	No	No	Yes	No
Mahalanobis distance:	Yes	No	No	Yes

11. Dice coefficient: For two vectors x and y, the dice coefficient [19] is given as

$$S_{Dic}(x, y) = \frac{2x.y}{x.x + y.y} \tag{1.13}$$

12. Mahalanobis distance: For two vectors u and v, Mahalanobis distance [20] is given by

$$d_{mah} = \sqrt{(u - v)S^{-1}(u - v)^T} \tag{1.14}$$

Here, S is the covariance matrix of the dataset.

All the aforementioned proximity measures have been mathematically analyzed in this paper for some preprocessing aspects. These are the effects of scales of variables, the sparse structure of data, existing correlation between the attributes, and metric/non-metric. This analysis is presented in Table 1.1.

*Only if the scales for all the variables are same.

1.2.2 Domain specific proximity measures

Apart from proximity measures mentioned in the previous subsection, it is worth mentioning some proximity measures that are relatively recent and have been designed in pertinent to a specific domain such as text mining. These are described below.

1. Extensive Similarity (ES): This similarity measure [21], extensively takes each and every document d_k present in the corpus to determine the similarity between the two documents d_i and d_j. According to ES, two documents are said to be exactly similar to each other if they both are similar to each other and they both are either similar or dissimilar to every other document contained in the corpus.

   The first step of ES is to calculate the value of $dis(di, dj)$ as follows.

$$dis(d_i, d_j) = \begin{cases} 1, & \text{if } \rho(d_i, d_j) \leq \Theta \\ 0, & \text{otherwise.} \end{cases} \tag{1.15}$$

   where $\rho(d_i, d_j)$ is a similarity measure (cosine has been used in the original work) and $\theta \in (0, 1)$. If $dis(d_i, d_j) = 0$, a score l is assigned as follows.

$$l_{i,j} = \sum_{k=1}^{N} |dis(d_i, d_k) - dis(d_j, d_k)| \tag{1.16}$$

   Here, N denotes the total documents in the corpus, and finally, ES for two documents d_i, d_j is

$$ES(d_i, d_j) = \begin{cases} N - l_{i,j}, & \text{if } dis(d_i, d_j) = 0 \\ -1, & \text{otherwise.} \end{cases} \tag{1.17}$$

   Thus, two documents d_i and d_j can have a maximum ES value of N when the distance between them is zero and for every k, the distance between d_i and d_k and the distance between d_j and d_k is the same.

2. Similarity Measure for Text Processing (SMTP): This measure considers the absence and presence of a feature in two documents to be more significant than the difference of feature values. For example, if a feature w_1 is absent in d_1 but present in d_2 so that $d_{11} = 0$ and $d_{21} = 2$, and another feature w_2 is present in both d_1 and d_2 so that $d_{12} = 3$ and $d_{22} = 5$, then w_1 is considered to be more important than w_2 in calculating the similarity between the documents d_1 and d_2 despite of the

same difference value which is 2. This property was shown to remain unsatisfied by other traditional proximity measures such as Euclidean, Cosine, Dice coefficient and IT-Sim, etc in a previous study [19]. Additionally, the study indicated that the usefulness of a similarity measure strongly depends on the following factors:

(a) Applications domains (e.g., image or text).

(b) Representation format of the feature, for example, Term Frequency Inverse Document Frequency (Tf-IDF) or word count in case of a text document.

(c) The classification/clustering algorithms used.

These results form the basis to propose a theoretical procedure in Section 4.

3. DRSim: Between the two document vectors x_i and x_j, DRSim [22] finds the similarity as

$$DRSIM(x_i, x_j) = \left(\frac{\sum_{k=1}^{m} |x_{ik} - x_{ij}|^2}{m} \right)^{1/m} \tag{1.18}$$

where m denotes the dimension of the feature vectors of documents. The results were shown to be better than those achieved using the Minkowski distance measure [22].

4. Style based: In a previous study [23], a novel similarity metric was proposed that considers the position of concepts(terms) in a document. The idea behind this was that two similar documents should share some structural arrangement of terms contained in them. Thus, the final metric of this study contained two aspects, one for concepts of terms derived using EuroWordNet[24] ontology and the other as the position of concepts.

5. Kernel induced: To capture patterns contained in the form of non-spherical clusters, Kannan et al. [25] used kernel functions instead of Euclidean distance. Kernel functions map the original feature space to a higher dimension space by using some non-linear transformation (such as Gaussian kernel, sigmoid kernel and polynomial kernel). The new distance function obtained out of this is as follows:

$$d^2(x_k - v_i) = 2(1 - K(x_k, v_i)) \tag{1.19}$$

Where,

$$K(x, y) = \exp\left(\frac{-||x - ||^2}{2\sigma^2} \right) \tag{1.20}$$

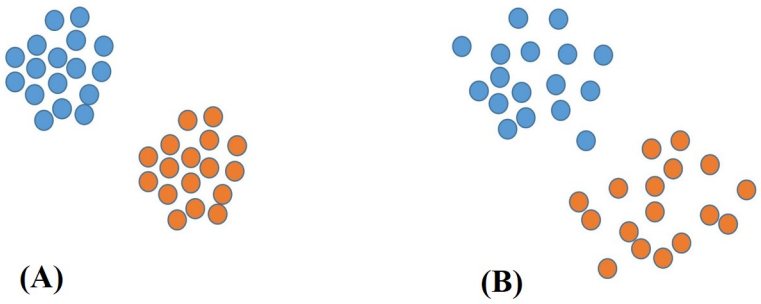

Figure 1.3: Comparison of compactness of clusters (Clusters with label (A) are more compact than the clusters with label (B).

1.3 Performance Metrics

To evaluate the quality of clusters formed using any clustering technique, several metrics have been defined in the literature. These metrics evaluate clustering quality based on several factors such as compactness of a cluster, inter-cluster distances (separation between clusters), and the number of data points correctly included in a cluster (as matched from given ground truth labels), etc. Based on these factors, the performance metrics can be classified into two broad categories, external metrics, and internal metrics. External metrics measure clustering performance with the help of externally provided ground truth labels for each data point. Examples of this kind are Normalized Mutual Information, F-measure and Adjusted Rand Index, etc. Internal metrics such as Silhouette coefficient, Calinski-Harabasz index, and Dunn index evaluate clustering quality based on the internal structure of a cluster such as compactness (Figure 1.3) and separation (Figure 1.4). The pros and cons of various clustering metrics have been identified in several research studies [26, 27, 28]. Clustering metrics that are widely used and most relevant to this research work are defined below.

1.3.1 Normalized Mutual Information (NMI)

As per its name, NMI is a normalized version of Mutual Information (MI) [26] to scale it in the range [0,1]. It is given by the following equation.

$$\text{NMI}(C1, C2) = \frac{\text{MI}(C1, C2)}{\text{mean}(H(C1), H(C2))} \tag{1.21}$$

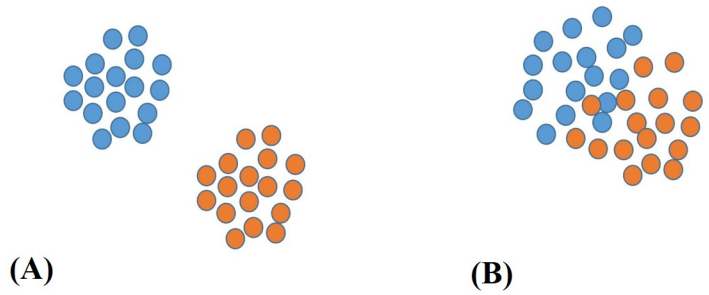

Figure 1.4: Comparison of separation of clusters (Clusters with label (A) are more separated than the clusters with label (B).

Where MI is given by

$$MI(C1, C2) = \sum_{i=1}^{|C1|} \sum_{j=1}^{|C2|} \frac{|C1_i \cap C2_j|}{N} \log \frac{N|C1_i \cap C2_j|}{|C1_i||C2_j|} \tag{1.22}$$

$H(C)$ is the entropy associated with a clustering C and is given by the following equation.

$$H(C) = -\sum_{i=1}^{|C|} P(i) \log(P(i)) \tag{1.23}$$

where $P(i) = |C_i|/N$ is the probability of an object taken at random from C falls into class C_i.

1.3.2 Adjusted Mutual Information (AMI)

It is an adjusted version of MI for the impact of number of clusters and number of samples on metrics of clustering performance evaluation. For two clusterings $C1$ and $C2$, AMI [26] is given as:

$$AMI(C1, C2) = \frac{[MI(C1, C2) - E(MI(C1, C2))]}{[avg(H(C1), H(C2)) - E(MI(C1, C2))]} \tag{1.24}$$

$E(MI(C1, C2))$ is the expected Mutual information between two clusterings $C1$ $C2$ (for more details refer [26]). MI and $H(C)$ are given by the same equations as for NMI.

Value of AMI ranges in [0, 1] where a large value implies better clustering.

1.3.3 F-measure

It is defined as the harmonic mean of precision and recall [29]. Hence, as per its definition more is the balance between precision and recall, better is the F-measure. Mathematically,

$$Precision = \frac{TP}{TP + FP} \tag{1.25}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.26}$$

where,

True Positives (TP) is the total number of correctly labeled positive instances whereas True Negatives (TN) are the correctly labeled negative instances. Similarly, False Positives (FP) is the total number of incorrectly labeled positive instances and False Negatives (FN) is the total number of incorrectly labeled negative instances. Based on these definitions, F-measure is given as:

$$F_{measure} = 2 \times \frac{precision \times recall}{precision + recall} \tag{1.27}$$

For a multiclass dataset (which is our case), the average value of the F-measure for each class is computed as the final score. This ranges from $[0, 1.0]$ where the score of 1.0 implies perfect clustering and 0 implies bad clustering. F-measure is a reliable measure even if the data is not evenly distributed.

1.3.4 Adjusted Rand Index (ARI)

Adjusted Rand Index (ARI): ARI is a widely used metric for assessing cluster quality in the case of availability of true labels [30]. It can be used to measure two different clustering assignments that ignore different permutations of the same clustering. Two similar clusterings achieve a score near +1.0 and completely different clusterings achieve a score approaching -1.0.

1.3.5 Purity

Given the cluster assignments and the actual class labels, Purity is calculated by first counting the number of documents (our case) in a cluster of the class which is the most frequent in this particular cluster [31]. This value is summed over all the clusters and is divided by the total number of documents. Formally, it is given as:

$$Purity = \frac{1}{N} \sum_{c \in C} \arg\max_{d \in D} |c \cap d| \tag{1.28}$$

Where C is the set of clusters, D is the set of actual classes and N is the total number of documents. It ranges in $[0, 1.0]$ where a better clustering has a greater value.

1.3.6 Silhouette Coefficient

Silhouette coefficient [32] measures the quality of clustering when truth labels are not available which is a more realistic case of clustering. It is a widely used metric that measures how dense and well separated the clusters are. It is given by:

$$s = \frac{b - a}{\max(b - a)} \tag{1.29}$$

In this equation, a is the average distance between a sample and all other points in the cluster. b is the average distance between a sample and all other points of the next nearest cluster. The range of the Silhouette coefficient is [-1,1], where a higher value indicates better clustering.

1.3.7 Calinski-Harabasz (CH) index

CH index is defined as the ratio of between clusters dispersion and within clusters, dispersion [33]. A higher CH score indicates a better clustering which is generally in the case of dense and well-separated clusters. However, it is more biased towards convex-shaped clusters than other shapes of clusters.

1.3.8 Dimensionality

Dimensionality is the number of features used to find clusters of documents. Less features lead to less execution time. However, accuracy should not get reduced using less number of features. Hence, a trade-off between execution time and accuracy of clustering is required based on the number of features.

1.3.9 Execution Time

The Time taken for finding the groups of similar kinds of texts/documents in a corpus is called the execution time of a clustering algorithm.

1.4 Application domains of Clustering

Various application domains of clustering are listed in the following subsections.

1.4.1 Medical data analysis

In medical domain, clustering is used for applications [34] such as predicting the likelihood of diseases [35, 36, 37], image segmentation of medical images [38, 39], resource decision making [40] and analysis of gene expressions data [41, 42].

1.4.2 Market data analysis

In the stock market domain, clustering is often useful to extract useful patterns using which profitable opportunities of buying and selling shares can be identified [43]. Secondly, clustering can be used to identify fraudulent insurance claims such as those related to health insurance [44] and vehicle insurance. Customer segmentation is another application of clustering in the market domain which helps to identify consumer structure especially for an emerging market [45] (such as electric vehicles).

1.4.3 Search engines

A search engine is used to retrieve information from the web corresponding to a user query. Clustering is used to organize the search results to find coherence groups according to

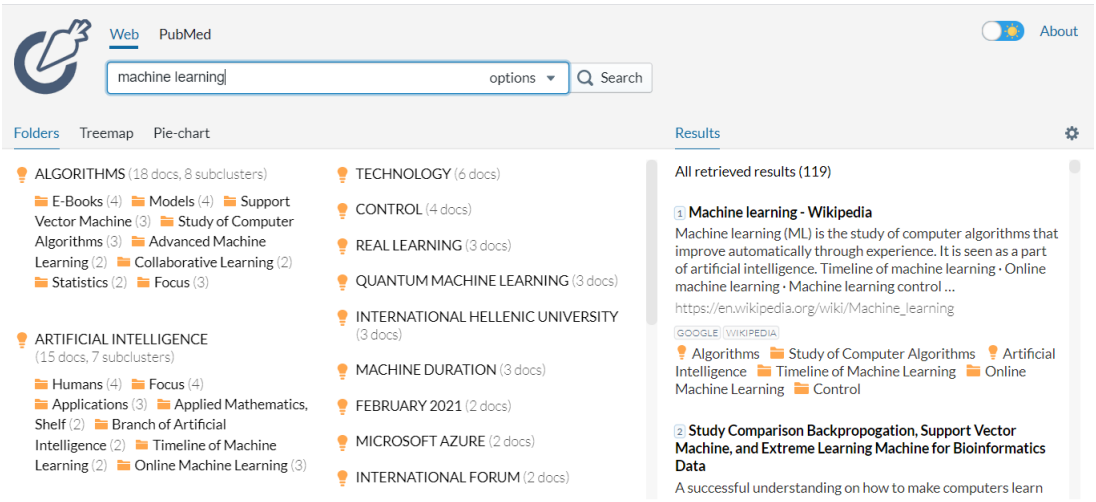

Figure 1.5: Search results clustering for web pages from carrotsearch.com [2].

subtopics. This is known as search results clustering (SRC) [46]. Specifically for images, the process is known as Image Search Results Clustering (ISRC) and for the case of web pages, this process is known as Page Search Results Clustering (PSRC) [47]. Examples of such type of search engines include Carrotsearch [2] and Yippy [3]. Figure 1.5 and Figure 1.6 shows clustering results for web pages and images respectively for the query "machine learning".

1.4.4 Text data analysis

With the advancement of the Internet and other related technologies along with its increasing availability, a massive amount of textual data is being generated in the form of electronic documents. According to the latest report published by the International Association of Scientific, Technical and Medical Publishers (STM) [48], around 33,100 scholarly journals of English language alone exist. This gives rise to the generation of more than 3 million articles per year. Additionally, the rate of growth of research articles is around 4% per year owing to the continued growth of investment in research and development and the increasing number of researchers. This in turn brings to us the challenges of harnessing valuable insights from such massive amounts of textual data. Clustering of data is the most fundamental technique that is used to group similar items in a cluster (or group). Some important applications [49] of clustering in text data analysis are discussed below.

1. Automatic document organization and browsing: Organizing a large collection of documents into coherent categories helps in systematic browsing. *Carrot* is a popular software used for document clustering and visualization [50].

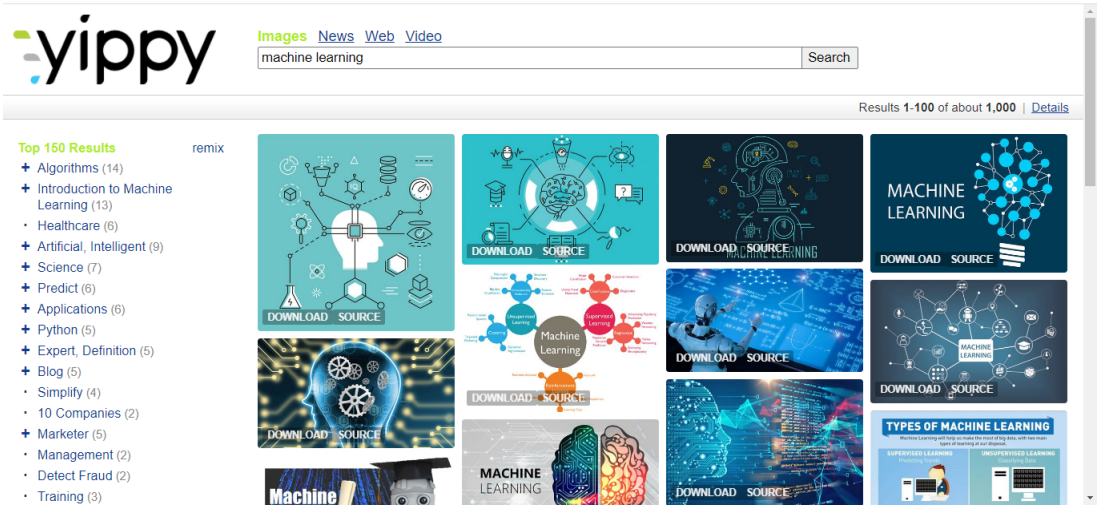

Figure 1.6: Search results clustering for images from carrotsearch.com [3].

2. Automatic text summarization: The task of producing a concise summary for a collection of documents for fast selection is known as text summarization. According to Radef et al. [51] "a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that". Clustering of documents is often used as an intermediate step to produce a summary for a corpus consisting of a large number of documents [52]. News websites and search engines use text summarization to provide a better browsing experience [53, 54].

3. Document classification: Classification of documents is a process in which a document is assigned to one (or more) pre-defined categories based on its context. Although clustering cannot assign a pre-defined category to a document, however, generation of clusters can be used to improve the process of document classification [55, 56, 57].

4. Social news clustering: In recent years, the emergence of social media as a huge platform for information sharing has led news organizations to extract useful information from it. Some recent attempts for the news clustering task involve modeling of pair of words that co-occur in a corpus of short texts [58, 59]. In [60] a cluster-then-label semi-supervised approach for labeling of tweets as spam/not spam was proposed. Grouping of news written in different languages is another interesting area of research; for instance, in [61], the Chinese-Vietnamese news dataset was used for clustering based on the semantic correlation between two languages.

5. Sentiment analysis: Various e-commerce websites such as Amazon and Flipkart offer their users to express their reviews regarding their products and services. Similarly,

social networking sites such as Facebook, YouTube, and Twitter, allow users to express their opinions regarding any social event or political news. "Sentiment analysis is the task of detecting, extracting and classifying opinions, sentiments and attitudes concerning different topics, as expressed in textual input" [62]. This type of analysis helps other users on the same platform to make purchase decisions and service providers to improve the quality of services. Clustering is an important step in the whole process of sentiment analysis for grouping similar sentiments together. Ravi et.al [63] presented a comprehensive survey of different techniques used in sentiment analysis. Recent research in this area has focused on i) a language-independent approach for sentiment analysis such as the one presented in [64], and ii) a novel vector space model for concept-level sentiment analysis that allows reasoning by analogy on natural language concepts namely *AffectiveSpace2* [65, 66].

As a part of pre-processing, following steps are generally performed in a text data analysis task to be somewhat more efficient.

1. Punctuation removal: To perform discrimination among text documents, punctuation marks such as a full stop (.), comma (,), colon (:), semi-colon(;), question mark (?) and quotation marks (" ") are not important. Hence, these are removed to make a process more accurate and computationally lighter.

2. Stop-words removal: A large number of words in any language are used to make the language grammatically correct but do not contribute to the actual theme of a text. These words include "the", "a", "and", "in", "of", "are", "that", "by", "for" and so on. These must be removed to compare any documents for their semantic aspect.

3. Tokenization: It is the process of converting a document (consisting of a large number of sentences concatenated together) into a list of words (or sentences). This is generally done for extracting features from any dataset on the basis of which any task such as categorization or classification can be performed.

4. Lowercasing: As the name suggests, it is the process of lowercasing each word in a document. This is generally done in order to reduce the size of corpus vocabulary.

5. Stemming: It is a process in which each word of a document is converted into its root word. For example, the three words *consist, consists, consisting* can be reduced only to the single word *consist*.

1.5 Thesis Organization

This thesis is organized as follows.

Chapter 1 initially gives an introduction about clustering and the characteristics of a good clustering method. Secondly, it describes various proximity measures used in clustering, followed by different performance metrics used to measure the accuracy of a given clustering. Then it lists various application domains of clustering. Finally, the chapter provides thesis organization and highlights thesis contributions.

Chapter 2 firstly presents a detailed review of different techniques used for representing text in a form suitable to clustering. Secondly, the chapter provides an extensive literature review of a wide variety of text clustering techniques. Based on this literature review, identified research gaps, problem formulation, and research objectives are also presented.

Chapter 3 describes the first proposed clustering technique named "Stamantic Clustering" (STC) for text datasets. It also provides the implementation details of STC such as datasets used, parameter settings, performance metrics, and compared techniques.

Chapter 4 describes the second proposed clustering technique named "WEClustering" for text datasets. Following this, it provides the details of implementation such as datasets used, parameter settings, performance metrics, and techniques compared.

Chapter 5 covers all the experimental results and analysis of the proposed text clustering techniques. It also provides an experimental comparison of the performance of the proposed text clustering techniques for a variety of datasets. This comparison is presented based on factors such as accuracy, execution time, and dimensionality reduction, etc.

Chapter 6 provides the concluding remarks on the thesis and in the end, the future scope of the work has been discussed.

1.6 Thesis Contributions

This thesis makes the following research contributions.

1. An extensive literature review of different techniques for text representation and text clustering is presented. Also, various research gaps are highlighted for devising a new text clustering technique with high scalability for a large number of

dimensions and a high volume of text data.

2. A novel text clustering technique named "Stamantic Clustering" (STC) is proposed and implemented on a variety of text datasets. This technique combines the statistical and semantic features of the text to reduce the effect of high dimensionality to produce better clustering results.

3. A comparative analysis of STC is performed with several widely used techniques for text clustering. The analysis shows more efficiency of STC in terms of accuracy, execution time, and dimensionality reduction, in comparison to the other text clustering techniques.

4. Another novel text clustering technique based on modern deep learning architecture is proposed and implemented on different text datasets. This technique named "WEClustering" captures the semantics of words in the form of numerical vectors known as word embeddings. These embeddings are useful for comparing different words and combining them into groups to achieve a significant amount of dimensionality reduction for a large volume of datasets.

5. A comparative analysis of WEClustering with other widely used and state of the art clustering techniques is performed. The analysis shows the efficiency of WEClustering in terms of higher accuracy, better dimensionality reduction, and thus more suitability for big datasets.

6. STC and WEClustering are compared among themselves for their performances on different text datasets. The results are analyzed for accuracy, execution time, scalability to high dimensional and high volume datasets, and language dependency. Lastly, concluding remarks and suggestions for future research are provided.

In the next chapter, an extensive literature review of text representation and clustering techniques is presented.

Chapter 2

Literature Review

In general, data clustering has been studied for the past several decades. Many review studies have been published from time to time which surveyed the existing clustering techniques. This chapter aims to present an exhaustive review of clustering techniques, especially in view of textual data. The advantages and disadvantages of different types of existing clustering techniques are also highlighted. A comparative analysis of a number of text-specific clustering techniques is also presented. This chapter is organized as follows. Section 2.1 presents different ways of representing a text dataset in a suitable form for clustering are presented. In section 2.2, an extensive review of various text clustering techniques which cover the broad categories of clustering as well as text-specific clustering techniques is presented. Based on the presented literature review, this chapter provides the identified research gaps, problem formulation and the research objectives in sections 2.3, 2.4 and 2.3 respectively.

2.1 Representation of Text for Clustering

The ways of representation of a set of documents for clustering are given in the following subsections.

2.1.1 Vector Space Model

To perform the clustering of documents contained in a corpus, all the documents are converted or represented into some numerical form. Generally, each document is converted into a numerical vector in which each numerical value reflects the importance of a word in deciding the category of document. It is also called a *document vector*. This representation is also known as *Bag of Words* [31] model for a document. Based upon this representation, each of the documents contained in the corpus is mapped to a common space of vectors. In a vector corresponding to a document, each vector component corresponds to a word contained in *vocabulary* of the whole corpus. This set is generally formed by eliminating words of less importance such as *stop words* and other items as

	d_1	d_2
mouse	1	0
quicker	1	1
cat	1	1
dog	0	1

Figure 2.1: An example of a term-document matrix.

punctuation and digits etc. This representation of a set of documents is known as *Vector Space Model*. The output of this representation for a collection of N documents is an M x N matrix called as *term-document matrix* having M words (also called as dimensions) in vocabulary.

