Tribhuvan University

Institute of Science and Technology

Nepali Document Clustering using DBSCAN and OPTICS Algorithm

Thesis

Submitted to

Central Department of Computer Science and Technology

Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

By

Prabin Maharjan

T.U. Registration No.: 5-2-22-1661-2007

T.U. Examination Roll No.:70/069

Date (April, 2019)

Supervisor

Mr. Dhiraj Kedar Pandey

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than listed here have been used in this work.

_____

Prabin Maharjan

Date: April, 2019

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Supervisor's Recommendation

I hereby recommend that this thesis prepared under my supervision by **Mr. Prabin Maharjan** titled "**Nepali Document Clustering using DBSCAN and OPTICS Algorithm**" in partial fulfillment of the requirements for the degree of MSc. in Computer Science and Information Technology be processed for the evaluation.

_____

Asst. Prof. Dhiraj Kedar Pandey

Central Department of Computer Science and Information Technology,

Tribhuvan University,

Kathmandu, Nepal

(Supervisor)

Date: April, 2019

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

We certify that, we have read this thesis and in our opinion it is satisfactory in the scope and quality as a thesis in partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology.

Evaluation Committee

Asst. Prof. Nawaraj Poudel Pandey

Central Department of CSIT

Tribhuwan University

Kirtipur, Kathmandu, Nepal

(Head of Department)

Asst. Prof. Dhiraj Kedar

Central Department of CSIT

Tribhuwan University

Kirtipur, Kathmandu, Nepal

(Supervisor)

(External Examiner)

(Internal Examiner)

Acknowledgement

I offer my profound gratitude to my supervisor Asst. Prof. Dhiraj Kedar Pandey (Tribhuvan University) for his generous advice, inspiring guidance and encouragement throughout my research for this dissertation. Without his kind and patient review of this work, it would have been impossible to complete this study.

I would like to extend my gratitude to Asst. Prof. Nawaraj Paudel (Head of Department, CDCSIT) and faculties for their guidance and help throughout my Masters Study and help for the completion of my dissertation.

I would like to express my gratitude to all my family members, friends and all other people who have helped me directly or indirectly in the completion of this dissertation.

I have given my best effort to make this thesis work complete and error free. However, I am always looking forward to the suggestions from the readers which will improve this work.

Abstract

Automated document clustering is the process of grouping documents into a small sets of meaningful collections based on similarity between them. This research evaluates density based clustering algorithms namely Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Ordering points to Identify Cluster Structure(OPTICS) algorithms using four performance metrics: Homogeneity, Completeness, V-Measure and Silhouette Coefficient on Nepali dataset. Features extraction is done using combination of Term Frequency – Inverse Document Frequency (TFIDF) with Latent Semantic Indexing (LSI). The results based on the performance metrics mentioned above shows that clustering result of DBSCAN is slightly better than OPTICS algorithm. The time required for processing is better for DBSCAN algorithm.

Keywords: Clustering, Machine Learning, Nepali Document Clustering, DBSCAN, OPTICS TFIDF, LSI

Table of Contents

List of Tables

List of Figures

Abbreviations

API	Application Programming Interface
CBAG	Continuous Bag of Words
CSV	Comma Separated Values.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise.
HTML	Hyper Text Markup Language.
IDE	Integrated Development Environment.
KU	Kathmandu University.
LSI	Latent Semantic Indexing.
MPP	Madan Puraskar Pustakalaya.
NER	Named-Entity Recognition.
NLP	Natural Language Processing.
OPTICS	Ordering Points to Identify the Clustering Structure.
PoS	Part-of-Speech.
TFIDF	Term Frequency – Inverse Document Frequency.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Cluster analysis or clustering is the task of grouping a set of objects in such a way that the same group (clusters) are more similar to each other than other groups (clusters). Due to availability of affordable computing devices, more and more documents are stored and transferred in digitized form each day because of which many local and web documents are created. Grouping these into a small number of meaningful and coherent clusters provides a good way for informative navigation as well as browsing mechanism. Data clustering useful in fields like data mining, statistics, machine learning, biology, marketing etc.