For example, let us assume that a corpus contains two documents *Mouse is quicker than cat*, say d_1 and *Cat is quicker than dog*, say d_2. Vocabulary formed from this set of two documents is the set {mouse, quicker, cat, dog} after eliminating stop words. A 4x2 term-document matrix resulting out of this is shown in Figure 2.1. The numerical values indexed at (i, j) are frequencies of i^{th} word in j^{th} document.

2.1.2 Term Scoring

In a document classification or clustering task, computing similarity between two documents is often done. As aforementioned, a document is represented as a vector that quantifies the importance of a word in that document. This quantification of the relative importance of each word can also be called *term scoring*. In a first attempt, term scoring can be done using the frequency of each term t occurring in a document d denoted by $TF(t, d)$. As two different words contribute differently for deciding the actual theme of a document, the term frequency approach suffers from the limitation of providing equal weights to every term. Hence, a mechanism is required that distinguishes the relative importance of each term in a document. For example, stop words occur too often in a document but contribute very little in deciding its similarity to another document.

For a term (or word) t, to give it a higher weight when it occurs in few documents and lower weight when it is frequent among all the documents, a scaling factor known as *inverse document frequency* [67] denoted by $IDF(t)$ is defined below.

$$IDF(t) = \log \frac{N}{df} \tag{2.1}$$

where N is the total number of documents in a corpus and df is the number of documents in which the term t occurs.

Now a weighting scheme known as $TF-IDF$ is defined by combining the term frequency and inverse document frequency that assigns a weight to a term t in a document d. It is given as follows.

$$TF - IDF = TF(t, d) \times IDF(t) \tag{2.2}$$

Thus $TF-IDF$ scoring mechanism gives the highest score to a term that is rare among documents but frequent within a document. Similarly, it assigns the lowest score to a term that is much frequent in a large number of documents in a collection of N documents. Other modifications in this scheme like $tf.rf$ where rf stands for $relevancefrequency$ are also defined in literature. [68, 69] presents a comparison of these kinds of schemes.

After a document is represented as a vector using the Bag of words model and $TF-IDF$ scoring scheme, it is often required to calculate the similarity between two documents. In section 1.2.1, several measures are listed to calculate this difference. Euclidean distance, for example, can be used as the simplest approach. However, it has a limitation that two documents that are very similar as per the semantics (meaning) of their content can have Euclidean distance much higher. To overcome this limitation, Cosine distance is generally used, which measures the cosine value of the angle between two document vectors. It is given in equation 1.5 and is again defined as follows for two documents d_1, d_2 having document vectors $\vec{d_1}$, $\vec{d_2}$ respectively.

$$sim(d_1, d_2) = \frac{\vec{d_1}.\vec{d_2}}{|\vec{d_1}||\vec{d_2}|} \tag{2.3}$$

Numerator in equation 2.3 is the dot product between the document vectors $\vec{d_1}$, $\vec{d_2}$ which is given by $\sum_{i=1}^{m} d_{1i}d_{2i}$, where m is the number of components in each vector. Denominator in equation 2.3 denotes the product of magnitude of each vector, which is given as $\sqrt{\sum_{i=1}^{m} x_i^2}$ for any vector $\vec{x}$. The denominator helps to normalize each document vector to unit length so as to neutralize the effect of document length on similarity.

As a final step in text clustering, any clustering algorithm reviewed in section 2.2 is applied on a term-document matrix formed using statistical features like term frequency, $TF-IDF$ scoring, or any other suitable scoring mechanism. A comparison of clustering algorithms especially for text documents was performed in [70]. The authors showed that

Table 2.1: Semantic relations in WordNet.

Relation Name	Syntactic category	Examples
Synonymy (similar)	Noun, Verb, Adjective, Adverb	(document, papers), (cluster, bunch)
Antonymy (opposing)	Noun, Verb, Adjective, Adverb	(tall, short), (light, dark)
Hyponymy (is-a relation)	Noun	(cluster, knot), (document, certificate)
Meronymy (part-of)	Noun	(car, engine), (tree, trunk)
Troponymy (manner-name)	Verb	(talk, orate), (eat, slurp)
Entailment	Verb	(eat, chew, masticate)

bisecting k-means algorithm performs better than the K-means and hierarchical clustering algorithms when compared on eight benchmark datasets. However, this kind of approach does not incorporate semantic features or characteristics of textual data. As a result of which many issues such as word sense disambiguation, polysemy and synonymy, etc. are not taken into account. In the next subsection, techniques involving semantic features of the text are reviewed.

2.1.3 WordNet and Lexical Chains

To discover semantic relations between different words in a text, an openly available[1] hierarchical database known as WordNet is generally used in various text processing tasks. WordNet is defined and briefly explained as follows:

"WordNet is a database that links English nouns, verbs, adjectives, and adverbs to sets of synonyms that are in turn linked through semantic relations that determine word definitions" [71]. Each set of synonyms is called a *synset*. WordNet 2.1 consists of a total of 155,327 words organized into 175,979 synsets. Each synset contains a definition known as *gloss* along with few example sentences. Additionally, each of the synsets is linked to other synsets using some semantic relation. Table 2.1 lists and explains all the semantic relations that WordNet has along with the syntactic category (part-of-speech) in which it exists [71].

In earliest papers using semantic feature-based document categorization [72, 73], a word sense disambiguation strategy using WordNet is used to deal with polysemy, and syn-

[1]https://wordnet.princeton.edu/

onyms of terms were included in the global set of features to deal with synonymy. In [72], specifically three improvements namely: *i.* Disambiguation using part-of-speech tagging, *ii.* including synonyms and *iii.* including hypernyms in the Bag of Words model were separately analyzed. Using a fixed level of hypernyms and simple disambiguation strategies were its limitations. In [74] WordNet was used for mapping each word of a document to one of the 41 lexical categories provided by WordNet. This was done to reduce the high dimensionality of the feature vector. In it, another information resource called ANNIE [75] was used for extraction of co-referenced words such as: "*Tony Blair, Mr. Tony Blair, Prime Minister*" etc. Then in [76], the importance of using only *nouns* as cluster features were highlighted and a more complex *Word Sense Disambiguation Strategy* (WSD) was used using WordNet ontology. The results in it highlighted that disambiguating polysemous and synonymous nouns yields better clustering performance. In a recent research paper[77], a similar WSD technique with some modification in semantic similarity measure was proposed. A semantic similarity measure computes the similarity between two senses. Various similarity measures for two given senses exists in literature [78] like path length, wu-palmer distance, Leacock-Chodorow, and Resnik, etc.
Afterward, lexical chains which are widely used for the task of document summarization had also been used for clustering of documents [79]. "*Lexical chaining is the process of grouping and identifying such words together to form chains which in turn will help in identifying and represent the topic and content of a document.*" These are based on semantic relations such as synonyms, hypernyms (hyponyms), Holonyms (meronyms), etc. Importance of *Lexical chains* was initially realized for finding the structure of a text to determine its meaning. In [80], lexical chains were computed as an indicator of text structure using a thesaurus as the knowledge base. In [81] lexical chains were computed using $WordNet$ specifically for the detection of *malapropisms* (existing words with the same sound as the correct word but with a different meaning and which generally goes undetected by a spell checker.) [82] presented an efficient lexical chain computation technique and used it for automatic *text summarization*. In a few papers, the importance of lexical chains for document clustering had also been realized [83, 79, 77].

2.1.4 Word Embeddings

Representation of a word is a very trivial task required in almost all Natural Language Processing (NLP) tasks. Conventionally, given a fixed size vocabulary $V = w_1, w_2, w_3, ..., w_n$, each word w is represented as a vector $\vec{w}$ of length of vocabulary size $|V|$. Each index of this vector corresponds to a word in the vocabulary, thus for a word only one component value is 1 and all others are zeros in its vector. Mathematically, it

can be given

$$\vec{w_i} = \begin{cases} 1, & \text{if } w = w_i \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

This representation is called the *one-hot* representation and has been used widely in NLP tasks due to its simplicity. However, it has some drawbacks, firstly, it is high dimensional and sparse representation because most of the values remain zero. Secondly, it does not capture semantic relatedness among words. Thirdly, if new words get added to the dictionary, the size of the vector gets increases for each word, thus it is not flexible [84].

To overcome these limitations of one-hot representation and also to provide a mechanism by which words can be compared among themselves, a *distributed representation* of a word known as *word embeddings* was introduced Tomas Mikolov in 2013 at Google [4]. Here, distributed representation means that the semantics of a word is mapped to a vector having continuous real values based on the context of the word. This is based on distributional hypothesis [85, 86] which says that words with similar context are semantically similar to each other. This representation is also *dense*, meaning that the semantics of a word is represented by more than one dimension of the vector. Additionally, this representation is flexible in contrary to one-hot representation because the length of the vector for all words remains fixed even if a new word is added to the vocabulary. Formally, word embedding can be defined as "a dense, distributed and fixed-length vector used to represent the semantics of a word using its context."

A general idea for the generation of word embeddings is to optimize an objective function such that the probability of a central word in a context window of a fixed size m is maximized (in the case of Word2Vec algorithm [4]). This is done by training a neural network architecture for a large corpus of text. The output of the network architecture is a numerical vector (or embedding) corresponding to a word. Other algorithms for word embeddings are Glove developed by Stanford university [87] and FastText by Facebook [88]. These all are open source projects and thus can be freely downloaded[1,2]. Several recent studies present a good survey on various algorithms used to generate word embeddings [89, 90, 91, 92]. The dimensionality of this vector generally lies from one hundred to one thousand. Several algorithms proposed in the literature since 2013 are described below.

[1]https://nlp.stanford.edu/projects/glove/
[2]https://fasttext.cc/

2.1.4.1 Word2Vec

Given a large corpus of text as a sequence of words, Word2Vec finds word vectors (embeddings) such that they can best predict the context words $w(t+j)$ for a given center word $w(t)$ in a window to fixed-size m. This idea is represented mathematically in the form of an objective function given in the following equation[89].

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j}|w_t;\theta) \tag{2.5}$$

where,

$$P(w_{t+j}|w_t;\theta) = \frac{\exp(\mu_o^T v_c)}{\sum_{w\in V}\exp(\mu_w^T v_c)} \tag{2.6}$$

The objective function in equation 2.5 is the average negative log-likelihood function in which θ denotes all the variables to be optimized. This optimization is generally done by minimizing it using stochastic gradient descent function [93]. The probability of occurrence of a context word $w(t+j)$ given a center word $w(t)$ is given as shown in equation 2.6. The numerator in equation 2.6 contains the exponential value of dot product of word vectors of a context word and the center word. This value is normalized (in the denominator) by summation of exponential values of the dot product of center word vector v_c and each word vector μ_w in the entire vocabulary V. However, the calculation of denominator becomes computation process very slow, hence the process of *negativesampling* [94] is applied in which a modified objective function to make the process efficient. This type of model of Word2Vec is known as *Skip-gram* model. In the original paper, [4], the second variant of Word2Vec was also proposed which is known as the Continous Bag Of Words (CBOW) model. In the CBOW model, the objective function tries to maximize the probability of a target center word given the context words. Figure 2.2 shows the two models of the Word2Vec algorithm.

2.1.4.2 Global Vectors (GloVe)

As compared to the one-hot representation of words, the Word2Vec model provides vectors of much smaller size (300) and better captures the semantics and relationship between words. However, Pennington et. al [87] observed that Word2Vec can capture information only from a local context of words ignoring the information from a global context.

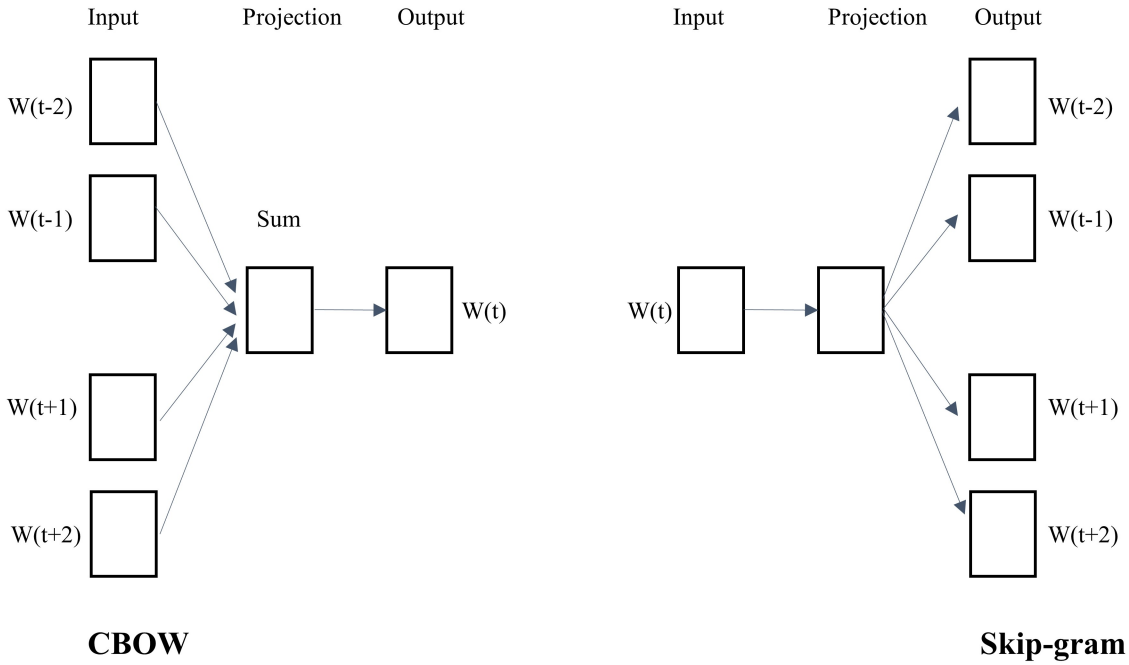

Figure 2.2: Architectures of Word2Vec model[4].

Hence, they proposed a different model for word vectors named *Global Vectors* (GloVe) [87]. In GloVe, information from the global context is used with the help of the global co-occurrence matrix X in which each element X_{ij} represents the co-occurrence frequency of the i^{th} word and j^{th} word. This observation is formulated into the following objective function to calculate word vectors.

$$J = \sum_{i,j} f(X_{ij})(v_i^T v_j + b_i + b_j - logX_{ij}) \tag{2.7}$$

In above equation [87], v_i and v_j are word vectors corresponding to i^{th} and j^{th} word respectively which are to be learned. b_i and b_j are word-specific biases that also need to be learned. All these parameters are learned by minimizing J for all these parameters. Function $f(x)$ is used as a weight function to not over-weight very frequent words and rare words. It is defined as follows.

$$f(x) = \begin{cases} (\frac{x}{x_{max}})^{\alpha}, & \text{if } x < x_{max} \\ 1, & \text{if } x \geq x_{max}. \end{cases} \tag{2.8}$$

where x_{max} (upper cutoff for co-occurence frequency) and α can be tuned for a given dataset.

2.1.4.3 FastText

Word2Vec and GloVe models fail to provide word vectors for words that are out of vocabulary (OOV) which means the words that do not appear in the training dataset. In [88], a model for word representation and text classification tasks was introduced and made publicly available as a library named fastText[1]. The architecture of the model is similar to Word2Vec but uses subwords (or character n-grams) to find vector representations and related words for queries of OOV words. For example, the word "gearshift" which may not exist on Wikipedia is broken down into subwords which are all the substrings contained in it like "gea", "ear", "rsh", and "shi" etc. The length of subwords is controlled using *minn* and *maxn* parameters. Each of these subwords gets a vector corresponding to them and the final vector for the complete word is achieved by summing up the vectors of these subwords. In comparison to Word2Vec and GloVe models, fastText is fast in its training on large corpora for classification tasks such as tag prediction and sentiment analysis [95, 88].

2.1.4.4 Bidirectional Encoders Representations using Transformers (BERT)

Representation of words and sentences in a way that can truly capture their meaning according to the context in which they fall is a rapidly evolving area of research in the field of Natural Language understanding. An important recent milestone in this direction was reached in late 2018 with the introduction of BERT. BERT is a deep learning model that made new records in dealing with language-based tasks such as sentence/sentiment classification, question answering system and Named Entity Recognition (NER). Soon after the paper release [96], various versions of BERT were open-sourced[2]. These versions are already pre-trained on huge datasets of books and Wikipedia. Hence, one can use these models as it is or also can fine-tune them for different supervised task mentioned before, to generate context-based embeddings.

A high-level architecture of the BERT model is shown in Figure 2.3. Basically, it is a stack of Transformer (encoder) layers. Two architectures $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ with 12 and 24 encoder layers respectively have been proposed in the original paper. The model takes as input a sequence of words, the first of which is a special token "[CLS]". The minimum length of the input sequence can be 1 and the maximum length is 512. Each encoder layer of BERT outputs a vector that is passed as input to the layer above it. For each individual word of the input sequence, the $\text{BERT}^{\text{base}}$ and $\text{BERT}_{\text{large}}$ model

[1] https://github.com/facebookresearch/fastText
[2] https://github.com/google-research/bert

gives a vector of length 768 and 1024 respectively as its final output. These vectors encode in them, the semantics and relationship among them. These vectors can be used for different supervised downstream tasks such as for question answering systems and sentiment analysis. This is generally done by adding an additional neural network layer plus a softmax function at the end of the model. The original paper reports outstanding results for these kinds of tasks in comparison to other state-of-the-art models.

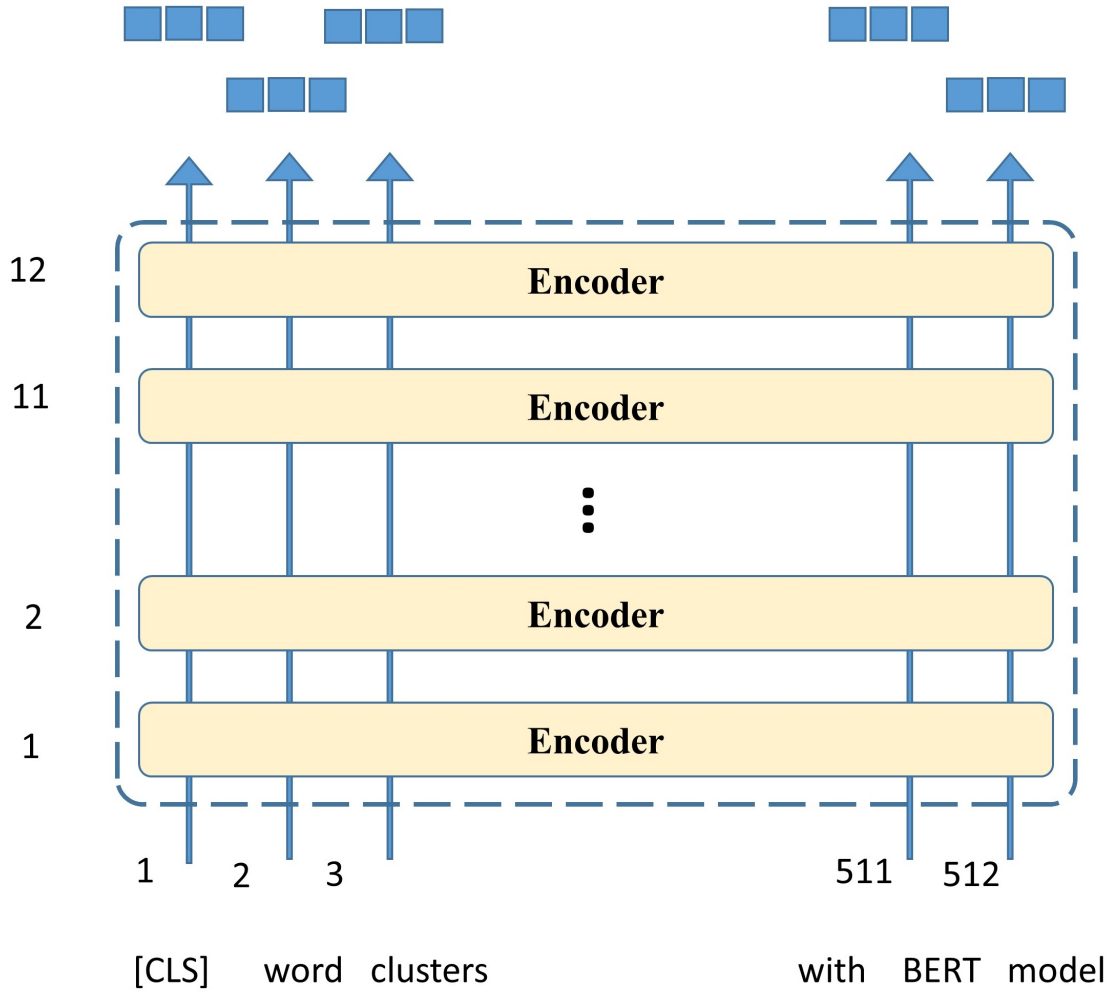

Figure 2.3: Architecture of BERT [5].

2.2 Clustering Techniques

In this section, a description of various clustering algorithm categories along with their advantages and disadvantages is provided.

2.2.1 Classification of clustering techniques

Fig. 2.4 shows a broad classification of clustering algorithms [8]. A brief description of various categories is as follows:

2.2.1.1 Partitioning based

Let a dataset D contain n number of objects. Given a value k (where $k \leq n$), partitioning methods partition the n objects into k clusters $C_1, C_2, ..., C_k$. The following conditions are to be satisfied by the obtained partitions: (1) none of the clusters should be empty and (2) each object must be contained in either one (Hard c-means) or more than one (Fuzzy c-means) clusters. First, k cluster centers are chosen either randomly or by using some more sophisticated methods, and then a relocation method is used to shift the cluster centers towards an optimal solution, for instance, (1) in K-means [97], the average value of all data points in the cluster is used to find the new cluster center whereas (2) k-medoids [98] represent a cluster by an object that is located near to the center of the cluster. The quality of clustering is measured by an objective function. This objective function is designed to achieve high intracluster similarity and low intercluster similarity. Other well-known algorithms in this category are: K-modes [99], PAM [100], CLARA [101] and FCM [102]. A detailed explanation of some widely used partitioning clustering techniques are as follows:

1. **K-means clustering algorithm**

   (a) A set of k points from the dataset D are chosen randomly as centers representing k clusters.

   (b) Each point is assigned to the cluster whose center is at the minimum distance from it using Euclidean distance.

   (c) Cluster centers are recomputed using current cluster memberships.

   (d) Go to step (ii) till the membership to clusters stops changing.

   Several variants of K-means can be found in the literature [9]. Some of these targets to initialize cluster centers more efficiently to reach a global optimum, whereas others use a different objective function. However, K-means and other similar algorithms of this type tend to get trapped in locally optimal solutions. This limitation was overcome when in 1990, the data clustering problem was solved by the application of nature-inspired metaheuristic algorithms by using simulated annealing

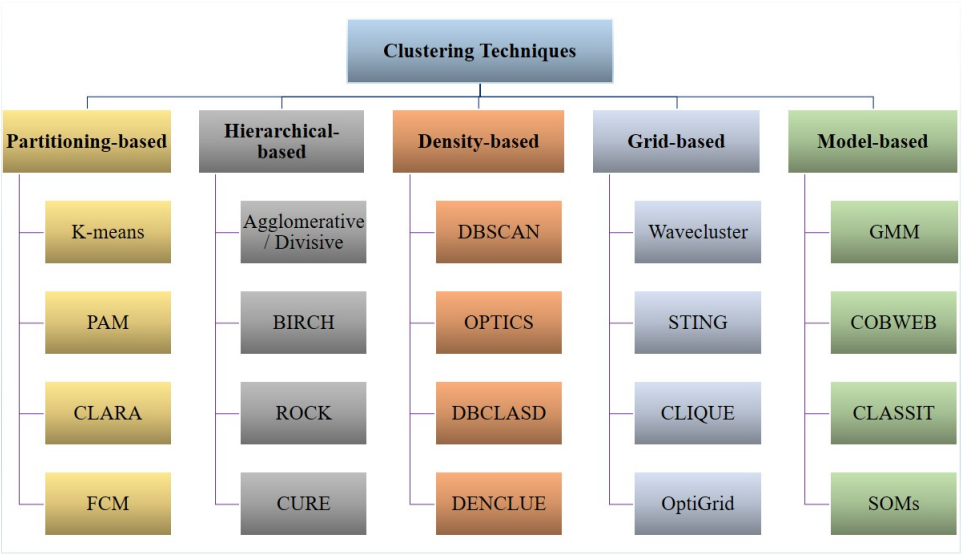

Figure 2.4: Broad classification of clustering algorithms.

[103]. A similar recent approach using the gravitational search algorithm (GSA) is presented in [104]. A detailed survey of the usage of these algorithms for data clustering is presented in [105]. In recent research, deep neural networks have been used to achieve a non-linear mapping of the original feature space to solve the problem of high dimensionality [106]. Learning of the parameters of the deep neural network and cluster centers was performed simultaneously by optimizing a KLD based objective function. However, because of simplicity and less computational cost, k-means is still used widely.

When it comes to applications where a data object can belong to more than one category (i.e., when a degree of ambiguity or uncertainty is involved), the role of fuzzy cluster analysis comes into the picture to provide a better partitioning of data objects. Fuzzy cluster analysis allows the degree of membership of an object to a cluster to be measured in the range [0, 1] (denoted by μ). This concept of degree of membership allows greater flexibility to express the belongingness of data objects to multiple clusters [107]. Given a dataset $X = \{x_1, x_2...x_n\}$ these memberships lead to the output of the clustering process to be a fuzzy label vector of degree of memberships to all clusters for each data point x_j. The fuzzy label vector can be represented as $\mu_j = (\mu_{1j}, \mu_{2j}, ..., \mu_{cj})^T$. The c x n matrix $U = (\mu_{ij})$ is called a fuzzy partition matrix, where c is the number of clusters and n is the total number of data points.

2. Fuzzy c-means algorithm

Let $D = \{x_1, ..., x_n\}$ be the set of data points and the number of required clusters be c $(1 < c < n)$. To find a fuzzy partition matrix $U = (\mu_{ij})$, Dunn [108] introduced the FCM algorithm. Bezdek [109] later improved this algorithm. In FCM, the goal of finding the optimum fuzzy c-partition matrix U is encoded with an objective function J_m [110] as

$$J_m(U, v) = \sum_{k=1}^{n} \sum_{i=1}^{c} (\mu_{ik})^{m'} (d_{ik})^2 \tag{2.9}$$

where,

$$(d_{ik})^2 = (d(x_k - v_i))^2 = \sum_{j=1}^{m} (x_{kj} - v_{ij})^2 \tag{2.10}$$

Here, m is the dimension of the dataset. The weighting parameter m' controls the extent to which the membership is shared between clusters. It ranges in $[1, \infty)$. When $m' = 1$, the membership value μ_{ij} is either 0 or 1. By contrast, when $m' \to \infty$, the value of $J_m \to 0$. In general, the bigger the value of m' is, the more the fuzzier is the partition matrix U.

An optimal partition matrix U corresponds to the minimum value of this objective function J_m, which is a solution to the following equation:

$$J_m^*(U^*, v^*) = \min J(U, v). \tag{2.11}$$

under the constraints

$$\sum_{j=1}^{n} \mu_{ij} > 0, \forall i \in \{1, ..., c\}, \tag{2.12}$$

and

$$\sum_{i=1}^{c} \mu_{ij} = 1, \forall j \in \{1, ..., n\} \tag{2.13}$$

An iterative algorithm introduced by Bezdek (1981) [109] popularly known as the FCM algorithm to get a solution to this equation is as follows:

(a) Choose c $(1 < c < n)$ and m'. Initialize the partition matrix $U^{(0)}$. As in [110], here, each step is labelled by r, where r = 0,1,2,...

(b) Calculate the c centers $v_i^{(r)}$ as:

$$v_{ij} = \frac{\sum_{k=1}^{n} (\mu_{ik})^{m'} . x_{ki}}{\sum_{k=1}^{n} (\mu_{ik})^{m'}} \tag{2.14}$$

(c) Update the partition matrix as follows:

$$\mu_{ik}^{(r+1)} = \left[\sum_{j=1}^{c} \left(\frac{d_{ik}^{(r)}}{d_{jk}^{(r)}} \right)^{\frac{2}{m'-1}} \right]^{-1} \tag{2.15}$$

(d) If $||U^{(r+1)} - U^{(r)}|| \leq \varepsilon_L$, stop; else, go to step 2 and make $r = r + 1$.

The next subsection presents different variants of FCM which are observed in the literature in the past three decades.

Variants of FCM based on distance functions:

Euclidean distance used in FCM had been replaced many times by some other measure leading to better clustering results. An attempt to cover them comprehensively is given below.

(a) Gustafson-Kessel algorithm:
The Euclidean distance which is originally used in FCM favors clusters that are spherical. In [111], it was replaced by the Mahalanobis distance, thus the algorithm could find clusters of arbitrary shape. The Mahalanobis distance related to a cluster i is given by the equation

$$d^2(x_j, C_i) = (x_j - C_i)^T \sum_i^{-1} (x_j - C_i) \tag{2.16}$$

where $\sum_i$ is the covariance matrix of the cluster. Cluster centers and membership degrees are calculated in the same way as in original FCM. The covariance matrix [112] is updated as

$$\sum_i = \frac{\sum_i^*}{\sqrt[p]{det(\sum_i^*)}} \tag{2.17}$$

where,

$$\sum_i^* = \frac{\sum_{j=1}^{n} \mu_{ij} (x_j - c_i)(x_j - c_i)^T}{\sum_{j=1}^{n} \mu_{ij}} \tag{2.18}$$

However, owing to matrix inversions, computational costs are higher for this

algorithm in comparison with FCM [112].

(b) Pedrycz-97:
Early semi-supervised fuzzy clustering techniques as in [113] also used Mahalanobis distance. In [113], a total of three experiments were performed based on three different datasets. The results of that study showed that when Mahalanobis distance was used, the convergence rate was the highest among all algorithms used.

(c) Kernel-based clustering:
In 2003 [114], a Fuzzy Kernel C-Means algorithm (FKCM) was proposed to deal with datasets that may consist of non-spherical clusters. In kernel-based clustering, the original feature space is transformed into a high-dimensional feature space. The transformation of space is denoted as ϕ: $X \rightarrow F$, where X is the original feature space and F is the transformed feature space. The transformed data is denoted by $\phi(x)$. In FKCM, FCM is integrated with a mercer kernel function to capture the non-spherical shape of clusters (such as the annular ring shape). The results presented in [114] showed that for spherical datasets, FCM and FKCM perform equally well, but for annular ring-shaped datasets, FKCM clusters more effectively.

(d) S^2KFCM:
In 2004 [115], using Gaussian kernel, a new Semi-Supervised Kernel Fuzzy C-Means (S^2KFCM) algorithm was introduced by Zhang et al. Gaussian kernel is given by the equation

$$K(x, y) = \exp\left(\frac{-||x - y||^2}{\sigma^2}\right) \tag{2.19}$$

Experiments conducted on benchmark datasets indicated that better classification results were obtained using S^2KFCM than other classical algorithms such as K-NN and SVM [115].

(e) Bouchachia et. al.:
In 2006 [116], Bouchachia et al. investigated the effect of four different distance measures named Euclidean, weighted Euclidean, fully adaptive, and kernel-

based distance with the same objective function. After performing experiments on three datasets (fully labeled), it was found that the relative performance (high to low) was in the order of usage: fully adaptive(e.g., Mahalanobis distance), weighted Euclidean, kernel-based distance, and Euclidean distance. Thus, the use of fully adaptive distance yielded the best results.

(f) Lai and Garibaldi:
In 2011 [117], Lai and Garibaldi compared four algorithms with different objective functions (Pedrycz-97 [113], Li-08 [118], Zhang-04 [115] and Endo-09 [119]) by using different distance metrics, namely Euclidean distance, Mahalanobis distance and Gaussian kernel-based distance. They indicated that Pedrycz-97 and Li-08 perform better than others owing to the presence of Mahalanobis distance. Because of the presence of the inverse of covariance matrix in Mahalanobis distance, different scales of variables are normalized, and the correlation between features is also handled. The results also showed that the infinity problem arises when using Euclidean distance with high dimensional datasets.
Based on the literature reviewed in this paper, in the next section(4) a concise view of factors upon which an appropriate selection of a proximity measure depends is presented. Based on the factors reviewed, a procedure for the same has also been proposed.

2.2.1.2 Hierarchical based

These methods perform a hierarchical breakdown of a given dataset which can be classified as agglomerative and divisive. In agglomerative methods, initially, each object is regarded as a cluster on its own and they are then successively merged till they satisfy a termination condition. By contrast, in the divisive approach, initially, the set of objects is considered as a single large cluster and is successively split up into smaller clusters until a termination condition is satisfied. The former is also called the bottom-up approach whereas the latter is called the top-down approach. A general algorithm for agglomerative clustering is as follows.

Agglomerative clustering algorithm

1. Let each data point be a cluster on its own.

2. Compute the proximity matrix of individual points.

3. Merge the two closest clusters and then update the proximity matrix.

4. Repeat step (iii) until a single cluster remains.

In the form of output, a hierarchical clustering algorithm yields a tree-like structure known as a *dendogram* which can be broken at different levels to give corresponding different data clusterings. Depending on how the inter-cluster similarity is defined, three important agglomerative hierarchical clustering algorithms include *single-linkage, complete linkage* and *average linkage*. Single linkage algorithm uses the distance between the *closest* pair of data points in clusters as a measure of inter-cluster similarity. Complete linkage algorithm uses the distance between the *farthest* pair of data points as the inter-cluster similarity; while the average linkage algorithm uses the distance between the *group average* of all data points contained in a cluster as the proximity measure between clusters. Fig. 2.5 [6] shows a dendogram created as a result of single linkage clustering applied on seven data points. Other popular hierarchical algorithms include BIRCH [120], ROCK [121], CURE [122] and Chameleon [123].

2.2.1.3 Density based

Methods described above find the clusters based on a proximity measure and hence face difficulty while finding clusters of arbitrary shape [8]. On the other hand, density-based methods discover clusters based on density. These methods can find clusters of arbitrary shapes. Here, a cluster is kept growing as long as the number of data objects in the neighborhood exceeds some threshold value. In any density-based clustering method, *density* at some point p is defined to be the number of data points lying in a circle of radius *eps* around p. Also, if a circle of radius *eps* consists of some minimum number of data points denoted by *minpts* then the region is called the dense region. A *core point* is a point that consists of a dense region around it. A border point is a point that has points less than *minpts* around it but itself lies inside the neighborhood of a core point. Lastly, a point that is neither a core point nor a border point is known as a noise point. DBSCAN [124], a basic density-based algorithm can be abstracted as follows [125].

DBSCAN algorithm

1. Identify all core points.

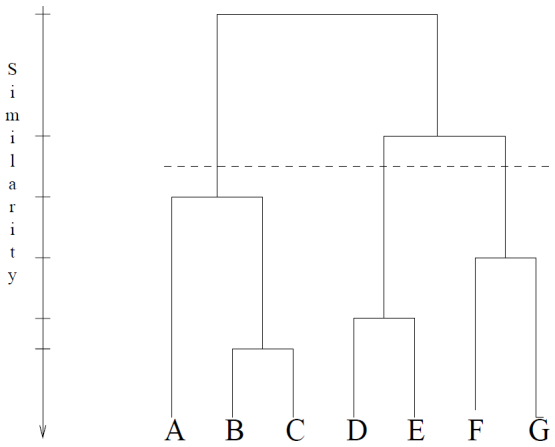

Figure 2.5: A dendogram showing clustering hierarchy on 7 points [6].

2. Assign neighboring core points into a single cluster.

3. **For each** non-core point **do**

   If possible, assign it as a border point to the cluster of the closest core point otherwise, add it to noise.

Other algorithms of this category include OPTICS [126], DBCLASD [127] and DENCLUE [128].

2.2.1.4 Grid based

Here, each dimension is divided into several cells thus forming a grid structure between dimensions. Clustering operations are then performed on this quantized space. The processing time of these methods is independent of the number of objects. Rather, it is determined by the number of cells in the grid structure. STING [129] , Wavecluster [130], CLIQUE [6] and OptiGrid [131] are well-known examples of this category.
An overview of CLIQUE which is one of the initial algorithms in this category is presented as follows [8].

1. Partition the d-dimensional data space into non-overlapping rectangular units(or cells) and identify dense units in all subspaces based on a density threshold l.

2. Dense cells in each subspace are then used to generate clusters by starting with an arbitrary dense cell and finding the maximal region covering the cell and working on the remaining dense cells.

A more detailed description of this algorithm can be found in [132].

2.2.1.5 Model based

These methods perform clustering by first hypothesizing a mathematical model and then finding its best fit for a given dataset. For example, the EM algorithm performs an expectation-maximization analysis [133], COBWEB performs a probability analysis [134] and a neural network based method, Self-Organizing Maps (SOMs) [135], performs clustering by mapping high dimensional data onto a 2-D or 3-D feature map. CLASSIT [136] is an extension algorithm of COBWEB for continuous-valued data.

The advantages and disadvantages of the aforementioned clustering algorithms determined from the literature are summarized in Table 2.2. In [137], a performance comparison of these algorithms is presented according to the size of datasets and time taken for cluster formation. In [20], a performance comparison of partitioning algorithms such as K-means and K-medoids, based on different proximity measures was performed; however, the range of data dimensionality did not cover datasets of much larger dimensionality (such as textual data). In this study, experiments to study and compare the performance of various clustering algorithms of different categories are presented (section 5) which cover a considerably wide range of data dimensionality. Having reviewed proximity measures in Section 2 and clustering algorithms in Section 3.1, in the next subsection, partitioning-based algorithm FCM is reviewed especially for proximity measures.

2.2.2 Semantic text clustering techniques

Apart from the clustering techniques mentioned in section 2.2 which are directly applicable to the VSM representation of text, various semantic text clustering techniques have been defined in the literature. Most of these techniques exploit WordNet to extract the semantics of text, while some use Wikipedia database as well. A few other techniques use some domain-specific ontology such as defined by L.Yue et al. [138]. A complete summary based on several important factors identified from the related research papers is provided in Table 2.3. Important fields used in Table 2.3 are defined below.

1. Solves synonymy
   A checkmark (✓) indicates that the technique uses the synonymy relation between words to be more efficient, while a crossmark (✗) indicates that it does not use synonymy relation.

Table 2.2: Advantages and disadvantages of different types of clustering techniques.

Clustering technique	Advantages	Disadvantages
Partitioning based	Algorithms converge faster and are robust to noise.	Tends to produce only convex clusters. Difficult to work with nominal/ordinal attributes.
Hierarchical based	Can find nonconvex clusters. The number of clusters is not required.	Fails in the presence of noise and is more time-consuming. It requires large memory space for large datasets.
Density-based	Arbitrary shaped clusters can be discovered and the number of clusters is also not required. Has the ability to treat outliers as noise robustly.	Clustering quality highly depends on an appropriate selection of parameters. In the case of varying density, algorithms do not perform well (in the case of DBSCAN). It also suffers from the curse of dimensionality.
Grid-based	More suitable to high dimensional datasets. The order of input of records has no negative effect.	Clustering quality is highly dependent on the size and number of grid cells.
Model-based	There exists a choice for appropriate statistical models to capture latent clusters. Data points are not explicitly assigned instead they have a probability of belongingness to multiple clusters.	EM algorithm can be considerably expensive if there are a large number of distributions and the algorithm does not guarantee to get settled on a global optimum (a more serious concern in high dimensions.).

2. Solves polysemy

   A checkmark (✓) indicates that the technique uses the polysemy relation between words to tackle the ambiguity problem, while a crossmark (✗) indicates the non-usage of the same.

3. Semantic source

   This field tells the usage of any semantic source (Wikipedia, and WordNet etc.) by the technique.

4. Dimensionality

   This field tells about the actual dimensionality used by the technique for clustering the documents. The "High" value implies no mechanism is used to reduce the dimensionality, while the "Medium" value implies some mechanism has been used to reduce the dimensionality.

5. Language independency

   In this field, a checkmark (✓) indicates that the technique is applicable for any language dataset while a crossmark (✗) indicates that it is for a specific language only (mostly English).

6. OOV words

   This field tells whether the technique is capable of handling the out of vocabulary (OOV) words. OOV words are those words that are not contained in the vocabulary of the semantic source used.

Table 2.3: A summary of various semantic text clustering techniques.

S.no.	Authors	Solves Synonymy	Solves Polysemy	Semantic source	Dimensionality	Language independency	OOV words	Reference
1	T. Wei et al.(2015)	✓	✓	WordNet	Medium	×	×	[77]
2	Y. Li et al.(2015)	×	×	Nil	High	✓	×	[139]
3	L. Yue et al.(2015)	✓	×	Dairy Ontology	Medium	×	×	[138]
4	J.A. Nasir et al.(2013)	✓	×	WordNet/ Wikipedia	Medium	×	×	[140]
5	C. Bouras et al.(2012)	✓	×	WordNet	Medium	×	×	[141]
6	S. Fodeh et al.(2011)	✓	✓	WordNet	Medium	×	×	[76]
7	A. Huang et al.	✓	✓	Wikipedia	High	×	×	[142]
8	R. Baghel et al.(2010)	✓	×	WordNet	Medium	×	×	[143]
9	C. Luo et al.(2009)	×	×	Nil	High	✓	×	[144]
10	D.R. Recupero et al.(2007)	✓	×	WordNet/ ANNIE	Medium	×	×	[74]
11	D. Jayarajan et al.(2007)	✓	×	WordNet	Medium	×	×	[79]

2.3 Research Gaps in Text Clustering

Based on the literature review presented in previous sections, the following research gaps have been found in text clustering techniques.

2.3.1 High dimensionality and sparsity

Vectors resulting from the BOW model lead to a very high dimensional representation of documents, for example, it may reach in order of 10^5 for some thousands of documents. This also leads to a very sparse term-document matrix that affects the accuracy of clustering techniques [145, 146]. Techniques such as *Latent Semantic Indexing* (LSI) [147] and Topic modelling [148, 149] reduces the original space to a lower-dimensional space. However, dimensionality reduction techniques exploiting more recent distributed [94] representation of words do not much exist.

2.3.2 Semantic ambiguities

Dealing with problems such as Synonymy (different words with the same meaning) and polysemy (the same word with different meaning) often decreases the accuracy of text clustering. Various approaches have explored methods to derive semantic relations using WordNet ontology [77, 72, 73]. However, these approaches are highly dependent on word coverage and the design of WordNet [71]. Additionally, these approaches are mainly useful for only a few languages. Hence, text clustering methods covering these limitations are required.

2.3.3 Use of distributed representation of words

Almost all the text clustering techniques use a Bag of Words model in which every single word is treated as a dimension. As a result, the term-document matrix becomes very sparse. However, with the invention of word embeddings in which a word is represented with a dense vector, semantic relations between words can use to combine the words around a single topic. This area of research has not been much explored especially for text clustering tasks except for only a few research studies [150, 151].

2.3.4 Clustering of large text datasets

When the number of the document becomes very large in a corpus, above mentioned problems like high dimensionality, sparsity, and semantic ambiguities, etc. become even more challenging for a text clustering task. For large datasets, efficient clustering methods that can use recently proposed context-based dense representations (based deep learning) such as Embeddings from Language Models (ElMo) [152] and BERT [96] are required. These types of embeddings can capture the semantics of a word based on its context.

2.4 Problem Formulation

On sufficiently large textual datasets of very high dimensionality (tens of thousands or more), traditional clustering techniques can not perform well. Most of the research gaps as highlighted exist in currently existing text clustering techniques. The target of this thesis is to develop such text clustering techniques which have the following features.

1. More accurate as compared to the existing techniques.
2. Includes semantics of text to solve the ambiguity problem.
3. Scalable to big datasets having high dimensionality.
4. Effectively handle the sparsity issue of textual datasets.

2.5 Research Objectives

Research objectives defined for this thesis are as follows.

1. To perform an in-depth study and analysis of text clustering techniques for big data analysis.
2. To propose efficient text clustering techniques for big datasets.
3. To implement the proposed techniques for big datasets.
4. To validate the efficiency of proposed techniques in terms of suitable metrics like NMI, F-measure and run time, etc.

   The concerned chapter number of this thesis and the publication for each of these objectives are given below (The publication number is in accordance with the order of publications given in "List of Publications" at the end of this thesis).

(a) Literature review as per Objective 1 is mainly reported in chapter 2 “Literature Review”, and published in Publication 1.

(b) A part of objective 2 is achieved in chapter 3 “Proposed Clustering Technique: Stamantic Clustering (STC)” and another part of it is achieved in chapter 4 “Proposed Clustering Technique: WEClustering”. A part of the work done corresponding to objective 2 is published in Publication 2 and another part is published in Publication 3.

(c) A part of objective 3 is achieved in chapter 3 “Proposed Clustering Technique: Stamantic Clustering (STC)” and another part of it is achieved in chapter 4 “Proposed Clustering Technique: WEClustering”. A part of the work done corresponding to objective 3 is published in Publication 2 and the other part is published in Publication 3.

(d) Objective 4 is achieved completely in chapter 5 Experimental Results and Validation of Proposed Clustering Techniques. These findings are partially published in Publication 2 and also in Publication 3.

Chapter 3

Proposed Clustering Technique: Stamantic Clustering (STC)

In this chapter, a novel text clustering technique is proposed based on the literature survey and the research gaps presented in the previous chapter. This proposed technique attempts to use two types of features of the text. First, the semantic features of the text, which relate to the meaning of the text, and second, are the statistical features of the text. The first kind of features is generated with the help of lexical chains formed using WordNet. The second kind of feature is incorporated using TF-IDF scores of words in the text. The proposed technique is named "Stamantic Clustering" (STC) based on these two types of features.
In this chapter, section 3.1 presents the overall working of STC with the help of an architecture diagram. Then, section 3.2 presents the detailed working of each phase of STC. Lastly, section 3.3 gives the implementation details of STC such as datasets used, parameter settings applied, performance metrics used to measure the performance, and the techniques compared with STC.

3.1 STC

STC, the proposed text clustering technique can be broadly divided into three phases which are as follows.

Phase 1: This phase consists of pre-processing and word sense disambiguation.

Phase 2: This phase consists of extraction of statistical and semantic features.

Phase 3: This phase generates clusters based on the inputs received from phase 2.

Figure 3.1 shows these phases of STC along with the sequence of processes involved in each phase. In the first phase, pre-processing steps (explained in section 1.4) such as punctuation removal, stop-words removal, tokenization, and lowercasing are performed on the set of raw documents. In previous works [77, 76, 82] only the nouns were given utmost importance for features extraction but in this work, both nouns and adjectives are

considered. The second step in this phase is to disambiguate all the words extracted in the first step. The Word sense disambiguation technique used in this paper is proposed in [77]. In the second phase, statistical scoring of concepts[1] is performed using TF-IDF. The list of concepts corresponding to nouns and adjectives is then used to create lexical chains in step ii. In step iii. semantic scoring of lexical chains is done using a novel scoring scheme. In step iv., the lexical chains are used to create a chain-document matrix. Finally, in the third phase, clustering is performed on this data matrix. In the next section, a detailed description of STC's phases is presented.

3.2 Phases of STC

STC consists of seven different processes which are Pre-processing, Word sense disambiguation, Statistical scoring based on TF-IDF, Lexical chaining, Semantic scoring of lexical chains, Formation of the chain-document matrix M and Clustering using K-means upon the data matrix M. These seven processes are grouped into three phases namely

1. Pre-processing and word sense disambiguation.
2. Extraction of statistical and semantic features.
3. Generation of document clusters.

These are described in detail in the following subsections.

3.2.1 Pre-processing and word sense disambiguation

The two processes contained in this phase are described as follows.

1. *Pre-processing*

   Let $D = \{d_1, d_2, d_3, ..., d_n\}$ be the corpus that consists of a set of n raw documents. In pre-processing phase, each document d_i goes through the following sub-processes in sequence: punctuation removal, tokenization, lower-casing, stop words removal, part of speech tagging, and the extraction of nouns and adjectives. The output of this phase is denoted by $D' = \{d'_1, d'_2, ..., d'_n\}$ which is the set of pre-processed documents. Each d'_i is the set of tokens all of which are either nouns or adjectives. It is denoted as $d'_i = \{na_1, na_2, na_3, ..., na_n\}$. Most of the previous research works

[1]The term concept here, refers to the sense of a word in a document. Sense and concept are used interchangeably.