Due to the accumulations of huge amounts of data in databases and flat files, cluster analysis has become highly significant, especially in research fields utilizing data mining techniques [1]. Clustering is especially useful for organizing documents to improve retrieval and support browsing [2]. So it is also possible to implement these clustering techniques on Nepali datasets.

Density based clustering such as DBSCAN and OPTICS has gained a lot of attention of late, mainly due to its advantage over traditional clustering techniques like K-means [1] because density based algorithms don't require the number of clusters to be estimated prior to clustering. So it is natural to implement density based algorithms on Nepali dataset and extract the benefits provided by them.

1.2 Problem Statement

Due to the accumulations of huge amounts of data in databases and flat files, cluster analysis has become highly significant, especially. Nepali is a morphologically rich and complex language due to its inflectional and derivative nature in formation of words .Thus, cluster analysis in Nepali language is considered as challenging task. Traditional partitioning based approaches has not been effective with data of different size and density and is difficult to predict the seed and the decide the number of partitions . DBSCAN and OPTICS algorithms overcomes these

limitations. The concentration of this research is on the comparison of the performances of density based algorithms DBSCAN and OPTICS algorithms on Nepali dataset.

1.3 Objectives

The objectives of the research are as follows:

1. To build feature representation scheme for text documents using TFIDF with LSI.

2. To cluster Nepali text documents using DBSCAN and OPTICS algorithms and compare their performances using metrics like homogeneity, completeness, v-measure and silhouette of each algorithm.

1.4 Scope and Limitation

This study will focus on studying the performance of DBSCAN and OPTICS algorithms on a Nepali dataset. Cluster quality will be analyze based on four performance measures: Homogeneity, Completeness, V-Measure and silhouette coefficient whereas performance analysis based on completion time will also be studied.

Application Programming Interfaces (APIs) from Scikit-Learn library will be used for feature extraction as well as clustering. The algorithm computes the $O(n^2)$ distance matrix which is costly from the perspective of memory and time .This study limits the dataset size upto 10,000 samples.

1.5 Report Organization

The outline of the dissertation is as follows:

Chapter 2 discusses about the review of existing techniques related to DBSCAN and OPTICS cluster analysis and Nepali Document Clustering.

Chapter 3 discusses about the methodology that are used in this dissertation.

Chapter 4 discuses about the implementation of the algorithms with programming language, frameworks and methods with parameters

Chapter 5 discusses about the experimental analysis of the algorithm using evaluation metrics for analyzing cluster quality and time.

Chapter 6 discusses about the conclusion and future work of this research

CHAPTER 2

LITERATURE REVIEW

2.1 BACKGROUND

2.1.1 Cluster Analysis

Cluster analysis emerged as a major topic in the 1960's and 1970's when the onograph 'Principles of Numerical Taxonomy' by Sokal and Sneath [3], published in 1963, motivated worldwide research on clustering methods. Cluster analysis is an important human activity. By automated clustering, we can identify dense and sparse regions in object space and, therefore, discover overall distribution patterns and interesting correlations among data attributes. Cluster analysis has been widely used in numerous applications, including market research, pattern recognition, data analysis, and image processing [1]. Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their similarity. Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases. Unlike classification, clustering and unsupervised learning do not rely on predefined classes and class-labeled training examples. For this reason, clustering is a form of learning by observation, rather than learning by examples [1].

2.2 Similarity measures:

For data mining tasks like clustering, classification, and information retrieval, a notion of similarity or distance between the documents is necessary. Clusters should consist of points separated by small distances, relative to the distances between clusters. However, there are a many definitions of distance in this context, and the results of a cluster analysis may depend quite strongly on the distance measure chosen [1, 4]. The most commonly used similarity measure in text data mining and information retrieval is the cosine of the angle between vectors representing the documents shown in eq. Given two document vectors $\vec{a}$ and $\vec{b}$, the cosine of the angle between them, θ, is given by:

$$\cos \theta = \frac{\vec{a}.\vec{b}}{|\vec{a}|.|\vec{b}|} \quad \ldots\ldots\ldots \quad (2.1)$$

Larger values of this measure indicate documents are close together, and smaller values indicate the documents are further apart [5].