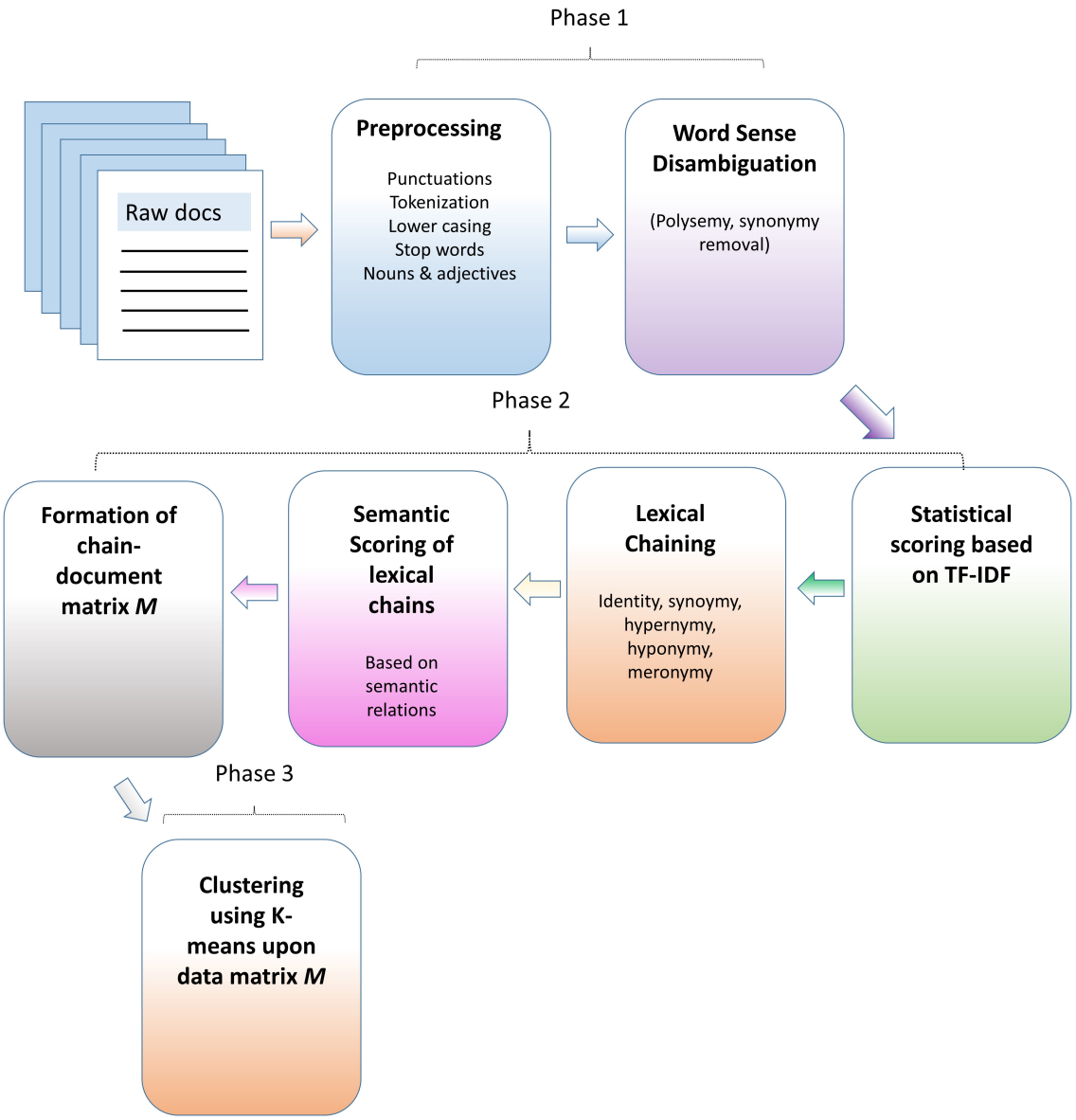

Figure 3.1: Architecture of the proposed text clustering technique: STC.

on semantic document clustering involved only nouns as document features. In this research work, adjectives which are an important part of speech in the English language are also used. It is found in this research work that adjectives play an important role in deciding the document category.

2. *Word Sense Disambiguation*:

Word sense disambiguation (WSD) is a process that determines (or disambiguates) the actual sense of a word in a document according to the context surrounding the term. Basically, it deals with the problems of *synonymy* and *polysemy*. Synonymy refers to the situation when multiple distinct words refer to a single sense or concept. Thus WSD maps those multiple words to a single sense. Polysemy is the situation when a single word may have different senses according to the surrounding context. Thus WSD decides a single sense of the word as per the context. Researches [76] shows that WSD significantly improves the accuracy of text mining tasks such as information retrieval, topic mining, document clustering, and classification, etc. Reviewing all the techniques of WSD goes beyond the scope of this research work. A quite recent WSD procedure mentioned in [77] is followed in this research work with addition nouns and adjectives both are used than just nouns. The procedure is originally proposed in [153] and is described as follows:

Let $d'_i = \{na_1, na_2, na_3, ..., na_n\}$ be the set of all pre-processed nouns and adjectives for a document d_i in corpus D and let $S_i = \{s_{i1}, s_{i2}, ..., s_{in}\}$ be the set of all senses corresponding to any token na_i. Each sense s_{in} contains the definition, examples, synonyms plus the definition of the senses related to it with relations of hypernym, hyponym, meronym, and holonym as found in WordNet ontology. The most appropriate sense $\hat{s_i}$ from this set is determined by the following equation.

$$\hat{s_i} = \underset{s_{ik} \in S_i}{\arg\max} \sum_{na_j \in d'_i} \underset{s_{jm} \in S_j}{\arg\max} \; sim(s_{ik}, s_{jm}) \tag{3.1}$$

This equation determines the most appropriate sense of a noun or an adjective na_i by calculating the sum of its similarity to all other senses of the same part of speech in a document d'_i. As stated in [76], a restriction is made up to only the top three senses of each term to make the process less computationally expensive. Furthermore, this task is done only once and then stored for later use. Also, this can be done in parallel as the task is for a single document and thus independent of other documents. To calculate the semantic similarity $sim(s_{ik}, s_{jm})$, many measures exist in the literature such as Path length, Leakcock-Chodorow, Wu-palmer,

and Resnik similarity, etc. In this paper, a quite recent semantic similarity measure as proposed by [77] has been followed. The said similarity measure combines the implicit as well as explicit semantic relationships using WordNet ontology. The implicit semantic relationship between two senses involves characteristics such length of the path between two senses. Its examples include Path Length and Wu-palmer semantic similarity measure. Whereas explicit relationships between two senses exploit relationships such as identity, synonym, hyponym, hypernym, holonym, and meronym, etc. The mathematical representation of this measure is:

$$sim(s_1, s_2) = \frac{2d + D}{l_1 + l_2 + D} \tag{3.2}$$

where

$$D = \frac{1}{sift4(str_1, str_2)} \tag{3.3}$$

i.e. inverse of sift4 [154] distance between two strings str_1 and str_2. str_1 and str_2 are the string descriptions of two senses s_1 and s_2 for which similarity is to be calculated. d is the depth of the of LCS from the root in the WordNet hierarchy. l_1, l_2 are the path lengths between LCS and s_1, s_2 respectively.

3.2.2 Extraction of statistical and semantic features

1. *Statistical scoring of concepts*:

   The output of the aforementioned steps results in disambiguated concepts for each term in each document of the corpus. In this step, TF-IDF term scoring is performed for each document. Let us again define a document $d'_i = \{s_1, s_2, s_3, ..., s_n\}$ as the set containing all the disambiguated senses s_i. In this step, for a document d'_i, TF-IDF scores for each of the sense is calculated using the following equation:

$$TF - IDF(s_i) = freq(s_i) * (\log(\frac{|D| + 1}{doc\_count(s_i) + 1}) + 1) \tag{3.4}$$

   Here, $|D|$ is the total number of documents in the corpus D, and $doc\_count(s_i)$ is the total number of documents that contain the term corresponding to sense s_i. These values are stored in the form of a map of keys and values where keys are senses s_i and values are $TF - IDF(s_i)$ values. This mapping of statistical scoring

of terms is used in the third step of this phase.

2. *Lexical chaining*:

   After taking benefit from statistical scoring in the above step, now semantic information will be used in this step to build lexical chains using WordNet. Lexical chains have been successfully used in [77] to extract a small subset of semantic features. In this work, for building the lexical chains, the procedure as described in [155] has been followed. Algorithm 3.1 depicts this process of formation of lexical chains. It uses four kinds of semantic relations which are identity, synonymy, hypernymy(hyponymy), and meronymy. However, in this research paper, adjectives are also included in disambiguated concepts. It should also be mentioned here that WordNet supports the semantic relations of hypernymy, hyponymy, and meronymy only for nouns. So for adjectives, only identity and synonymy will be used.

3. *Semantic scoring of lexical chains*:

   After construction of lexical chains, final scoring of lexical chains contained in a document are performed in this step. This step incorporates the semantic information from the lexical chains as well as the statistical information in the form of TF-IDF score mapping which was obtained in step i. Let $d'_{i\_map} = \{s_1 : ti_1, s_2 : ti_2, s_3 : ti_3, .., s_n : ti_n\}$ be a document represented as a map between the disambiguated senses s_i and their TF-IDF values ti_i. Let TF-IDF score of a sense s_i be denoted as $score(s_i)$.
   Score of a lexical chain is defined as the sum of all the concepts (senses) contained in it. Let $L_{d_i} = \{l_1, l_2, ..., l_n\}$ be the set of lexical chains corresponding to a document d_i . Also, let l_i contains the senses $\{s_1, s_2, ..., s_n\}$. Mathematically, it is given as

$$Score\_chain(l_i) = \sum_{s_i \in l_i} score(s_i) \tag{3.5}$$

3.2.3 Generation of document clusters

Once the lexical chains from each document d_i of the corpus D are extracted, they are combined in this step to create a global set of features denoted as $Feat_{global} = \{f_1, f_2, ..., f_n\}$. This set is finally used to create a chain-document matrix $M_{mxn} = (a_{ij})$, where m is the total number of documents in the corpus and n is the length of the total number of unique chains in the whole corpus. The values a_{ij} are computed using the formula given by equation 3.5. Each feature vector in this matrix is further normalized

Algorithm 3.1 Algorithm for construction of lexical chains

Input: document as a list of senses s_i, $d'_i = [s_1, s_2, ..., s_n]$

Output: Set of lexical chains L

procedure LexChain(d')
Initialize: $L = \{\}$, $V = \text{set}(d'_i)$, $E = \{\}$, $G = (V,E)$
  for each s_i in d'_i **do**
    for each s'_i in d'_i **do**
      if s_i in $identity(s'_i)$ OR s_i in $synonymy(s'_i)$
      OR s_i in $hypernym(s'_i)$ OR s'_i in $hypernym(s_i)$
      OR s_i in $meronym(s'_i)$ OR s'_i in $meronym(s_i)$ **then**
        $E(s_i, s'_i) = 1$
      end if
    end for
  end for
  L = Connected components(G) ▷ Each connected component is a lexical chain
end procedure

Algorithm 3.2 Algorithm for phase 2 and phase 3.

Input: List of all documents $D = [d'_1, d'_2, d'_3, ..., d'_n]$ where each d'_i is a list of disambiguated senses s_{ij} i.e. $d'_i = [s_{i1}, s_{i2}, ..., s_{in}]$.

Output: document features,

Initialize: $Feat_{global} = \{\}$
for each d'_i in D **do**
  Initialize: $d'_{i\_map} = \{\}$, $L = \{\}$
  for each s_{ij} in d'_i **do**
    $ti_{ij} \leftarrow TF - IDF(s_{ij})$ ▷ according to Eqn. 4.2
  end for
  Add pair $(s_{ij} : t_{ij})$ as an element to d'_{i_map}
  $L \leftarrow$ LexChain(d'_i) ▷ Identify the lexical chains
  for each l in L **do**
    Add l to $Feat_{global}$
  end for
end for
Keep only the unique items in $Feat_{global}$.
for each d'_i in D **do**
  for each l_j in $Feat_{global}$ **do**
    $M[i][j] \leftarrow Score\_chain(l_j)$ ▷ according to Eqn. 3.5
  end for
end for
Perform clustering on M using K-means clustering algorithm.

to a unit scale using L2 normalization. This normalized matrix is used to cluster the documents using the K-means algorithm. It is because of the usage of lexical chains instead of single words that the STC technique can reduce dimensionality to a great extent. Algorithm 3.2 describes phases 2 and 3 in an algorithmic form.

3.3 Implementation of STC

3.3.1 Datasets used

To validate the efficiency of the proposed STC technique, it has been implemented to generate clusters in a variety of datasets. Table 3.1 shows different characteristics for each dataset such as the number of documents, and the number of classes, etc. A description of each dataset is given as follows.

1. *Reuters_15_50*: This dataset is prepared as a subset of Reuters-21578[1] a dataset that consists of a minimum of 15 and a maximum of 50 documents in every class.

2. *Reuters_15_500*: Similar to above but with a minimum of 15 and a maximum of 500.

3. *Reuters_408_3945*: Similar to (i.) but with a minimum of 408 and a maximum of 3945.

4. *Articles_253*: This dataset consists of 253 articles obtained from 5 different publication houses. This dataset has been previously used in [156]. The publication houses which also correspond to the 5 classes of this dataset are Transactions on Mobile Computing, American Political Science Review, Monthly Weather Review, British Food Journal, and DNA Research.

5. *Classic_3*[2]: This collection consists of 3891 articles distributed into three classes namely: aerodynamics, medical and computing algorithms.

6. *Pubmed_4k*: This collection of the dataset consists of 4000 medical articles equally distributed among 4 categories namely: alzheimer, cancer, diabetes, and HIV. (see footnote 3 to download.)

7. *Scopus_2.8k*: This dataset is originally prepared using titles and abstracts of articles in the Scopus database. The total number of articles is 2800 equally distributed

[1]Available at: https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization +collection. Accessed: 2020-02-03.

[2]Available at: https://vhasoares.github.io/downloads.html. Accessed: 2020-02-03.

Table 3.1: Characteristics of datasets used for validating STC.

S.no	Dataset name	Total categories	Total documents
1.	Reuters_15_50	24	280
2.	Reuters_15_500	55	2172
3.	Reuters_408_3945	6	3624
4.	Articles-253	5	253
5.	Classic-3	3	3891
6.	Pubmed_4k	4	4000
7.	Scopus_2.8k	7	2800
8.	Webkb	4	3877
9.	Newsgroups	NA	700

among 7 categories namely: investment, neural network, hyperactivity, photosynthesis, proton, concrete, and tectonic plates. (see footnote 3 to download.)

8. *Webkb*[1]: This is a dataset of webpages widely used in text mining community. Originally it consists of 7 classes, however 4 classes namely: course, department, faculty and student are used in this paper.

9. *Newsgroups*[2]: This dataset is extracted from a widely used 20 Newsgroups dataset. The extracted dataset consists of 700 documents and the corresponding categories are not taken in the experiments. This dataset is used to demonstrate the Elbow method to determine the number of clusters in this dataset.

3.3.2 Parameter settings

At the time of implementation, a few parameter values are required by the proposed STC clustering technique as well as other techniques used in the comparison. K-means clustering performed in phase 3 of STC and other techniques requires no. of clusters k which is set to be the same as the number of classes. For each dataset, k-means clustering is repeated 25 times and the maximum score is reported. This is because the clustering results depend on random data points which are chosen as initial clusters. In the next section, metrics used to measure the performance of all clustering techniques are given.

[1]Available at: http://www.cs.cmu.edu/~webkb/. Accessed: 2020-02-03.
[2]Available at: https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups. Accessed: 2020-11-12.

3.3.3 Performance Metrics

Measures of clustering performance are generally categorized into two types: Internal and External validity measures. Internal measures are used when no ground truth labels are available whereas external measures are used when clustering performance is to be measured against available ground truth labels. Metrics used here are defined below:

1. *Silhouette coefficient*:
   Silhouette coefficient measures the quality of clustering when truth labels are not available which is the actual case of clustering. It is a widely used metric that measures how dense and well separated the clusters are. It is given by:

$$s = \frac{b - a}{\max(b - a)} \tag{3.6}$$

   In this equation, a is the average distance between a sample and all other points in the cluster. b is the average distance between a sample and all other points of the next nearest cluster. The range of the Silhouette coefficient is [-1,1], where a higher value indicates better clustering.
   Additionally, if the k value is not available for k-means clustering, a method called as "Elbow method" is generally used to find the optimal k value [8]. In this method, the Silhouette coefficient (or similar scores) are plotted against different values of k. In the resulting graph, the most significant turning point (also known as the elbow) often indicates the optimal k value. Here, this method is used for the Newsgroups dataset for which categories are not available.

2. *Purity*:
   Given the cluster assignments and the actual class labels, purity is calculated by first counting the number of documents (our case) in a cluster of the class which is the most frequent in this particular cluster [31]. This value is summed over all the clusters and is divided by the total number of documents. Formally, it is given as:

$$Purity = \frac{1}{N} \sum_{c \in C} \arg\max_{d \in D} |c \cap d| \tag{3.7}$$

   Where C is the set of clusters, D is the set of actual classes and N is the total number of documents. It ranges in $[0, 1.0]$ where better clusterings have a greater value.

3. *AMI*: It is an adjusted version of Mutual Information (MI) for the impact of the number of clusters and number of samples on metrics of clustering performance evaluation. For two clusterings $C1$ and $C2$, AMI [26] is given as:

$$AMI(C1, C2) = \frac{[MI(C1, C2) - E(MI(C1, C2))]}{[avg(H(C1), H(C2)) - E(MI(C1, C2))]} \tag{3.8}$$

Where MI is given by

$$MI(C1, C2) = \sum_{i=1}^{|C1|} \sum_{j=1}^{|C2|} \frac{|C_i \cap C_j|}{N} \log \frac{N|C_i \cap C_j|}{|C_i||C_j|} \tag{3.9}$$

$H(C)$ is the entropy associated with a clustering C and $E(MI(C1, C2))$ is the expected Mutual information (refer [26]) between two clusterings $C1$ $C2$. Value of AMI ranges in [0, 1] where a large value implies better clustering.

3.3.4 Techniques compared

To assess the effectiveness of STC, its comparative analysis is performed with several other clustering techniques which are listed below.

3.3.4.1 Bag of All Words (BOAW)

In this, features of the term-document of the corpus are composed of all the words coming as a result of the following pre-processes: punctuation removal, lowercasing, and stop words removal. Term frequency is used to score the features in a document. Once a term-document matrix is prepared, K-means [10] is applied to obtain document clusters.

3.3.4.2 Bag of nouns (BONW)

Here, the pre-processing techniques and the scoring scheme are the same as that of BOAW but instead of all the words, only nouns are used as features of the term-document matrix.

3.3.4.3 Bag of nouns and adjectives (BONA)

Here, adjectives are additionally used to nouns to create the features. Pre-processing and scoring are the same as those for BONW.

3.3.4.4 Disambiguated concepts (DC)

This is identical to BONW but in pre-processing, a Word Sense Disambiguation strategy is also used using Wu-palmer semantic similarity measure [77].

3.3.4.5 Disambiguated core semantics (DCS)

This approach used lexical chains to find the most important concepts in a document along with a WSD strategy [77].

In conclusion, this chapter described the details of the proposed text clustering technique STC. The details covered the architecture and the three different phases of STC. It also provided the implementation details of STC such as datasets used, parameter settings, performance metrics, and the compared techniques. In the next chapter, a second text clustering technique is proposed and similar details for it are provided.

Chapter 4

Proposed Clustering Technique: WEClustering

In this chapter, another text clustering technique is proposed which solves the problem of high dimensionality more precisely, which makes it even more suitable to Big datasets. This technique leverages the power of context-based numerical representation of words that are generated by a recently introduced deep learning architecture called BERT (already introduced in chapter 2). The proposed technique is named "Word Embeddings Based Clustering" (WEClustering). To cluster similar documents, WEClustering performs a semantic comparison between documents based on a very small number of concepts extracted from the whole corpus. Hence, WEClustering achieves better clustering accuracy in comparison to other state-of-the-art text clustering techniques.
Firstly, this chapter introduces the underlying idea of WEClustering with the help of a flowchart. Secondly, it gives the details of all phases of WEClustering. Finally, it provides the implementation details such as datasets used, parameter settings, and performance metrics used for validating the efficiency of WEClustering.

4.1 WEClustering

WEClustering comprises of five different phases which are listed below.
Phase 1: Pre-processing.
Phase 2: Embeddings extraction and filtration.
Phase 3: Clustering of word embeddings.
Phase 4: Generation of Concept-Document (CD) matrix.
Phase 5: Clustering CD matrix.

All these phases of WEClustering are shown in the form of a flowchart in Figure 4.1. The Figure also shows the sub-processes involved in each of these phases. For example, the pre-processing phase consists of two sub-processes, first, lower-casing of the whole text and second, splitting the text into a list of sentences. The input to WEClustering is a corpus (a set of documents) and the output is the clusters of documents. The next

section describes each phase of WEClustering in detail.

4.2 Phases of WEClustering

The proposed clustering technique attempts to incorporate the context-based meaning of words (semantics) using state-of-the-art BERT architecture. The technique is divided into five different phases which are pre-processing, embeddings extraction and filtration, clustering of embeddings, generation of concept document matrix CD, and clustering the matrix CD. This process is shown in Figure 4.1. Each phase is described in the following subsections.

4.2.1 Pre-processing

In this first phase, all the documents are prepared in a format suitable for giving it as input to the BERT model. Firstly, all the documents are converted in lower case. Although BERT is capable of producing case-sensitive embeddings for words, to simplify, all the documents are converted in lower case. Secondly, each document is split into sentences. This is done because BERT finds contextually dependent embeddings for which it takes a complete sentence as its input. So the output of the first phase is a list of documents in which each document is a list of sentences contained in it.

4.2.2 Embeddings extraction and filtration

In this phase, firstly, the pre-processed data is fed into the pre-trained (weight parameters are fixed) BERT model. As a result of this, each word of all the documents is converted into a vector (embedding) of size 1024. Secondly, embeddings that are not so semantically important and hence do not play role in discriminating the documents are removed. These include embeddings corresponding to digits, punctuations, and stop words.
Algorithm 4.1 shows detailed steps corresponding to the two aforementioned phases.

4.2.3 Clustering of word embeddings

The conversion of all the words (string format) into a numerical vector format makes it very easy and accurate to measure similarity (or dissimilarity) between words. This kind of semantic comparison between words was not much accurate before the introduction of

Algorithm 4.1 Algorithm for pre-processing and extraction of BERT embeddings from a document.

Input: raw document in string format as variable *doc*, pre-trained BERT model as variable *model*, list of all stop words as *stop_words*, list of all punctuations as *punc*.

Output: list of word embeddings as *result*.

```
1: procedure PREPROCESS(doc, model)
2:
3:     sentences = doc.split('.')              ▷ split the document based on sentences.
4:     for each sentence s in sentences do
5:         s = s.lower()                                  ▷ convert into lower case.
6:         result ← model(sentences)
7:     end for
8:     r1 ← stop_words                          ▷ fetch the list of all stop words.
9:     r2 ← punc
10:    r3 ← digits
11:    remove = r1 + r2 + r3                          ▷ append r1, r2 and r3.
12:    for item in remove do
         remove word embedding from result corresponding to item.
13:    end for
         return result
14: end procedure
```

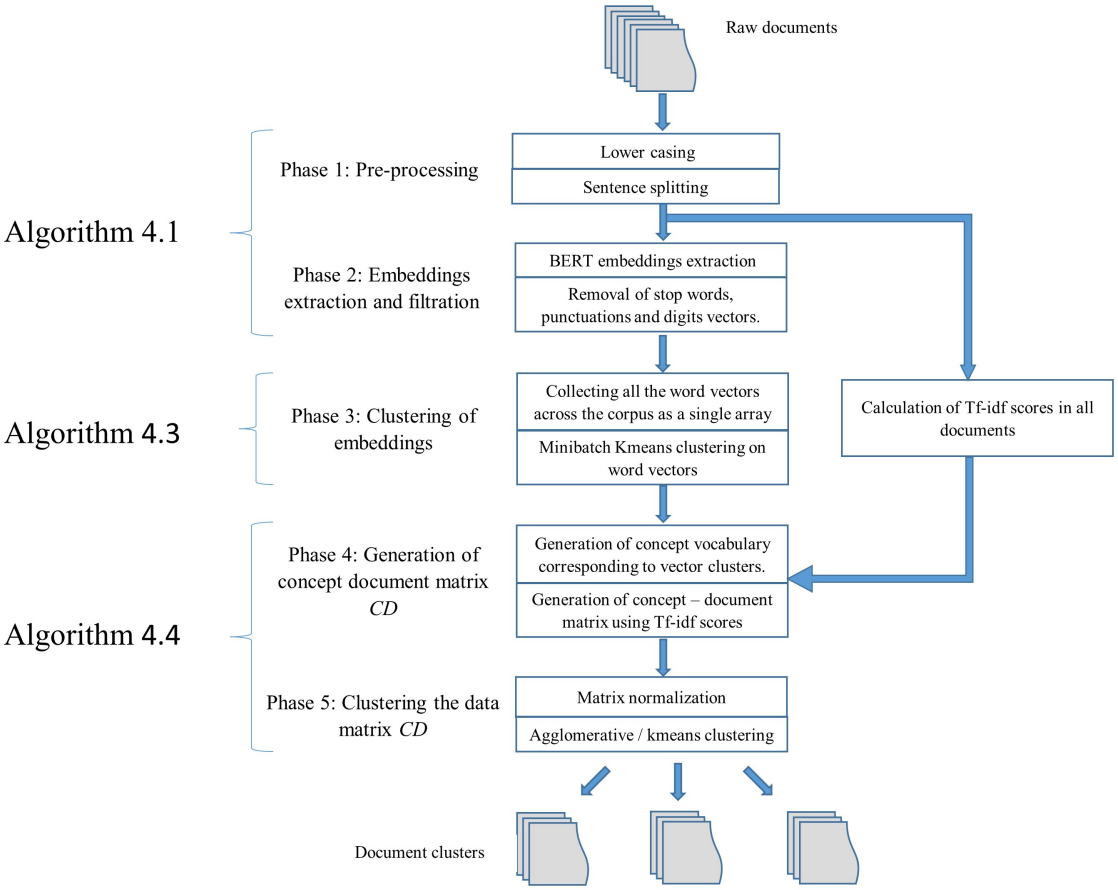

Figure 4.1: Flowchart of proposed the text clustering technique: WEClustering.

models like BERT. Hence, in this research, the difference in semantics of words based on embeddings is leveraged to form document clusters. In this phase, all the word vectors achieved out of Phase 2 are arranged in the form of a matrix of dimension (no. of words x 1024). Then, Minibatch K-means (Algorithm 4.2) clustering algorithm is applied to this matrix. As a result, clusters of words, now onwards called as a *concept* are formed. These clusters represent a unique theme contained in some documents. The idea is to use these concepts instead of individual words as a vocabulary (also called features) to represent any document. The total size of the vocabulary will be equal to the number of clusters k_{voc} that is empirically chosen for different datasets. Hence, the size of vocabulary gets reduced drastically from tens of thousands to less than one hundred.

This phase is shown in the form of Algorithm 4.3. The reason behind choosing this algo-

rithm is that it drastically reduces the computational time as compared to the standard K-means algorithm. As the name suggests, it uses mini batches instead of the complete dataset for each iteration of training.

Algorithm 4.2 Minibatch K-means.

Input: k, mini-batch size b, iterations t, data set X

```
1: procedure MiniBatchKmeans(X, k, b, t)
2:
3:     Initialize each c ∈ C with an x picked randomly from X
4:     v ← 0
5:     for i = 1 to t do
6:         M ← b examples picked randomly from X
7:         for x ∈ M do
8:             d[x] ← f(C, x)                    ▷ Cache the center nearest to x
9:         end for
10:        for x ∈ M do
11:            c ← d[x]
12:            v[c] ← v[c] + 1                   ▷ Update per-center counts
13:            η ← 1/v[c]                        ▷ Get per-center learning rate
14:            c ← (1-η)c + ηx                   ▷ Take gradient step
15:        end for
16:    end for
17: end procedure
```

4.2.4 Generation of Concept-Document (*CD*) matrix

After generating concepts in phase 3, each document now is represented in terms of all the concepts. As a result, all the corpus are collectively represented in the form of a matrix which is hereafter called a concept-document *CD* matrix. Each concept is given a score in each document to represent its degree of relation to that document. The scoring mechanism for an i$^{\text{th}}$ document d_i for j$^{\text{th}}$ concept c_j is represented by CD_{ij} and is defined as follows.

$$CD_{ij} = \sum_{k} TF - IDF(w_{jk}) \tag{4.1}$$

where

$$TF - IDF(w_{jk}) = freq(w_{jk}) * (\log(\frac{|D| + 1}{doc\_count(w_{jk}) + 1}) + 1) \tag{4.2}$$

Algorithm 4.3 Algorithm for generating the set of concepts from the corpus.

Input: number of concepts to find as k_{voc}, list of all documents as *corpus*, BERT embeddings for all documents as *corpus_bert_embeds*, b as batch size for clustering embeddings and t as number of iterations to be used in Minibatch K-means.

Output: Set of concepts as *vocab_set*.

procedure CONCEPT_EXTRACTION(k_{voc}, *corpus*, *corpus_bert_embeds*)
Initialize: TF-IDF_corpus = [ ], $vec\_for\_clustering$ = [ ], $word\_for\_clustering$ = [ ], *vocab_set* = [ ]
    for each doc in corpus **do**
        Assign doc_TF-IDF the TF-IDF scoring for each word in the document and store it as a map between the word and its score.
        Add doc_TF-IDF to the list TF-IDF_corpus.
        for each word in doc **do**
            Add word to the single list $word\_for\_clustering$.
            Add corresponding BERT embedding to the single list $vec\_for\_clustering$.
        end for
    end for
    embed_labels $\leftarrow$ MINIBATCHKMEANS(k_{voc}, $vec\_for\_clustering$, b, t)
    Generate cluster of words as *cluster_words* corresponding to clustering achieved as *embed_labels* in the previous step.
    Append *cluster_words* to *vocab_set*.
     return *vocab_set*, TF-IDF_corpus
end procedure

Here, TF-IDF values of all k words contained in the concept c_j corresponding to the document d_i are added together. $|D|$ is the total number of documents in the corpus D, $freq(w_{jk})$ is the frequency of word w_{jk} in document d_i and $doc\_count(w_{jk})$ is the total number of documents that contain the word w_{jk}. The size of matrix CD comes out to be (no. of documents x vocabulary size).

4.2.5 Clustering *CD* matrix

In this final phase, document clustering is performed by applying either hierarchical agglomerative clustering or k-means on concept-document matrix CD. Because the number of features that are used to represent a document is drastically reduced, a traditional algorithm like hierarchical agglomerative clustering or k-means performs nicely on the input matrix. As a result of this phase, well separated clusters of documents are achieved. Phases 4 and 5 are collectively shown in the form of Algorithm 4.4.

Algorithm 4.4 Getting document clusters.

Input: Set of concepts as *vocab_set*, TF-IDF scores for all documents as TF-IDF_corpus .

Output: Clusters of documents.

1: **procedure** DocumentClusters(k_{voc}, *corpus*, *corpus_bert_embeds*)
2: **Initialize:** Matrix of size (no. of documents x length of *vocab_set*) as CD
3: **for** each doc d_i in corpus **do**
4: **for** each concept c_j in *vocab_set* **do**
Assign CD_{ij} a value $\sum_k TF - IDF(w_{jk})$ using TF-IDF_corpus ▷ according to equation 4.2
5: **end for**
6: **end for**
7: Normalize the matrix CD.
8: Perform document clustering using either Kmeans or Agglomerative algorithm.
9: **end procedure**

4.3 Implementation of WEClustering

4.3.1 Datasets used

A total of seven benchmark datasets of different sizes and domains are used for validating the proposed technique. Relevant details of these datasets are summarized in the form of Table 4.1 and described below.

1. Articles-253

   This corpus[1] is a collection of five different categories of research articles. Each document consists of title, abstract, and references. The categories of this dataset correspond to the publication houses from which they are obtained. These are Transactions on Mobile Computing, American Political Science Review, Monthly Weather Review, British Food journal, and DNA research. The number 253 in the title depicts the total number of articles in this dataset.

2. Scopus

   This dataset is a part of a complete dataset and contains 500 articles. These articles are equally divided into 5 categories namely 'concrete', 'hyperactivity', 'investment', 'photosynthesis', and 'tectonicplates'. As per its name, it is obtained from scopus database and each document consists of a title and an abstract(see footnote 1 to download the complete dataset).

3. 20NG

   This dataset is a subset obtained out of widely used 20 newsgroups dataset[2] which consists of news articles of 20 different categories. The subset of categories included for this dataset are 'alt.atheism', 'talk.religion.misc', 'comp.graphics', and 'sci.space. The total number of documents in this dataset is 700.

4. Classic4

   This collection is made up of research articles of different domains which are aerodynamics, medical, computing algorithms, and information retrieval. In the implementation, however, only the first three categories are included because in the fourth category documents were very short. The total number of documents in this corpus is 800(see footnote 1 to download).

5. Scopus-long

   This is a collection of 2800 research articles from Scopus database containing the titles and abstracts. All the documents are equally divided into 7 categories each containing 400 articles. The categories are investment, neural network, hyperactivity, concrete, proton, photosynthesis, and tectonic plates(see footnote 1).

6. Classic4-long

   This is a large version of the Classic4 dataset consisting of 3891 documents divided into three categories.

7. 20NG-long

[1]Available at: https://vhasoares.github.io/downloads.html. Accessed: 2020-11-18.
[2]http://qwone.com/ jason/20Newsgroups/

Table 4.1: Properties of datasets used for validating WEClustering.

S.no	Dataset	Total categories	Total documents
1.	Articles-253	5	253
2.	Scopus	5	500
3.	20NG	4	700
4.	Classic4	4	800
5.	Scopus-long	7	2800
6.	Classic4-long	4	3891
7.	20NG-long	9	8131

This is a large part obtained from 20 newsgroups dataset(footnote 2). The categories included in this dataset are 'alt.atheism', 'talk.religion.misc', 'comp.graphics', 'sci.space', 'rec.motorcycles', 'rec.sport.hockey', 'sci.med', 'sci.electronics', 'talk.politics.misc'. Total number of documents in this corpus are 8131 divided into aforementioned 9 categories.

4.3.2 Parameter settings

Different parameter settings used in different phases of the proposed clustering technique are explained below.

1. Embeddings Extraction and Filtration (Phase 2): Two different BERT models are available to use:
   (a) $\text{BERT}_{\text{small}}$, that generates embedding vectors of size 768. Further, this can be case sensitive or case insensitive.
   (b) $\text{BERT}_{\text{large}}$, that generates embedding vectors of size 1024. This is available as only case-sensitive model[1]. In our approach, $\text{BERT}_{\text{large}}$ is used in the embeddings extraction and filtration phase.
2. Clustering of Word Embeddings (Phase 3): In this phase of the technique, the Mini-batch K-means algorithm is used to perform clustering of embeddings (as already explained). Important parameters of this algorithm are the number of clusters of words k_{voc}, batch size b, and number of iterations t. It is easy to determine the value of these parameters empirically. Table 4.2 lists the value of all these parameters used for all seven datasets. For the first six datasets, a value of 25 or 35 has been determined empirically to work with a batch size of 5000. In the largest dataset

[1]pypi.org

Table 4.2: Values of parameters k_{voc}, batch size b and number of iterations t used for each dataset in phase 3 (subsection 4.2.3).

S.no	**Dataset**	k_{voc}	b	t	**Total documents**
1.	Articles-253	35	5000	2000	253
2.	Scopus	35	5000	2000	500
3.	20NG	25	5000	1200	700
4.	Classic4	35	5000	2000	800
5.	Scopus-long	35	5000	2000	2800
6.	Classic4-long	35	5000	2000	3891
7.	20NG-long	75	25000	4000	8131

that is 20NG-long, a higher value of 75 was determined to work best with a batch size of 25000. Along with this, the number of iterations used to achieve the results is also listed.

3. Document Clustering (Phase 5): In the last phase of the proposed clustering technique that is document clustering, a simple clustering algorithm such as Agglomerative clustering or partitioning algorithm such as K-means is used. In Agglomerative clustering, an important parameter is 'linkage' as defined in subsection 2.1.1 of chapter 2. In this paper, ward linkage is used to complete the process of document clustering. While using K-means clustering, the parameter c that is the number of clusters is the number of classes/categories contained in the dataset. Also, as K-means is sensitive to the initialization of the centroids, the method of K-means++ [157] is used for initialization. Additionally, the number of times it is run is 10 and the best result is reported.

4.3.3 Performance Metrics

Several metrics are defined in the literature to assess the quality of clustering. Based on the availability of ground truth labels, they can be classified as external (when true labels are available) and internal (when true labels are not known). For assessment of the results produced in this research study, the following metrics are used.

1. Silhouette coefficient: This is a widely used metric when true labels are not available which is the actual case in a clustering task. It is a measure of how dense and well

separated the clusters are. Its mathematical formulation is given as:

$$s = (b - a)/max(b - a) \tag{4.3}$$

Where a is the average distance between a sample and all other points in the cluster and b is the average distance between a sample and all other points of the next nearest cluster. Its range lies between -1 and +1 (both inclusive). A higher value indicates dense and well separated clusters.

2. Adjusted Rand Index (ARI): ARI is a widely used metric for assessing cluster quality in the case of availability of true labels [30]. It can be used to measure two different clustering assignments that ignore different permutations of the same clustering. Two similar clusterings achieve a score near +1.0 and completely different clusterings achieve a score approaching -1.0.

3. Purity: This measure is also an external measure that calculates the quality of clustering by first assigning all the data points in a cluster to the class for which the maximum number of data points are present in this cluster. This is done and summed over all the clusters and then normalized by the total number of data points [31].

4.3.4 Techniques compared

For performing a comparative analysis of the proposed technique with other existing techniques, a comparison with the following clustering techniques based on the Bag of words model representation of documents is performed.

4.3.4.1 K-means

It is a popular partitioning based [10] clustering algorithm. Originally, it was proposed in 1967 [158] but because of its simplicity and less computational cost, it is widely used still.

4.3.4.2 Agglomerative clustering

This is a hierarchical type of clustering algorithm [159] in which data points are combined to gradually form clusters to give a tree-like structure known as dendogram [160]. This dendogram is cut at a specified level to give required clusters.

4.3.4.3 Hierarchical Density Based Spatial Clustering of Applications (HDBSCAN)

This algorithm [161, 162] is a robust variant of density-based clustering algorithms like DBSCAN [124]. It is a quite recent clustering technique that has shown better performance than several other algorithms.

4.3.4.4 Genie

It is quite recent hierarchical clustering algorithm [163] that form clusters based on Gini index [164] (a popular statistical measure used for measuring dispersion in a given list of frequency values) value. Genie makes sure that the value of the Gini index should not exceed a given threshold. If it exceeds then the smallest cluster is merged with its nearest neighbor.

In the next chapter, all the experiments conducted to validate the proposed text clustering techniques (STC and WEClustering) are presented. Also, an analysis and comparison of results with other text clustering techniques are provided.

Chapter 5

Experimental Results and Validation of Proposed Clustering Techniques

In the previous chapters, the design and implementation of the proposed text clustering techniques i.e. STC and WEClustering were discussed. To validate their efficiency, the proposed techniques have been implemented and tested on a variety of text datasets. Besides, their performances are compared with several other text clustering techniques. This chapter presents all the related experiments and also discusses the results obtained. Mainly, the performance comparison is performed based on the accuracy, execution time, and scalability.
Firstly, this chapter presents the analysis of the results individually for STC and WEClustering. Secondly, STC and WEClustering are compared among themselves based on several important factors such as their accuracy, execution time, and scalability.

5.1 Result analysis of STC

In this section, a discussion on the results of STC obtained by its implementation on different datasets along with a comparison with other techniques (already mentioned in chapter 3) is presented.

5.1.1 Accuracy

Analysis of the accuracy of STC for different performance metrics is as follows.

5.1.1.1 Silhouette coefficient

The Silhouette coefficient measures how dense and well separated the clusters are, especially when truth labels are not available. Hence, it is one of the most important clustering performance metrics (see chapter 1 for more details). In Table 5.1, the clustering performance of all the six techniques including STC is given as values of the Silhouette

Table 5.1: Silhouette coefficient values for clustering with different techniques.

S.no.	Datasets	BOAW	BONW	BONA	DC	DCS	**STC**
1.	Reuters_15_50	0.017	0.045	0.014	0.055	0.051	**0.085**
2.	Reuters_15_500	0.012	-0.026	-0.016	0.045	0.050	**0.077**
3.	Reuters_408_3945	0.082	0.087	0.091	0.032	0.033	**0.099**
4.	Articles-253	0.108	0.123	0.126	0.133	0.099	**0.247**
5.	Classic-3	-0.01	0.027	0.142	0.017	0.015	**0.400**
6.	Pubmed_4K	0.022	0.066	0.057	0.017	0.022	**0.070**
7.	Scopus_2.8k	-0.040	-0.022	-0.034	0.027	0.027	**0.047**
8.	Webkb	0.070	0.095	0.098	0.026	0.030	**0.102**
9.	Newsgroups	0.012	0.015	0.012	**0.032**	0.030	**0.032**

coefficient. The best values are indicated in bold. Owing to the capability of STC to reduce dimensionality, as well as efficiency due to a combination of semantic and statistical features, STC has outperformed all other clustering techniques. For the Newsgroups dataset, the DC technique also achieves the same good value of 0.032. In addition, as the number of categories (k value) is not available for Newsgroups dataset, hence k value is determined using Elbow method as described in subsection 3.3.3 of chapter 3. The range of k values taken is 2 to 8 and the elbow is detected at value 4. Hence, values listed in Table 5.1 for Newsgroups dataset are for k value 4. For elbow identification, the Silhouette coefficient values are plotted for different values of k in Figure 5.2. Figure 5.1 shows a column chart corresponding to values contained in Table 5.1.

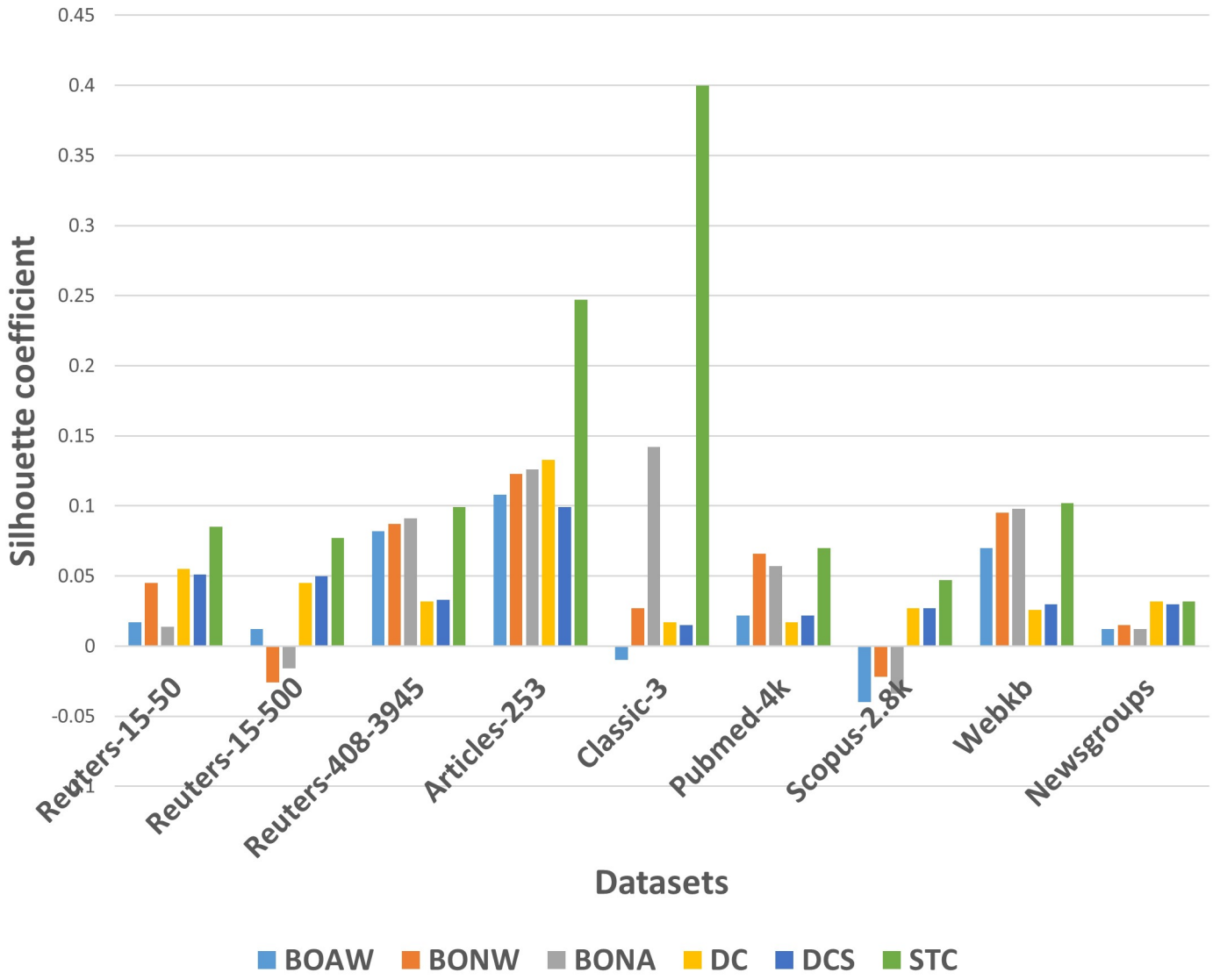

Figure 5.1: Results comparison of STC and other clustering techniques using Silhouette coefficient scores.

Table 5.2: Purity values for clustering with different techniques.

S.no.	Datasets	BOAW	BONW	BONA	DC	DCS	**STC**
1.	Reuters_15_50	0.418	0.475	0.446	0.629	0.570	**0.672**
2.	Reuters_15_500	0.447	0.490	0.466	**0.666**	0.599	0.640
3.	Reuters_408_3945	0.725	0.773	0.767	**0.849**	0.809	0.843
4.	Articles-253	0.909	0.909	0.901	0.992	0.945	**0.996**
5.	Classic-3	0.726	0.590	0.623	0.678	0.638	**0.802**
6.	Pubmed_4K	0.602	0.448	0.565	0.660	0.547	**0.845**
7.	Scopus_2.8k	0.782	0.766	0.785	0.863	0.475	**0.883**
8.	Webkb	0.464	0.479	0.466	0.615	0.560	**0.652**

5.1.1.2 Purity

Purity is an external measure of performance i.e. it measures the quality of clustering for available ground truth labels. In terms of purity, results are shown in the form of Table 5.2. The actual values show that the proposed STC technique outperforms all other techniques in six datasets out of eight. For other datasets as well, it does not lag much because it is only the DC technique that performs a little well. For all the other four techniques, STC outperforms all the datasets. Figure 5.3 shows these results in the form of a column chart which also indicates the high performance achieved by STC.

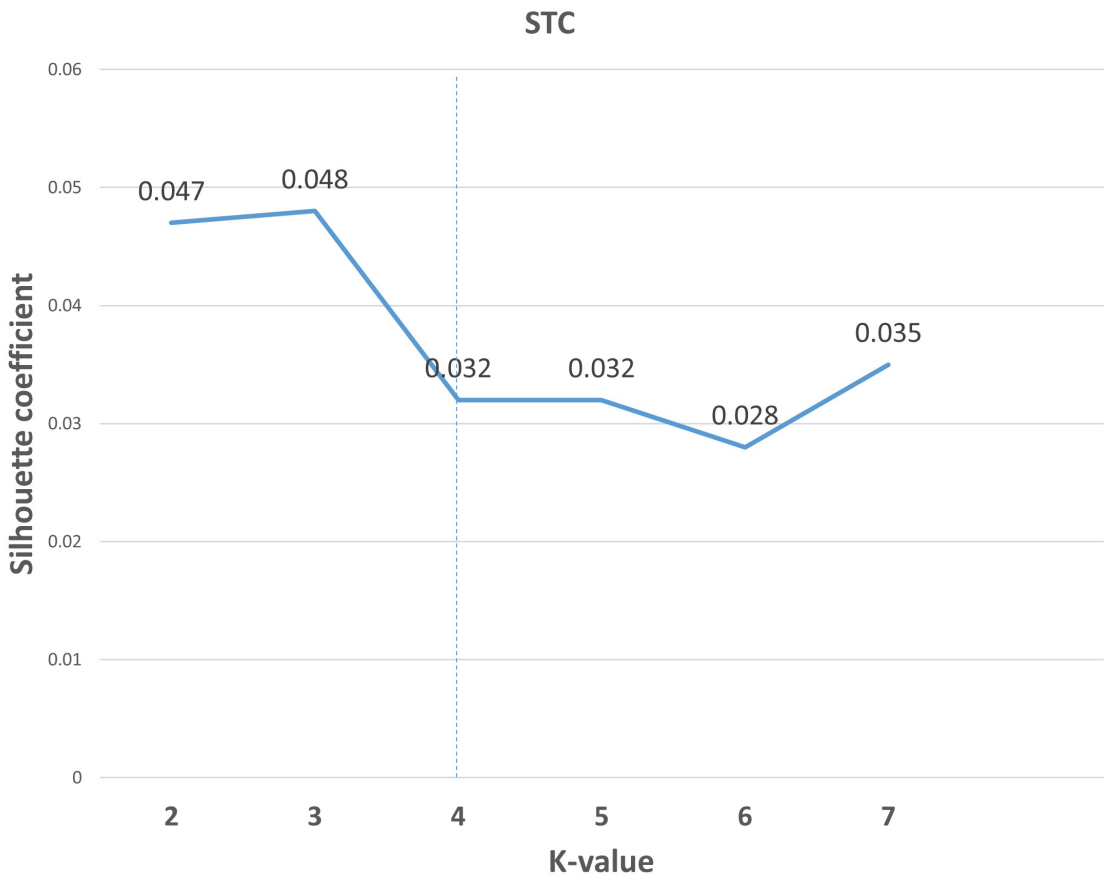

Figure 5.2: Elbow method executed on Newsgroups dataset.

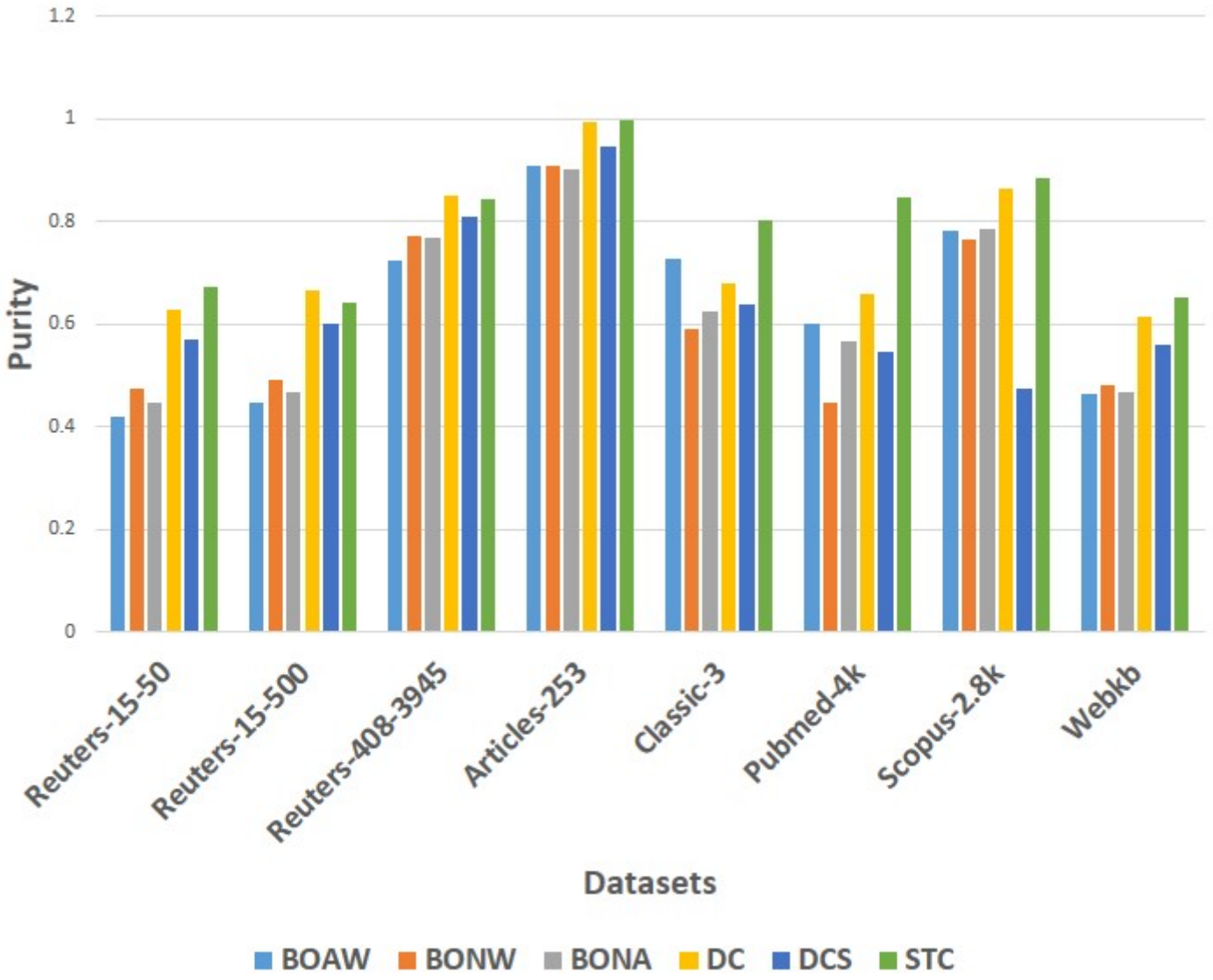

Figure 5.3: Results comparison of STC and other clustering techniques using Purity scores.

Table 5.3: AMI values for clustering with different techniques.

S.no	Datasets	BOAW	BONW	BONA	DC	DCS	**STC**
1	Reuters_15_50	0.323	0.372	0.311	0.457	0.430	**0.470**
2	Reuters_15_500	0.325	0.343	0.327	0.480	0.425	**0.495**
3	Reuters_408_3945	0.241	0.333	0.310	**0.449**	0.420	0.430
4	Articles-253	0.848	0.849	0.839	0.979	0.867	**0.988**
5	Classic-3	0.495	0.279	0.351	0.409	0.390	**0.501**
6	Pubmed_4k	0.361	0.182	0.349	0.395	0.250	**0.420**
7	Scopus_2.8k	0.697	0.641	0.738	**0.759**	0.399	0.748
8	Webkb	0.036	0.035	0.240	0.164	0.224	**0.255**

5.1.1.3 AMI

AMI is also an external measure of performance which measures the quality of clustering for available ground truth labels. In Table 5.3, AMI values are listed corresponding to all the experiments. Again, for the majority i.e. six out of eight datasets, the STC clustering technique achieved higher values than all other techniques. For the other two datasets also, it is only the DC technique that shows a good competition to STC. Figure 5.4 shows a column chart corresponding to the AMI values achieved by all the clustering techniques.

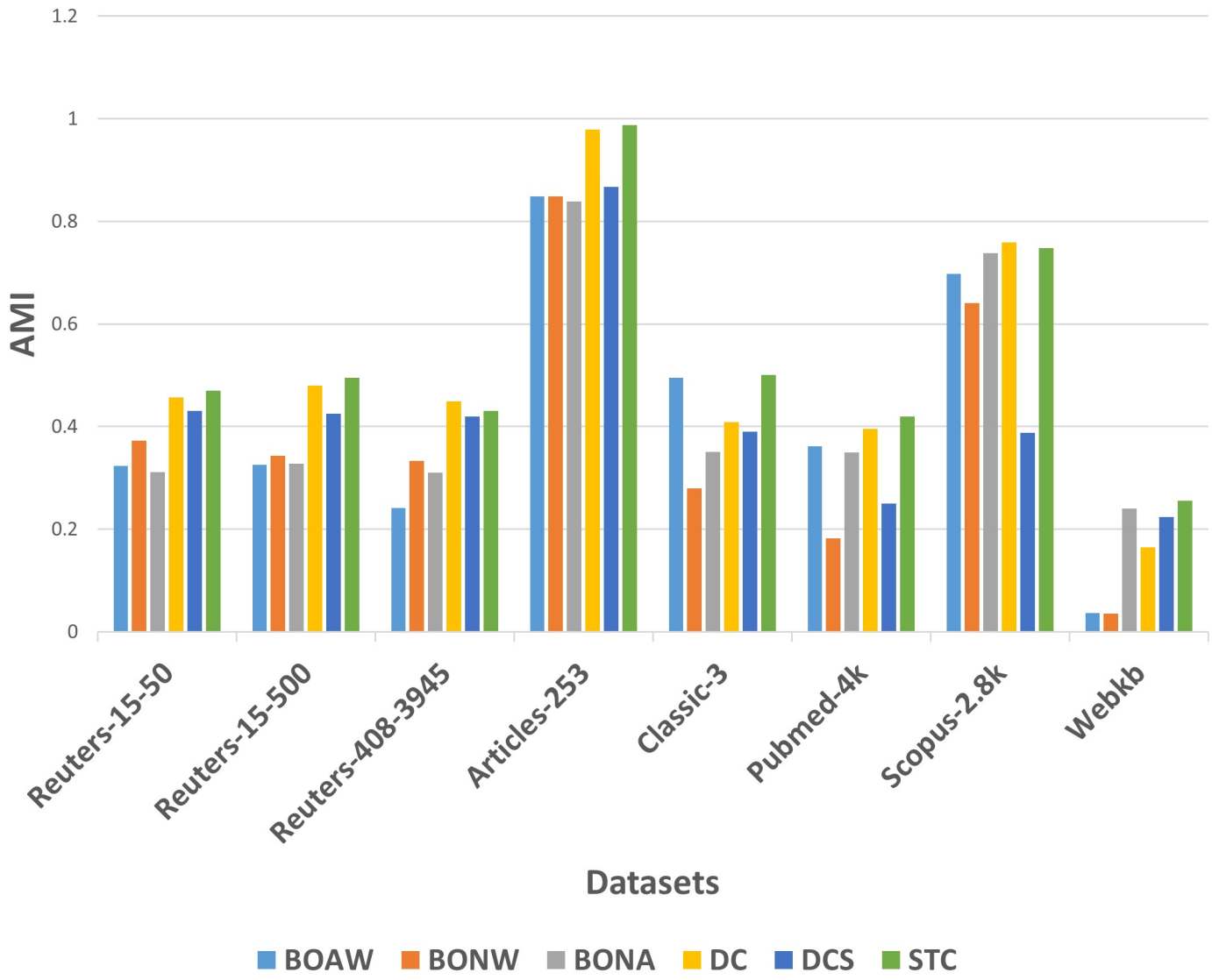

Figure 5.4: Results comparison of STC and other clustering techniques using AMI scores.

Table 5.4: Dimensionality used by different techniques.

S.no.	Datasets	BOAW	BONW	BONA	DC	DCS	**STC**
1	Reuters_15_50	4776	2330	3186	2330	1894	2563
2	Reuters_15_500	13963	7771	10222	7771	7541	8465
3	Reuters_408_3945	19271	11008	14173	11008	9996	10813
4	Articles-253	33083	22064	28450	22064	7882	8964
5	Classic-3	21741	12499	17674	12499	11195	13696
6	Pubmed_4K	22839	13974	18640	13974	11568	13874
7	Scopus_2.8k	29644	18657	25289	18657	12474	14651
8	Webkb	44667	28555	36355	28555	14280	17140
9	Newsgroups	13994	7996	8099	7996	5572	6543

Apart from the accuracy, analysis of the proposed STC technique, space analysis for dimensionality reduction, and execution time analysis is also presented below.

5.1.2 Dimensionality reduction

In addition to accuracy, space efficiency in terms of dimensionality reduction is also of much importance in the field of text mining. Using the proposed STC technique, a significant dimensionality reduction is achieved as well. Table 5.4 shows the dimensionality of different datasets used by different techniques including STC‘ to form clusters. The highest number of dimensions for each dataset is used by BOAW. DCS can reduce dimensionality greater than others. Following DCS, STC is also able to reduce the high dimensionality by a significant amount as shown for each dataset. However, for a small increase in the number of dimensions, STC achieves better performance accuracies than DCS. This demonstrates the capability of the proposed technique to deal with high-volume datasets.

5.1.3 Execution time

The execution time of a clustering algorithm depends directly on the dimensionality of the dataset. If the number of dimensions increases, correspondingly execution time also

Table 5.5: Execution time (in seconds) used by different techniques.

S.no.	Datasets	BOAW	BONW	BONA	DC	DCS	**STC**
1	Reuters_15_50	3.19	1.85	1.99	1.51	0.82	1.02
2	Reuters_15_500	53.86	26.77	28.47	27.43	25.66	29.677
3	Reuters_408_3945	73.00	42.49	44.90	69.45	28.58	32.57
4	Articles-253	15.81	12.93	12.50	11.24	0.84	1.04
5	Classic-3	90.192	35.02	47.95	36.78	31.53	32.55
6	Pubmed_4K	83.02	42.85	50.71	41.67	37.48	40.54
7	Scopus_2.8k	59.48	49.10	65.64	50.26	34.41	48.04
8	Webkb	80.30	42.61	48.17	42.70	35.38	52.28
9	Newsgroups	6.24	2.41	2.91	2.33	1.82	2.09

increases and vice-versa. In Table 5.5, the time taken (in seconds) in WEClustering to form clusters using its low dimensional concept-document matrix representation for each dataset is shown. Similarly, the time taken by other techniques to form clusters using the high-dimensional term-document matrix representation for each dataset is also shown. The highest execution time is taken by the BOAW technique followed by BONW, BONA, and DC. The lowest execution time is taken by DCS and then by STC. However, for this small difference, STC achieves better performance in the accuracy of clustering.

In the next section, a similar analysis of WEClustering is presented.

5.2 Result analysis of WEClustering

Analysis of the accuracy of WEClustering for different performance metrics is given below.

5.2.1 Accuracy

After running the proposed clustering technique i.e. WEClustering many important results are obtained. In this subsection, the results of WEClustering and its comparative analysis with other techniques are presented in detail with the help of suitable graphics

and tables. From now onwards, WEClustering-K and WEClustering-A denote that in the last phase of WEClustering, K-means and Agglomerative clustering are used respectively. Analysis for each of the performance metrics is as follows.

5.2.1.1 Silhouette coefficient

Table 5.6 shows the performance of WEClustering and other techniques based on Silhouette coefficient. This metric is most important of all the three performance metrics because, in a real scenario of clustering, true labels are not available to us. As already mentioned, the Silhouette coefficient determines the quality of clusters without requiring external labels. For all the seven datasets, the value of this metric goes much higher for WEClustering. Best values are indicated in bold. To get some more insights a column chart corresponding to the Table 5.6 is plotted in the form of Figure 5.5. It makes it very clear that WEClustering outperforms all other clustering techniques. On the other hand, in the proposed technique itself, WEClustering-K performs marginally better than WEClustering-A. Additional minute details of performances can be seen with the help of Table 5.9 and Figure 5.8. Table 5.9 highlights the minimum %age improvement made by the proposed clustering technique over other techniques for all datasets and performance measures. Figure 5.8 shows a column chart corresponding to this improvement in terms of silhouette coefficient. Minimum %age improvement is calculated in this paper as:

$$min\_\%age\_improvement = 100 \times \frac{Score_{proposed} - MaxScore_{others}}{MaxScore_{others}} \tag{5.1}$$

where

$Score_{proposed}$ = score achieved by the proposed technique

$MaxScore_{others}$ =maximum score of all other techniques

For example, the silhouette coefficient score achieved by the proposed technique for dataset Articles-253 is 0.458 and the maximum score among all scores of the other four techniques is 0.097. Hence, according to equation 5.1, minimum %age improvement becomes 372.164. A very large %age improvement can be seen for the Silhouette coefficient in all the datasets. This can be attributed to the fact that by its definition silhouette coefficient measures how well separated and dense the resulting clusters are formed. As the proposed technique can reduce the dimensionality of datasets from tens of thousands to less than 100, the resulting clusters formed in the last phase are quite well separated and dense.

Table 5.6: Silhouette coefficient values of WEClustering and other techniques on different datasets.

S.no	Datasets	WEClustering-K	WEClustering-A	K-means	Agglomerative	HDBSCAN	Genie
1.	Articles-253	**0.458**	0.454	0.097	0.097	0.087	0.077
2.	Classic4	**0.264**	0.258	0.013	0.011	0.010	0.014
3.	Classic4-long	**0.259**	0.253	0.010	0.010	0.008	0.011
4.	Scopus	**0.314**	0.298	0.017	0.015	0.005	0.016
5.	Scopus-long	**0.240**	0.238	0.012	0.012	-0.016	0.003
6.	20NG	**0.130**	0.110	0.002	0.009	0.002	0.007
7.	20NG-long	**0.086**	0.023	0.007	0.004	0.001	0.007

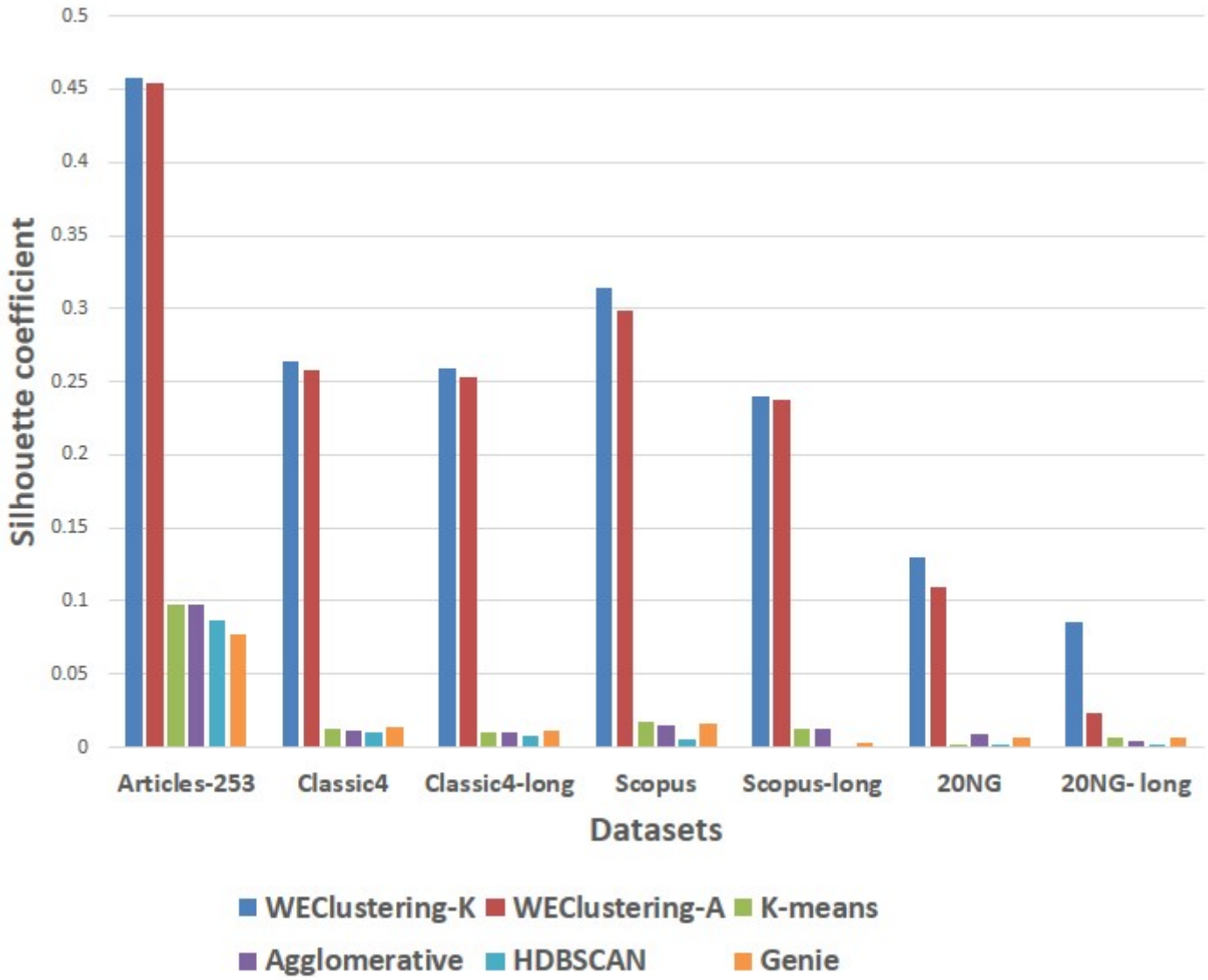

Figure 5.5: Column chart indicating the values of Silhouette coefficient for WEClustering and other clustering techniques.

Table 5.7: Purity values of WEClustering and other techniques on different datasets.

S.no	Datasets	WEClustering-K	WEClustering-A	K-means	Agglomerative	HDBSCAN	Genie
1.	Articles-253	0.988	**1.0**	**1.0**	0.992	0.893	0.798
2.	Classic4	0.970	**0.981**	0.887	0.961	0.830	0.971
3.	Classic4-long	0.961	**0.974**	0.845	0.966	0.802	0.947
4.	Scopus	**0.956**	00.936	0.916	0.866	0.378	0.838
5.	Scopus-long	0.765	**0.775**	0.750	0.729	0.439	0.576
6.	20NG	**0.637**	0.600	0.607	0.506	0.339	0.341
7.	20NG-long	**0.409**	0.277	0.365	0.363	0.120	0.128

5.2.1.2 Purity

For the case when external labels are available, purity can be used to measure the clustering results. Table 5.7 highlights the purity values achieved by all the clusterings for all the datasets. Again, it is very clear that the proposed clustering technique outperforms all other techniques. A visualization chart corresponding to these values is provided in Figure 5.6. In column 4 of Table 5.9, values of %age improvement gained by the proposed technique over others are provided. This improvement is calculated in the same way as for silhouette coefficient i.e. using equation 5.1. Figure 5.9 shows a column chart corresponding to these improvement values. It can be inferred from the table that for each dataset, there is a significant improvement of performance. For the Articles-253 dataset, the proposed technique and K-means achieved the maximum Purity score i.e. 1.0, therefore improvement comes out to be zero. It should also be noticed from the figure that as the size of the dataset grows more improvement performance is taking place. This trend proves the efficiency of the proposed technique for large datasets.

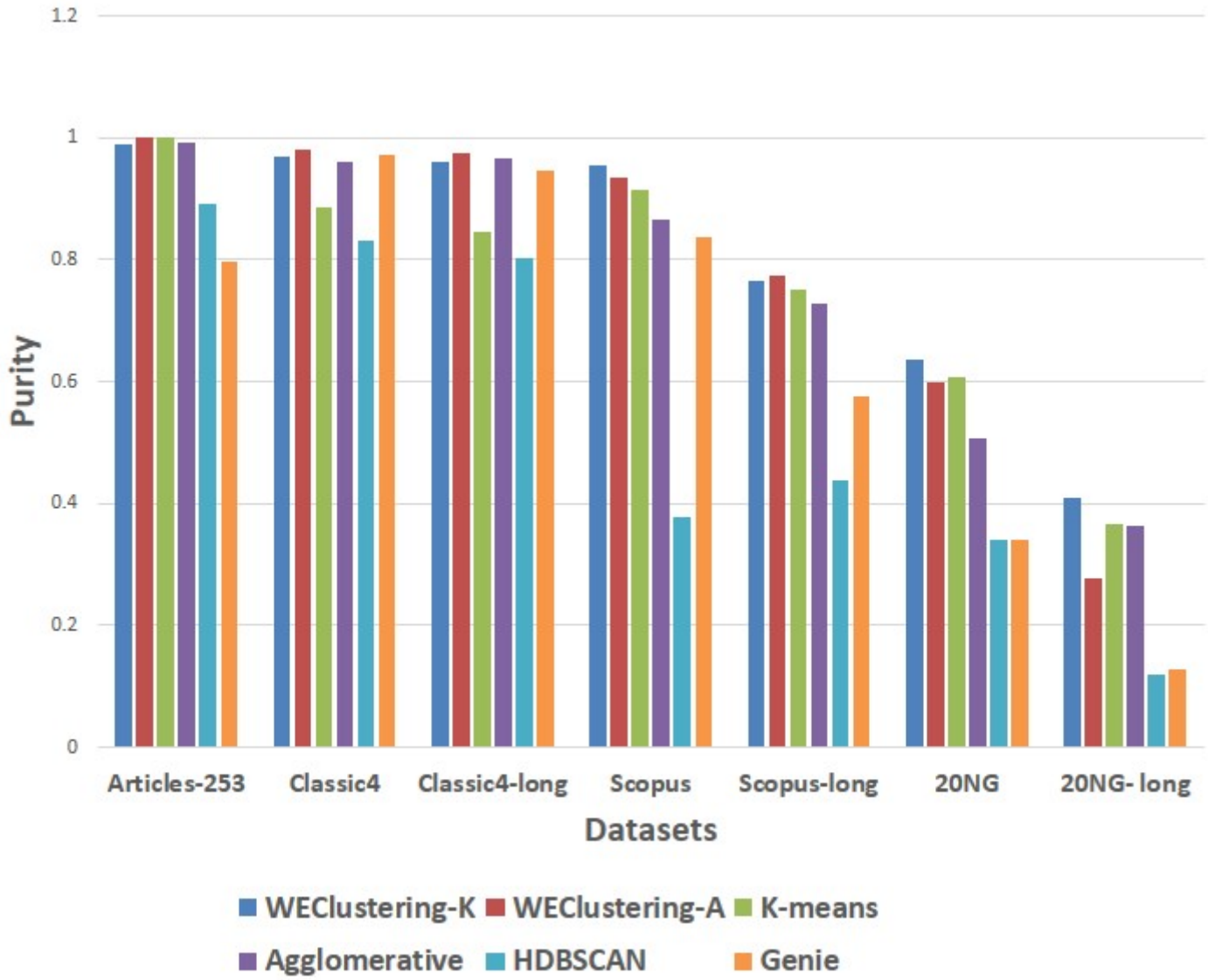

Figure 5.6: Column chart indicating the values of purity for WEClustering and clustering techniques.

5.2.1.3 ARI

To more verify the results achieved as measured by the above two performance metrics, ARI is also used. Like purity, it measures performance for available ground truth labels. Individual values of ARI for clustering achieved by different clustering algorithms for all datasets are shown in Table 5.8. It can be seen that for Articles-253, WEClustering and K-means both achieved the maximum ARI score of 1.0. It is probably because it is a small dataset containing 253 articles, so it is easy to find clusters in it. For all other datasets, WEClustering achieves greater ARI values. The best values are shown in bold. Figure 5.7 depicts the same trend with the help of a column chart. Additionally, minimum %age performance improvement in terms of ARI is shown with the help of Figure 5.10. Again, it can be seen that there is a significant performance improvement made by the proposed clustering technique. As one goes from smaller to larger datasets, performance improvement also increases.

The reason behind these results can be attributed to the fact that word embeddings achieved from the BERT model capture the semantics of the word and its context better. Other clustering techniques which are just based on a scoring mechanism like TF-IDF cannot capture the meaning of a word with respect to its context. The proposed technique combines the advantages of statistical scoring mechanisms like TF-IDF as well as the semantics of the word. Additionally, the clustering of word embeddings using Minibatch K-means combines the words with similar context into a single group or a cluster. Hence, it reduces the dimensionality of the problem drastically i.e. from tens of thousands to less than a hundred. This enables clustering algorithms like K-means and Agglomerative clustering to give better clusters.

Table 5.8: ARI values of WEClustering and other techniques on different datasets.

S.no	Datasets	WEClustering-K	WEClustering-A	K-means	Agglomerative	HDBSCAN	Genie
1.	Articles-253	0.970	**1.0**	**1.0**	0.978	0.734	0.687
2.	Classic4	0.912	**0.945**	0.737	0.886	0.456	0.916
3.	Classic4-long	0.888	**0.922**	0.641	0.902	0.477	0.859
4.	Scopus	**0.893**	0.849	0.807	0.702	0.108	0.680
5.	Scopus-long	0.600	**0.611**	0.518	0.524	0.010	0.279
6.	20NG	**0.344**	0.264	0.272	0.149	0.003	0.001
7.	20NG-long	**0.165**	0.131	0.079	0.085	0.001	0.000

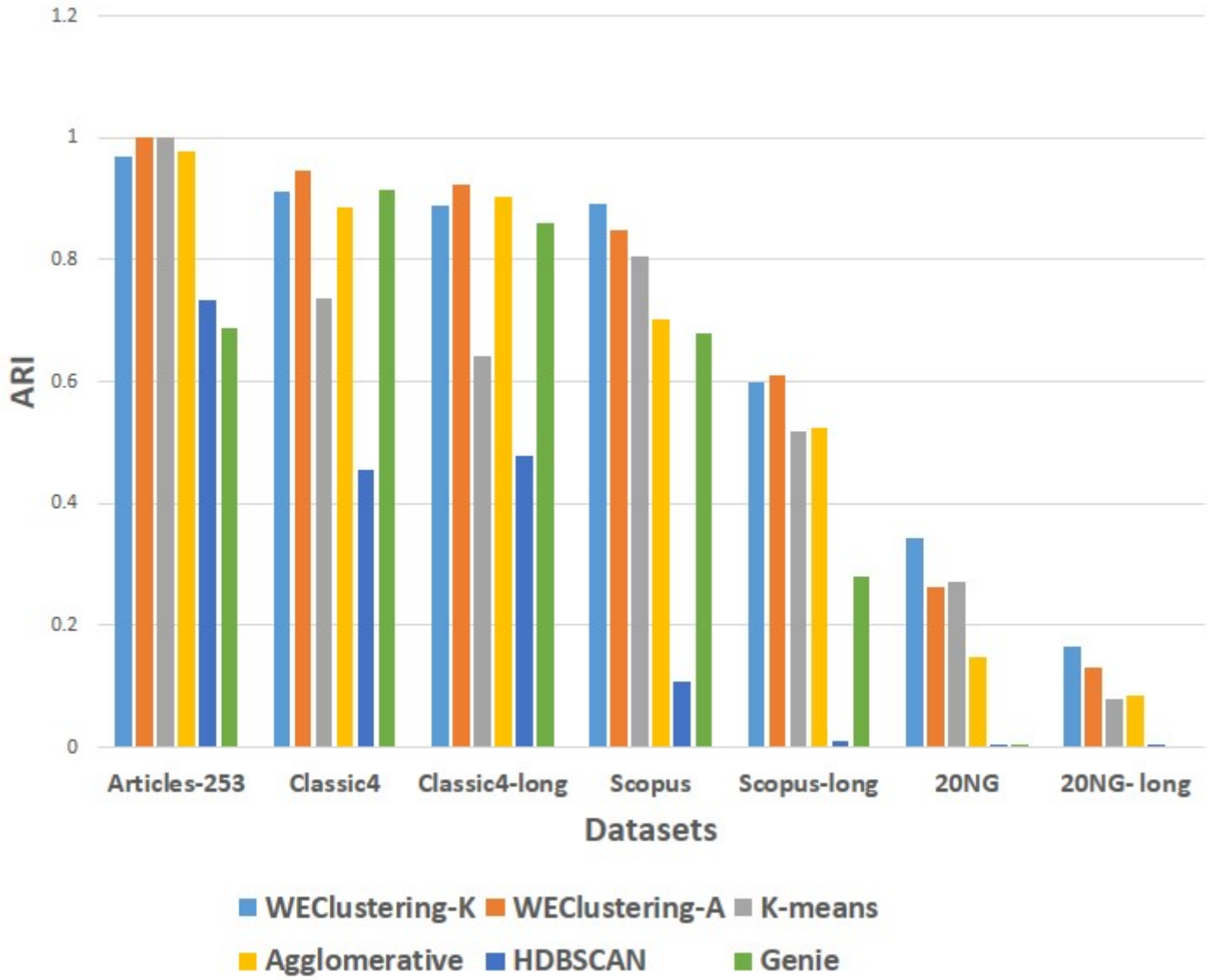

Figure 5.7: Column chart indicating the values of ARI for clustering techniques.

Table 5.9: Minimum %age improvement in performance using WEClustering.

S.no	Datasets	Minimum %age improvement in Silhouette coefficient	Minimum %age improvement in Purity	Minimum %age improvement in ARI
1.	Articles-253	372.164	0	0
2.	Classic4	1785.714	1.029	3.165
3.	Classic4-long	2254.545	0.828	2.217
4.	Scopus	1747.058	4.366	10.656
5.	Scopus-long	1900	3.333	16.603
6.	20NG	1344.444	4.942	26.470
7.	20NG-long	1128.571	12.054	94.117

5.2.2 Dimensionality reduction

In WEClustering, the idea of forming groups of vocabulary words, (which are called concepts in this research work) reduces the dimensionality of the CD data matrix significantly. Essentially, the values of k_{voc} parameter shown in Table 4.2 of chapter 4 are the actual number of dimensions used for each dataset. It is also shown again in Table 5.10. Hence, instead of thousands of words that were used to form term-document matrix by other clustering techniques, WEClustering combines them to form only less than a hundred concepts to form the CD matrix. This also results in higher clustering accuracy.

5.2.3 Execution time

The time taken to form clusters from a data matrix directly depends on the dimensionality of the data matrix. As WEClustering reduces the high number of dimensions to a low number of dimensions, the time taken to form clusters is significantly lower than other techniques. Table 5.11 shows time taken (in seconds) to form clusters by WEClustering and other techniques. For all seven datasets, the time taken to form document clusters by WEClustering is very less than all other techniques i.e. K-means, Agglomerative, HDBSCAN, and Genie.

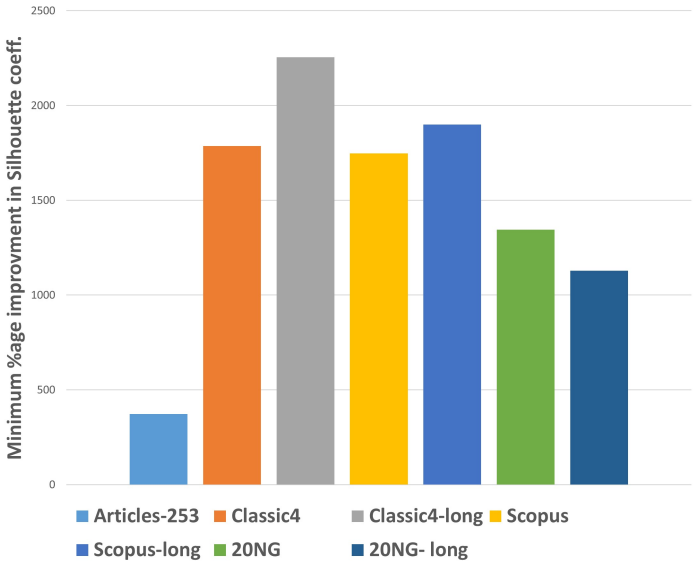

Figure 5.8: Column chart indicating the improvement of performance using WEClustering w.r.t Silhouette coefficient.

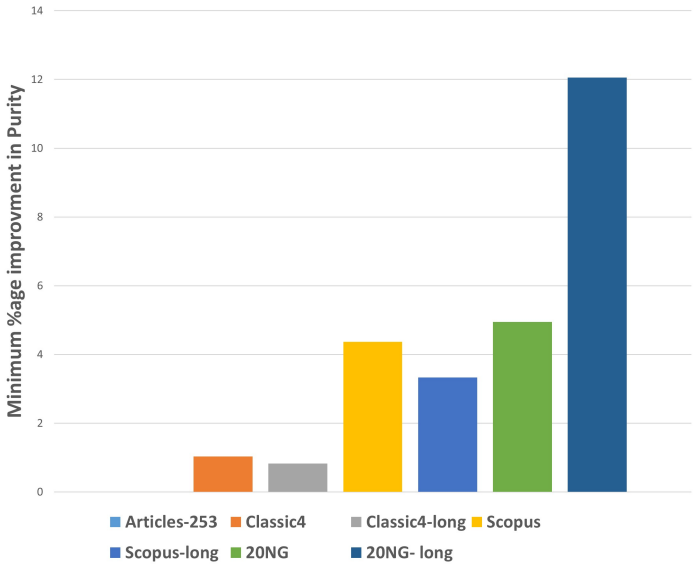

Figure 5.9: Column chart indicating the improvement of performance using WEClustering w.r.t purity.

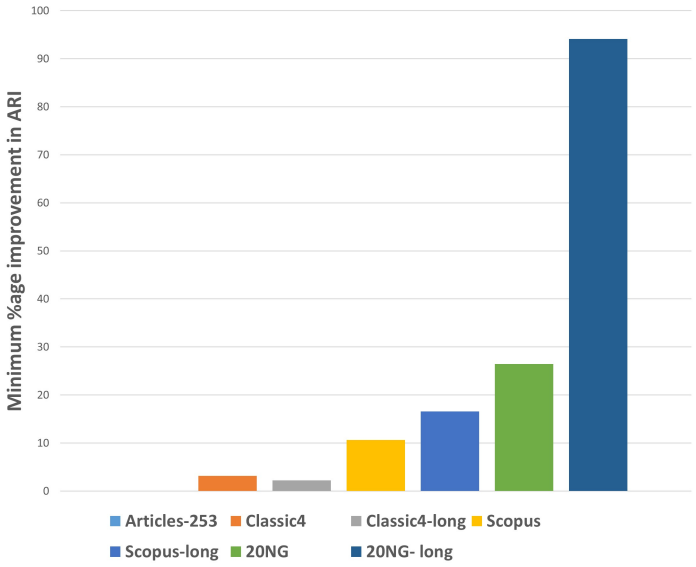

Figure 5.10: Column chart indicating the improvement of performance using WEClustering w.r.t ARI.

Table 5.10: Values of parameters k_{voc} showing reduced dimensionality obtained by WEClustering and the actual dimensionality.

S.no	**Dataset**	**k_{voc}/reduced dimensionality**	**Actual dimensionality**
1.	Articles-253	35	44555
2.	Scopus	35	12680
3.	20NG	25	21916
4.	Classic4	35	18330
5.	Scopus-long	35	82970
6.	Classic4-long	35	80970
7.	20NG-long	75	85513

Table 5.11: Execution time (in seconds) of WEClustering and other techniques on different datasets.

S.no	Datasets	WEClustering-K	WEClustering-A	K-means	Agglomerative	HDBSCAN	Genie
1.	Articles-253	0.082	0.009	10.692	0.275	2.387	0.353
2.	Classic4	0.107	0.022	13.018	1.142	5.358	1.462
3.	Classic4-long	0.266	0.638	73.029	70.011	270.604	71.791
4.	Scopus	0.152	0.010	11.053	0.464	2.606	0.558
5.	Scopus-long	0.060	0.258	2.120	1.140	203.293	44.779
6.	20NG	0.089	0.017	9.567	1.147	6.823	1.161
7.	20NG-long	2.274	5.223	129.195	1017.126	4886.316	803.929

5.3 Comparison between STC and WEClustering

In this section, a comparison of performance between the two proposed clustering techniques which are STC and WEClustering is presented. This comparison is based on accuracy, execution time, scalability, and language dependency of the two techniques. Datasets used for this comparison are the same as listed in Table 4.1 of chapter 4.

5.3.1 Accuracy

The clustering accuracy of both techniques is determined using three performance metrics which are the Silhouette coefficient, Purity, and ARI. These are explained in sequence as follows.

1. Silhouette coefficient

   In almost all the datasets, WEClustering achieved a higher value of silhouette coefficient value as shown in Table 5.12. In the dataset Classic4, STC achieved a higher value (0.400) than that of WEClustering (0.264). This can be attributed to the fact that WEClustering lowers down the dimensionality of each dataset very well. Hence, values of silhouette coefficient which measures the compactness of clusters are higher for WEClustering. Line chart shown in Figure 5.11 gives a visual comparison of these values for the two techniques.

Table 5.12: Silhouette coefficient values of STC and WEClustering on different datasets.

S.no	Datasets	STC	WEClustering
1.	Articles-253	0.247	0.458
2.	Classic4	0.400	0.264
3.	Classic4-long	0.026	0.259
4.	Scopus	0.056	0.314
5.	Scopus-long	0.047	0.240
6.	20NG	0.032	0.130
7.	20NG-long	0.025	0.086

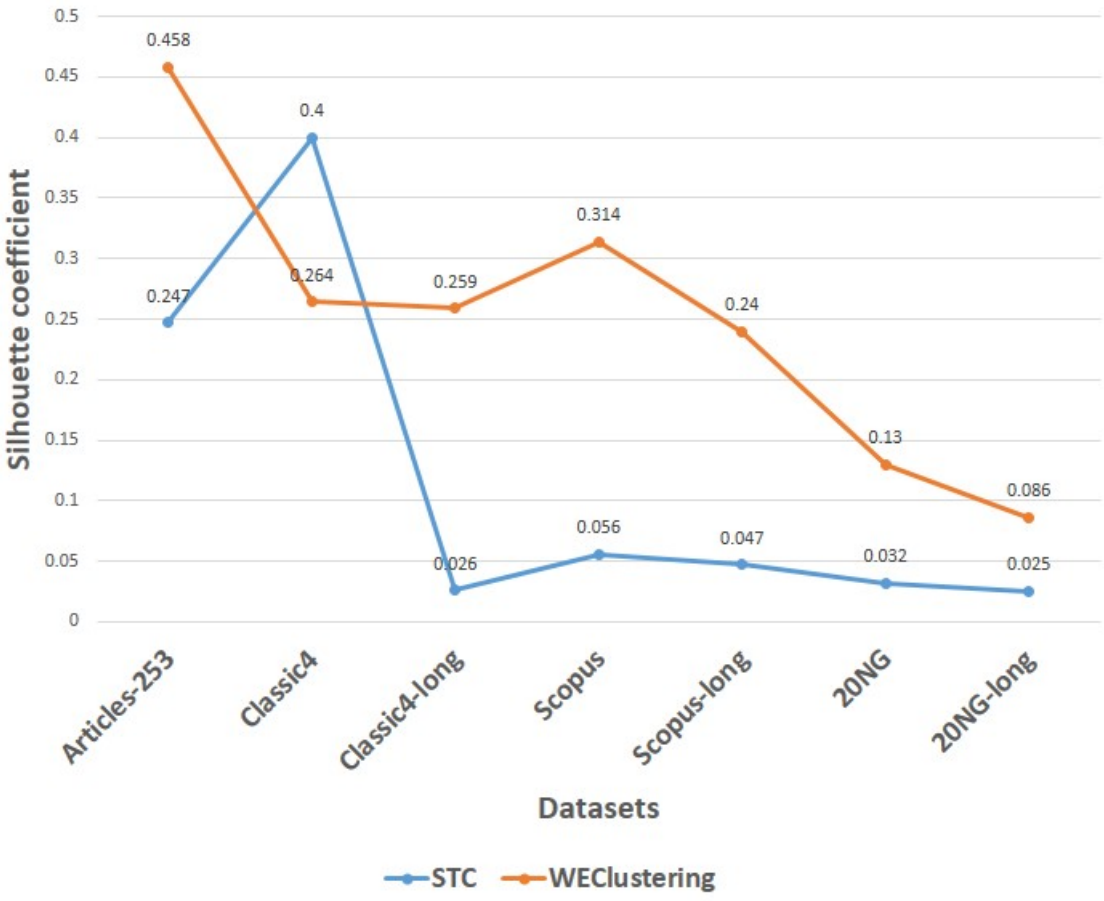

Figure 5.11: Results comparison of STC and WEClustering using Silhouette coefficient scores.

Table 5.13: Purity values of STC and WEClustering on different datasets.

S.no	Datasets	STC	WEClustering
1.	Articles-253	0.996	0.988
2.	Classic4	0.785	0.970
3.	Classic4-long	0.746	0.961
4.	Scopus	0.750	0.956
5.	Scopus-long	0.883	0.765
6.	20NG	0.345	0.637
7.	20NG-long	0.221	0.409

2. Purity

   The trend in values of Purity is somewhat different than that of the Silhouette coefficient. Purity values as listed in Table 5.13 show that there are two datasets in which STC achieved a higher value of purity than WEClustering. These two datasets are Articles-253 and Scopus-long. For the other five datasets, WEClustering performs better than STC. Overall, WEClustering achieved a higher accuracy. This can be attributed to the fact that using word embeddings, WEClustering captures better semantic relationships among different words than STC which captures the semantic relationships based on WordNet which has some limitations. A visual comparison of Purity values is shown in Figure 5.12 in which the orange line corresponding to WEClustering remains above the blue line of STC except for two datasets.

3. ARI

   The pattern of ARI values is nearly the same as that of Purity values. Again for the dataset Articles-253, STC achieved a higher value than WEClustering. However, for all the other six datasets, WEClustering shows better performance than STC. The values are shown in Table 5.14. Articles-253 is relatively a smaller dataset than others, hence the performance of both STC and WEClustering are nearly the same (STC being a little higher). For other datasets which are larger, the use of word embeddings to capture the semantic relationships among words shows its positive effect in clustering results. Figure 5.13 shows the linechart of ARI values for both clustering algorithms.

Figure 5.12: Results comparison of STC and WEClustering using Purity scores.

Table 5.14: ARI values of STC and WEClustering on different datasets.

S.no	Datasets	STC	WEClustering
1.	Articles-253	0.989	0.970
2.	Classic4	0.458	0.912
3.	Classic4-long	0.467	0.888
4.	Scopus	0.579	0.893
5.	Scopus-long	0.258	0.600
6.	20NG	0.180	0.344
7.	20NG-long	0.098	0.165

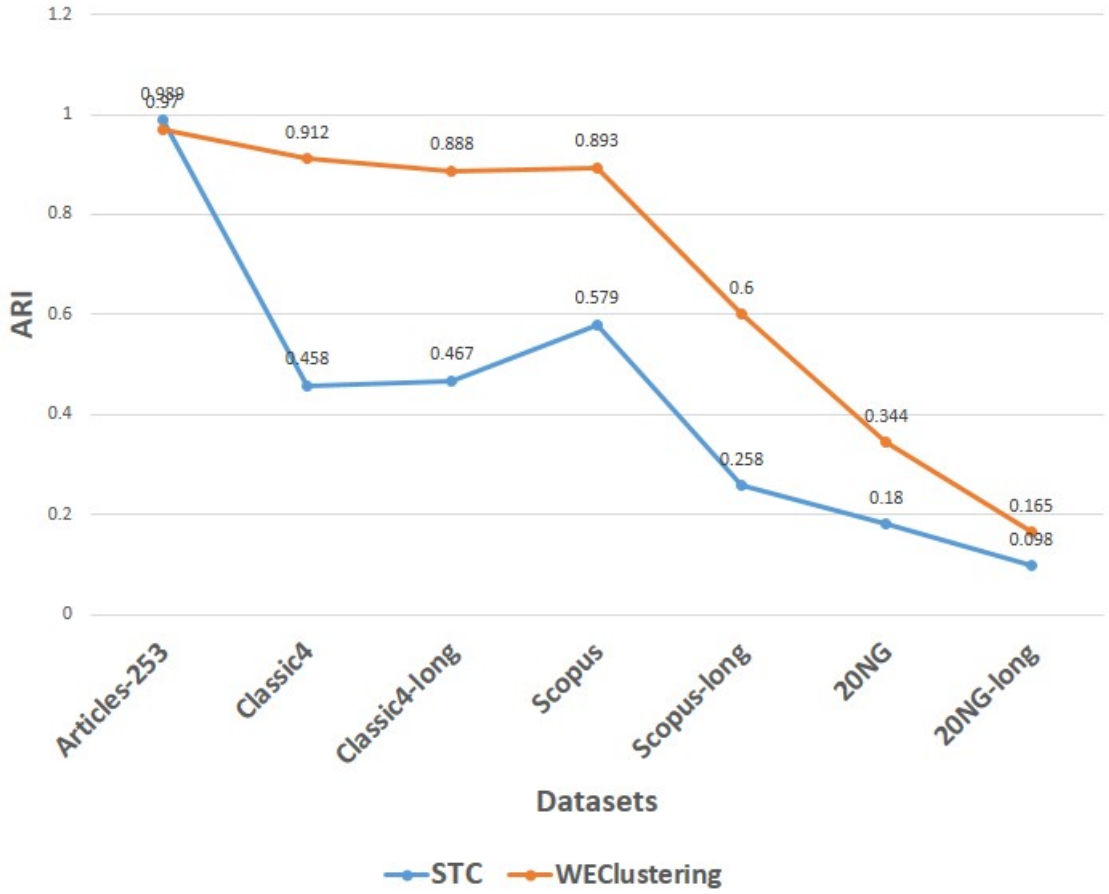

Figure 5.13: Results comparison of STC and WEClustering using ARI scores.

Table 5.15: Reduced dimensionality obtained by STC and WEClustering on different datasets.

S.no	Datasets	STC	WEClustering
1.	Articles-253	8964	35
2.	Classic4	13696	35
3.	Classic4-long	6003	35
4.	Scopus	6017	35
5.	Scopus-long	14651	35
6.	20NG	6543	25
7.	20NG-long	25153	75

Table 5.16: Execution time (in seconds) of STC and WEClustering on different datasets.

S.no	Datasets	STC	WEClustering
1.	Articles-253	1.168	0.082
2.	Classic4	32.55	0.109
3.	Classic4-long	3.332	0.266
4.	Scopus	2.516	0.152
5.	Scopus-long	48.04	0.328
6.	20NG	2.09	0.089
7.	20NG-long	35.38	0.172

Both the proposed clustering techniques achieved a significant reduction in dimensionality to perform clustering on large text datasets. Both techniques have the same underlying idea that combining the semantically related words to form groups can achieve dimensionality reduction. STC attempts to group the related words using a lexical database known as WordNet which provides typical semantic relations such as synonyms, hypernyms, hyponyms, and meronyms. On the other hand, WEClustering attempts to group semantically related words using a recently proposed deep learning architecture called BERT. BERT provides a simple numerical representation of each word known as a word embedding. Hence, comparing and grouping words based on a numerical vector is more effective than grouping words using lexical chains. That is why WEClustering provides a much better dimensionality reduction than STC.

As shown in Table 5.15, WEClustering reduces the number of dimensions to less than a hundred by forming large groups of words known as concepts in this research work. This is made possible only due to a numerical representation of words which makes the semantic comparison of words much easier. In WEClustering, Minibatch K-means is used to perform grouping (or clustering) of words falling in the complete vocabulary of a corpus using these numerical word vectors. This was not possible in STC in which simple string-based comparison was used to form groups of words which are called lexical chains.

5.3.2 Execution time

Table 5.16 shows execution time taken to obtain clusters on the final data matrix representation of documents in each dataset. This data matrix is termed as a chain-document matrix in the case of STC and concept-document matrix in the case of WEClustering. Due to a very less number of features formed in the concept-document matrix than the chain-document matrix, the execution time of obtaining clusters of documents in each dataset is very less in the case of WEClustering. Hence, it can be said that WEClustering is more time-efficient than STC.

5.3.3 Scalability to big datasets

Accuracy of clustering in big textual datasets mainly depends upon two factors:

1. How effectively the semantics of text have been incorporated to determine the category of the document.
2. How less the number of features have been used for clustering.

Both the proposed clustering techniques achieved a significant reduction in dimensionality to perform clustering on large text datasets. Both techniques have the same underlying idea that combining the semantically related words to form groups can achieve dimensionality reduction. STC attempts to group the related words using a lexical database known as WordNet which provides typical semantic relations such as synonyms, hypernyms, hyponyms, and meronyms. On the other hand, WEClustering attempts to group semantically related words using a recently proposed deep learning architecture called BERT. BERT provides a simple numerical representation of each word known as a word embedding. Hence, comparing and grouping words based on a numerical vector is more effective than grouping words using lexical chains. That is why WEClustering provides better dimensionality reduction than STC.

As shown in Table 5.15, WEClustering reduces the number of dimensions to less than a hundred by forming large groups of words known as concepts in this research work. This is made possible only due to a numerical representation of words which makes the semantic comparison of words much easier. In WEClustering, Minibatch K-means is used to perform grouping (or clustering) of words falling in the complete vocabulary of a corpus using these numerical word vectors. Hence, the proposed techniques are more scalable to big datasets in comparison to other techniques to which they have been compared. Among the two proposed techniques, WEClustering is more scalable to big datasets.

5.3.4 Language dependency

STC and WEClustering both perform a semantic comparison of words for each dataset. STC uses WordNet and WEClustering uses word embeddings of BERT to perform this comparison. Unfortunately, a lexical database like WordNet is not available for every language. WordNet is the database corresponding to the English language only. Hence, STC can perform the clustering of text datasets that consists of English language documents alone. However, BERT is a general deep learning architecture for language. It takes a large corpus of documents of a language and produces embeddings for all the words contained in the corpus. Hence, the corpus may belong to any language rather than a single language. This makes WEClustering a language-independent technique that can be used to cluster text documents of any language.

In the next chapter, various conclusions drawn out of the research work presented in this thesis are provided. Additionally, the scope of future work which can be performed for further improvement is given.

Chapter 6

Conclusions and Future Scope

This chapter presents the concluding remarks on the whole research work presented in this thesis. It highlights various contributions and outcomes of the work that marks its validity, applicability, and completeness with respect to the defined objectives. It also highlights the novelty, efficiency, and scalability of each of the two proposed text clustering techniques "STC" and "WEClustering" with respect to big datasets. Finally, the future scope of this research work is presented.

6.1 Conclusions

This thesis entitled "Efficient Text Clustering Techniques for Big Datasets" focuses mainly on clustering the big textual datasets having high dimensionality. The major outcomes of the research carried out and presented in this thesis are as follows.
In chapter 2 entitled "Literature Review", an extensive literature review of methods and techniques used for text representation and text clustering is presented. A comparative analysis of various existing text clustering techniques with respect to factors more concerned with big datasets is also presented. This review along with the highlighted research gaps can help the related research community in getting a quick review and analysis of the presented techniques. This, in turn, may help to design novel text clustering techniques which can handle the challenges encountered in big datasets.
In chapter 3 entitled "Proposed Clustering Technique: Stamantic Clustering (STC)", a novel text clustering technique is proposed that combines the statistical features with semantic features using WordNet and lexical chains. This is done in order to include semantics as well as reduce the dimensionality of datasets significantly. This reduction in dimensionality makes the proposed STC clustering technique scalable to big datasets. This idea of using lexical chains can motivate the researchers to use the same for other text mining tasks such as text summarization and text classification.
In chapter 4 entitled "Proposed Clustering Technique: WEClustering", another novel text clustering technique is proposed based on the state-of-the-art deep learning architecture BERT. This proposed technique presents a unique idea of the incorporation of word

embeddings into text clustering. This helps in solving the ambiguity problem and more importantly the high dimensionality problem. The proposed technique WEClustering is much more scalable to big datasets due to its ability to reduce the high number of dimensions to a much lower value. The core idea used in WEClustering can be used by other researchers to apply it in other text mining tasks as well.
In chapter 5 "Experimental Results and Validation of Proposed Clustering Techniques", the two proposed text clustering techniques are validated on several big datasets having as high dimensionality as tens of thousands. The results presented for both the techniques prove the efficiency of the proposed techniques in terms of accuracy, significant dimensionality reduction, and thus more scalability to big datasets as compared with the existing text clustering techniques.

6.2 Future Scope

1. **Usage of a different procedure for extracting semantic features**
   In the proposed clustering techniques STC and WEClustering, lexical chains and word embeddings are used respectively for semantics extraction. In the future, some change or a completely different mechanism for semantics extraction can be explored for more efficiency.

2. **Implementation on other language datasets**
   The proposed technique WEClustering can be implemented on textual datasets of some different languages than English.

3. **Scalability to much bigger dimensions**
   As the size of datasets keeps on increasing day by day, the proposed text clustering techniques can be implemented on datasets having millions of documents.

4. **Usage in other text mining tasks**
   Other tasks which use text clustering as an intermediate step such as multi-document summarization can be designed with the help of proposed text clustering techniques.

5. **Fine-tuning of BERT for specific domain**
   The BERT model used in WEClustering can be fine-tuned for more suitability to textual datasets of specific domains such as medicine, computer science dataset, and so on.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] "Carrotsearch." `https://search.carrotsearch.com/`. Accessed: 2021-17-01.

[3] "Yippy." `yippy.com`. Accessed: 2021-17-01.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[5] J. Alammar, "The illustrated bert, elmo, and co.." `http://jalammar.github.io/illustrated-bert/`. Accessed: 2021-01-25.

[6] A. K. Jain and R. C. Dubes, "Algorithms for clustering data," *Englewood Cliffs: Prentice Hall, 1988*, 1988.

[7] C. C. Aggarwal and C. K. Reddy, "Data clustering," *Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra*, 2014.

[8] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[10] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[11] B. McCune, J. B. Grace, and D. L. Urban, *Analysis of ecological communities*, vol. 28. MjM software design Gleneden Beach, OR, 2002.

[12] A. R. Hanna, C. Rao, and T. Athanasiou, "Graphs in statistical analysis," in *Key Topics in Surgical Research and Methodology*, pp. 441–475, Springer, 2010.

[13] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[14] S. Glen, "Kullbackleibler kl divergence." `https://www.statisticshowto.datasciencecentral.com/kl-divergence`. Accessed: 2018-04-28.

[15] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pp. 49–56, 2008.

[16] T. W. Schoenharl and G. Madey, "Evaluation of measurement techniques for the

validation of agent-based simulations against streaming data," in *International Conference on Computational Science*, pp. 6–15, Springer, 2008.

[17] S. Glen, "Bray curtis dissimilarity." `http://www.statisticshowto.com/bray-curtis-dissimilarity/`. Accessed: 2018-04-28.

[18] "Jaccard index." `https://en.wikipedia.org/wiki/Jaccard_index`. Accessed: 2018-04-28.

[19] Y.-S. Lin, J.-Y. Jiang, and S.-J. Lee, "A similarity measure for text classification and clustering," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 7, pp. 1575–1590, 2014.

[20] A. S. Shirkhorshidi, S. Aghabozorgi, and T. Y. Wah, "A comparison study on similarity and dissimilarity measures in clustering continuous data," *PloS one*, vol. 10, no. 12, p. e0144059, 2015.

[21] T. Basu and C. Murthy, "A similarity assessment technique for effective grouping of documents," *Information Sciences*, vol. 311, pp. 149–162, 2015.

[22] R. Saraçoğlu, K. Tütüncü, and N. Allahverdi, "A fuzzy clustering approach for finding similar documents using a novel similarity measure," *Expert systems with applications*, vol. 33, no. 3, pp. 600–605, 2007.

[23] A. Leoncini, F. Sangiacomo, C. Peretti, S. Argentesi, R. Zunino, and E. Cambria, "Semantic models for style-based text clustering," in *2011 IEEE Fifth International Conference on Semantic Computing*, pp. 75–82, IEEE, 2011.

[24] P. Vossen, "Eurowordnet general document version 3," *University of Amsterdam*, 2002.

[25] S. Kannan, S. Ramathilagam, R. Devi, and E. Hines, "Strong fuzzy c-means in medical image data analysis," *Journal of Systems and Software*, vol. 85, no. 11, pp. 2425–2438, 2012.

[26] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[27] J.-O. Palacio-Niño and F. Berzal, "Evaluation metrics for unsupervised learning algorithms," *arXiv preprint arXiv:1905.05667*, 2019.

[28] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *2010 IEEE International Conference on Data Mining*, pp. 911–916, IEEE, 2010.

[29] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, 2020.

[30] J. M. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," in *International conference on artificial*

neural networks, pp. 175–184, Springer, 2009.

[31] C. D. Manning, H. Schütze, and P. Raghavan, *Introduction to information retrieval*. Cambridge university press, 2008.

[32] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[33] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[34] S. Khanmohammadi, N. Adibeig, and S. Shanehbandy, "An improved overlapping k-means clustering method for medical applications," *Expert Systems with Applications*, vol. 67, pp. 12–18, 2017.

[35] W. Vogt and D. Nagel, "Cluster analysis in diagnosis," *Clinical Chemistry*, vol. 38, no. 2, pp. 182–198, 1992.

[36] R. Paul and A. S. M. L. Hoque, "Clustering medical data to predict the likelihood of diseases," in *2010 fifth international conference on digital information management (ICDIM)*, pp. 44–49, IEEE, 2010.

[37] F. A. da Veiga, "Structure discovery in medical databases: a conceptual clustering approach," *Artificial intelligence in medicine*, vol. 8, no. 5, pp. 473–491, 1996.

[38] Z. Ma, J. M. R. Tavares, and R. N. Jorge, "A review on the current segmentation algorithms for medical images," in *Proceedings of the 1st International Conference on Imaging Theory and Applications (IMAGAPP)*, 2009.

[39] D. Naik and P. Shah, "A review on image segmentation clustering algorithms," *Int J Comput Sci Inform Technol*, vol. 5, no. 3, pp. 3289–93, 2014.

[40] D. Dilts, J. Khamalah, and A. Plotkin, "Using cluster analysis for medical resource decision making," *Medical Decision Making*, vol. 15, no. 4, pp. 333–346, 1995.

[41] R. Nugent and M. Meila, "An overview of clustering applied to molecular biology," in *Statistical methods in molecular biology*, pp. 369–404, Springer, 2010.

[42] C. Wiwie, J. Baumbach, and R. Röttger, "Comparing the performance of biomedical clustering methods," *Nature methods*, vol. 12, no. 11, p. 1033, 2015.

[43] B. B. Nair, P. S. Kumar, N. Sakthivel, and U. Vipin, "Clustering stock price time series data to generate stock trading recommendations: An empirical study," *Expert Systems with Applications*, vol. 70, pp. 20–36, 2017.

[44] T. Hillerman, J. C. F. Souza, A. C. B. Reis, and R. N. Carvalho, "Applying clustering and ahp methods for evaluating suspect healthcare claims," *Journal of Computational Science*, vol. 19, pp. 97–111, 2017.

[45] C. Morton, J. Anable, and J. D. Nelson, "Consumer structure in the emerging market for electric vehicles: Identifying market segments using cluster analysis,"

International Journal of Sustainable Transportation, vol. 11, no. 6, pp. 443–459, 2017.

[46] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita, "Topical clustering of search results," in *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 223–232, 2012.

[47] F. Jing, C. Wang, Y. Yao, K. Deng, L. Zhang, and W.-Y. Ma, "Igroup: web image search results clustering," in *Proceedings of the 14th ACM international conference on Multimedia*, pp. 377–384, 2006.

[48] R. Johnson, A. Watkinson, and M. Mabe, "The stm report," *An overview of scientific and scholarly publishing. 5th edition October*, 2018.

[49] C. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Mining text data*, pp. 77–128, Springer, 2012.

[50] "Carrot2." `https://carrot2.github.io/release/latest`. Accessed: 2021-23-01.

[51] D. R. Radev, E. Hovy, and K. McKeown, "Introduction to the special issue on summarization," *Computational linguistics*, vol. 28, no. 4, pp. 399–408, 2002.

[52] D. R. Radev, H. Jing, M. Styś, and D. Tam, "Centroid-based summarization of multiple documents," *Information Processing & Management*, vol. 40, no. 6, pp. 919–938, 2004.

[53] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams, "Fast generation of result snippets in web search," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 127–134, 2007.

[54] H. Saggion and T. Poibeau, "Automatic text summarization: Past, present and future," in *Multi-source, multilingual information extraction and summarization*, pp. 3–21, Springer, 2013.

[55] H.-j. Zeng, X. Wang, Z. Chen, B. Zhang, and W.-y. Ma, "Clustering based text classification," Apr. 29 2008. US Patent 7,366,705.

[56] H.-J. Zeng, X.-H. Wang, Z. Chen, H. Lu, and W.-Y. Ma, "Cbc: Clustering based text classification requiring minimal labeled data," in *Third IEEE International Conference on Data Mining*, pp. 443–450, IEEE, 2003.

[57] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "On feature distributional clustering for text categorization," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 146–153, 2001.

[58] Y. Xia, N. Tang, A. Hussain, and E. Cambria, "Discriminative bi-term topic model for headline-based social news clustering," in *The Twenty-Eighth International Flairs Conference*, pp. 311–316, 2015.

[59] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 1445–1456, ACM, 2013.

[60] T. G. Jan, "Clustering of tweets: A novel approach to label the unlabelled tweets," in *Proceedings of ICRIC 2019*, pp. 671–685, Springer, 2020.

[61] X. Hong, Z. Yu, M. Tang, and Y. Xian, "Cross-lingual event-centered news clustering based on elements semantic correlations of different news," *Multimedia Tools and Applications*, vol. 76, no. 23, pp. 25129–25143, 2017.

[62] A. Montoyo, P. MartíNez-Barco, and A. Balahur, "Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments," *Decis. Support Syst*, vol. 53, pp. 675–689, 2012.

[63] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015.

[64] A. García-Pablos, M. Cuadros, and G. Rigau, "W2vlda: almost unsupervised system for aspect based sentiment analysis," *Expert Systems with Applications*, vol. 91, pp. 127–137, 2018.

[65] E. Cambria, T. Mazzocco, A. Hussain, and C. Eckl, "Sentic medoids: Organizing affective common sense knowledge in a multi-dimensional vector space," in *International Symposium on Neural Networks*, pp. 601–610, Springer, 2011.

[66] E. Cambria, J. Fu, F. Bisio, and S. Poria, "Affectivespace 2: Enabling affective intuition for concept-level sentiment analysis," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 508–514, 2015.

[67] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[68] M. Lan, S.-Y. Sung, H.-B. Low, and C.-L. Tan, "A comparative study on term weighting schemes for text categorization," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 1, pp. 546–551, IEEE, 2005.

[69] H. Altınçay and Z. Erenel, "Analytical evaluation of term weighting schemes for text categorization," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1310–1323, 2010.

[70] M. Steinbach, G. Karypis, V. Kumar, *et al.*, "A comparison of document clustering techniques," in *KDD workshop on text mining*, vol. 400, pp. 525–526, Boston, 2000.

[71] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[72] J. Sedding and D. Kazakov, "Wordnet-based text document clustering," in *proceedings of the 3rd workshop on robust methods in analysis of natural language data*, pp. 104–113, Association for Computational Linguistics, 2004.

[73] A. Hotho, S. Staab, and G. Stumme, "Ontologies improve text document clustering," in *Third IEEE international conference on data mining*, pp. 541–544, IEEE, 2003.

[74] D. R. Recupero, "A new unsupervised method for document clustering by using wordnet lexical and conceptual relations," *Information Retrieval*, vol. 10, no. 6, pp. 563–579, 2007.

[75] H. Cunningham, "Gate: A framework and graphical development environment for robust nlp tools and applications," in *Proc. 40th annual meeting of the association for computational linguistics (ACL 2002)*, pp. 168–175, 2002.

[76] S. Fodeh, B. Punch, and P.-N. Tan, "On ontology-driven document clustering using core semantic features," *Knowledge and information systems*, vol. 28, no. 2, pp. 395–421, 2011.

[77] T. Wei, Y. Lu, H. Chang, Q. Zhou, and X. Bao, "A semantic approach for text clustering using wordnet and lexical chains," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2264–2275, 2015.

[78] T. Slimani, "Description and evaluation of semantic similarity measures approaches," *arXiv preprint arXiv:1310.8059*, 2013.

[79] D. Jayarajan, D. Deodhare, and B. Ravindran, "Document clustering using lexical chains," 2007.

[80] J. Morris and G. Hirst, "Lexical cohesion computed by thesaural relations as an indicator of the structure of text," *Computational linguistics*, vol. 17, no. 1, pp. 21–48, 1991.

[81] G. Hirst, D. St-Onge, *et al.*, "Lexical chains as representations of context for the detection and correction of malapropisms," *WordNet: An electronic lexical database*, vol. 305, pp. 305–332, 1998.

[82] H. G. Silber and K. F. McCoy, "Efficiently computed lexical chains as an intermediate representation for automatic text summarization," *Computational Linguistics*, vol. 28, no. 4, pp. 487–496, 2002.

[83] D. Jayarajan, D. Deodhare, and B. Ravindran, "Lexical chains as document features," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, 2008.

[84] Z. Liu, Y. Lin, and M. Sun, *Representation Learning for Natural Language Processing*. Springer Nature, 2020.

[85] J. R. Firth, "A synopsis of linguistic theory, 1930-1955," *Studies in linguistic analysis*, 1957.

[86] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[87] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word

representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[88] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[89] S. Wang, W. Zhou, and C. Jiang, "A survey of word embeddings based on deep learning," *Computing*, vol. 102, no. 3, pp. 717–740, 2020.

[90] J. Camacho-Collados and M. T. Pilehvar, "From word to sense embeddings: A survey on vector representations of meaning," *Journal of Artificial Intelligence Research*, vol. 63, pp. 743–788, 2018.

[91] F. Almeida and G. Xexéo, "Word embeddings: A survey," *arXiv preprint arXiv:1901.09069*, 2019.

[92] A. Bakarov, "A survey of word embeddings evaluation methods," *arXiv preprint arXiv:1801.09536*, 2018.

[93] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.

[94] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, pp. 3111–3119, 2013.

[95] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[96] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[97] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[98] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[99] Z. Huang, "A fast clustering algorithm to cluster very large categorical data sets in data mining.," *DMKD*, vol. 3, no. 8, pp. 34–39, 1997.

[100] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proceedings of VLDB*, pp. 144–155, 1994.

[101] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *IEEE transactions on knowledge and data engineering*, vol. 14, no. 5, pp. 1003–1016, 2002.

[102] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algo-

rithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[103] S. Z. Selim and K. Alsultan, "A simulated annealing algorithm for the clustering problem," *Pattern recognition*, vol. 24, no. 10, pp. 1003–1008, 1991.

[104] X. Han, L. Quan, X. Xiong, M. Almeter, J. Xiang, and Y. Lan, "A novel data clustering algorithm based on modified gravitational search algorithm," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 1–7, 2017.

[105] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary computation*, vol. 16, pp. 1–18, 2014.

[106] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, pp. 478–487, 2016.

[107] R. Kruse, C. Döring, and M.-J. Lesot, *Advances in fuzzy clustering and its applications*. John Wiley and Sons, England, 2007.

[108] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, pp. 32–57, 1973.

[109] J. C. Bezdek, "Objective function clustering," in *Pattern recognition with fuzzy objective function algorithms*, pp. 43–93, Springer, 1981.

[110] T. J. Ross, *Fuzzy logic with engineering applications*. John Wiley & Sons, 2005.

[111] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *1978 IEEE conference on decision and control including the 17th symposium on adaptive processes*, pp. 761–766, IEEE, 1979.

[112] M. L. Rudolf Kruse, Christian Dring, "Fundamentals of fuzzy clustering," in *Advances in Fuzzy Clustering and its Applications* (W. P. J. Valente de Oliveira, ed.), ch. 1, pp. 3–30, Oxford: wiley, 2007.

[113] W. Pedrycz and J. Waletzky, "Fuzzy clustering with partial supervision," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 5, pp. 787–795, 1997.

[114] Z.-d. Wu, W.-x. Xie, and J.-p. Yu, "Fuzzy c-means clustering algorithm based on kernel method," in *Proceedings Fifth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2003*, pp. 49–54, IEEE, 2003.

[115] D. Zhang, K. Tan, and S. Chen, "Semi-supervised kernel-based fuzzy c-means," in *International Conference on Neural Information Processing*, pp. 1229–1234, Springer, 2004.

[116] A. Bouchachia and W. Pedrycz, "Enhancement of fuzzy clustering by mechanisms of partial supervision," *Fuzzy Sets and Systems*, vol. 157, no. 13, pp. 1733–1759, 2006.

[117] D. T. C. Lai and J. M. Garibaldi, "A comparison of distance-based semi-supervised

fuzzy c-means clustering algorithms," in *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pp. 1580–1586, IEEE, 2011.

[118] C. Li, L. Liu, and W. Jiang, "Objective function of semi-supervised fuzzy c-means clustering algorithm," in *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, pp. 737–742, IEEE, 2008.

[119] E. Yasunori, H. Yukihiro, Y. Makito, and M. Sadaaki, "On semi-supervised fuzzy c-means clustering," in *2009 IEEE International Conference on Fuzzy Systems*, pp. 1119–1124, IEEE, 2009.

[120] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, pp. 103–114, ACM, 1996.

[121] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM Sigmod Record*, vol. 27, pp. 73–84, ACM, 1998.

[122] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," *Information systems*, vol. 25, no. 5, pp. 345–366, 2000.

[123] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[124] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[125] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[126] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod record*, vol. 28, pp. 49–60, ACM, 1999.

[127] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Data Engineering, 1998. Proceedings., 14th International Conference on*, pp. 324–331, IEEE, 1998.

[128] A. Hinneburg, D. A. Keim, *et al.*, "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, vol. 98, pp. 58–65, 1998.

[129] W. Wang, J. Yang, R. Muntz, *et al.*, "Sting: A statistical information grid approach to spatial data mining," in *VLDB*, vol. 97, pp. 186–195, 1997.

[130] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," in *VLDB*, vol. 98, pp. 428–439, 1998.

[131] A. Hinneburg and D. A. Keim, "Optimal grid-clustering: Towards breaking the

curse of dimensionality in high-dimensional clustering," in *Proceedings of the 25th International Conference on Very Large Databases, 1999*, pp. 506–517, 1999.

[132] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data," *Data Mining and Knowledge Discovery*, vol. 11, no. 1, pp. 5–33, 2005.

[133] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[134] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine learning*, vol. 2, no. 2, pp. 139–172, 1987.

[135] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.

[136] J. H. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," *Artificial intelligence*, vol. 40, no. 1-3, pp. 11–61, 1989.

[137] G. Sehgal and D. K. Garg, "Comparison of various clustering algorithms," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 3074–3076, 2014.

[138] L. Yue, W. Zuo, T. Peng, Y. Wang, and X. Han, "A fuzzy document clustering approach based on domain-specified ontology," *Data & Knowledge Engineering*, vol. 100, pp. 148–166, 2015.

[139] Y. Li, C. Luo, and S. M. Chung, "A parallel text document clustering algorithm based on neighbors," *Cluster Computing*, vol. 18, no. 2, pp. 933–948, 2015.

[140] J. A. Nasir, I. Varlamis, A. Karim, and G. Tsatsaronis, "Semantic smoothing for text clustering," *Knowledge-Based Systems*, vol. 54, pp. 216–229, 2013.

[141] C. Bouras and V. Tsogkas, "A clustering technique for news articles using wordnet," *Knowledge-Based Systems*, vol. 36, pp. 115–128, 2012.

[142] A. Huang, D. Milne, E. Frank, and I. H. Witten, "Clustering documents using a wikipedia-based concept representation," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 628–636, Springer, 2009.

[143] R. Baghel and R. Dhir, "Text document clustering based on frequent concepts," in *2010 First International Conference On Parallel, Distributed and Grid Computing (PDGC 2010)*, pp. 366–371, IEEE, 2010.

[144] C. Luo, Y. Li, and S. M. Chung, "Text document clustering based on neighbors," *Data & Knowledge Engineering*, vol. 68, no. 11, pp. 1271–1288, 2009.

[145] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?," in *International conference on database theory*, pp. 217–235, Springer, 1999.

[146] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in statistical physics*, pp. 273–309, Springer, 2004.

[147] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[148] T. Hofmann, "Probabilistic latent semantic analysis," *arXiv preprint arXiv:1301.6705*, 2013.

[149] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[150] J. Kim, J. Yoon, E. Park, and S. Choi, "Patent document clustering with deep embeddings," *Scientometrics*, pp. 1–15, 2020.

[151] T. Duan, Q. Lou, S. N. Srihari, and X. Xie, "Sequential embedding induced text clustering, a non-parametric bayesian approach," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 68–80, Springer, 2019.

[152] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[153] S. J. Fodeh, W. F. Punch, and P.-N. Tan, "Combining statistics and semantics via ensemble model for document clustering," in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 1446–1450, ACM, 2009.

[154] S. Zackwehdex, "Sift4 distance." `https://siderite.dev/blog/super-fast-and-accurate-string-distance.html`, 2014. Accessed: 2019-10-15.

[155] B.-Y. Kang, D.-W. Kim, and S.-J. Lee, "Exploiting concept clusters for content-based information retrieval," *Information sciences*, vol. 170, no. 2-4, pp. 443–462, 2005.

[156] V. H. A. Soares, R. J. Campello, S. Nourashrafeddin, E. Milios, and M. C. Naldi, "Combining semantic and term frequency similarities for text clustering," *Knowledge and Information Systems*, pp. 1–32, 2019.

[157] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," tech. rep., Stanford, 2006.

[158] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, vol. 48, no. 12, pp. 4743–4759, 2018.

[159] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[160] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments,"

Neurocomputing, vol. 267, pp. 664–681, 2017.

[161] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 33–42, IEEE, 2017.

[162] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[163] M. Gagolewski, M. Bartoszuk, and A. Cena, "Genie: A new, fast, and outlier-resistant hierarchical clustering algorithm," *Information Sciences*, vol. 363, pp. 8–23, 2016.

[164] L. Ceriani and P. Verme, "The origins of the gini index: extracts from variabilità e mutabilità (1912) by corrado gini," *The Journal of Economic Inequality*, vol. 10, no. 3, pp. 421–443, 2012.

List of Publications

1. V. Mehta, S. Bawa, and J. Singh, "Analytical review of clustering techniques and proximity measures," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5995-6023, 2020. (**published, SCI indexed, Impact Factor: 8.139**)

2. V. Mehta, S. Bawa, and J. Singh, "Stamantic Clustering: Combining Statistical and Semantic Features for Clustering of Large Text Datasets," *Expert Systems with Applications*, vol. 174, pp. 114710, 2021. (**published, SCIE indexed, Impact Factor: 6.954**)

3. V. Mehta, S. Bawa, and J. Singh, "WEClustering: Word Embeddings Based Text Clustering Technique for Large Datasets," *Complex and Intelligent Systems*, 2021, doi: https://doi.org/10.1007/s40747-021-00512-9. (**published, SCIE indexed, Impact Factor: 4.927**)