One of the popular distance metric in analyzing the clusters is Euclidean distance which is defined as:

$$d_{i,j}=\sqrt{\sum_{k=1}^{n}\left(x_{i,k}- x_{j,k}\right)^2} \quad \ldots\ldots\ldots\ldots \quad (2.2)$$

where $x_{i,1}$, $x_{i,2}$ $x_{i,3}$…….. $x_{i,n}$ and $x_{j,1}$, $x_{j,2}$ $x_{j,3}$…….. $x_{j,n}$ two n dimensional objects[1].

Another well-known distance metric is Manhattan (or city block) distance, defined as:

$$d_{i,j}=\sum_{k=1}^{n}\left(\left|x_{i,j} - x_{j,k}\right|\right) \quad \ldots\ldots\ldots\ldots \quad (2.3)$$

Manhattan distance metric is consistently more preferable than the Euclidean distance metric for high dimensional data mining applications [6].

Minkowski distance is a generalization of both Euclidean distance and Manhattan distance. It is defined as:

$$d_{i,j}=\sum_{k=1}^{n}\left(\left|x_{i,j} - x_{j,k}\right|^{p}\right)^{1/p} \quad \ldots\ldots\ldots\ldots \quad (2.4)$$

This distance is also called Lp norm. It represents the Manhattan distance when p = 1 (L1 norm) and Euclidean distance when p = 2 (L2 norm). [1, 2].

2.3 Types of Clustering Methods

It can be hard to categorize the clustering algorithms as they may have features from several categories. However, it is still useful to divide them into separate groups. The major clustering algorithms can be broadly organized into following categories [2]:

- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Model-Based Methods

2.3.1 Partitioning Methods

For a given dataset with n objects, partitioning method partitions the data into k unit, where $0<k\leqslant n$ and each partition is considered as cluster. The partitions must satisfy the following requirements:

- Each group must contain at least one object, and
- Each object must belong to exactly one group.

Most applications uses one of a few popular heuristic methods, such as the K-Means algorithm, where each cluster is represented by the mean value of the objects in the cluster, and the K-Medoids algorithm, where each cluster is represented by one of the objects located near the center of the cluster [2, 7].

2.3.2 Hierarchical Methods

Hierarchical techniques produce a nested sequence of partitions, with a single, all-inclusive cluster at the top and singleton clusters of individual points at the bottom. Each intermediate level can be viewed as combining two clusters from the next lower level (or splitting a cluster from the next higher level). The result of a hierarchical clustering algorithm can be graphically displayed as tree, called a dendogram [8]. A hierarchical method can be classified as being either

agglomerative or divisive, based on how the hierarchical decomposition is formed. The agglomerative (bottom-up) approach starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition is reached. The divisive (top-down) approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until each object is in one cluster. In hierarchical methods, once a merge or split is done, it can never be undone. This rigid behaviour leads to smaller computation costs but it may cause problems when error corrections have to be done.

2.3.3 Density-Based Methods

Most partitioning methods cluster objects based on the distance between objects. They works by detecting areas where point are concentrated and where they are separated by areas that are empty or sparse. A given cluster can grow as long as the density (number of objects or data points) in the “neighborhood” exceeds some threshold. This means, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise or outliers and discover clusters of arbitrary shape. DBSCAN and Ordering Points to Identify the Clustering Structure (OPTICS) are typical examples of density-based partition methods [2].

2.3.4 Grid-Based Methods

Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on this grid. The main advantage of this approach is its fast processing time as it does not depend upon the number of objects in the dataset. Quality of cluster can be controlled by varying the number of cells in the quantized space. STING can be taken as example of Grid Based method[2, 9, 10].

2.3.5 Model-Based Methods

Clustering algorithms can also be developed based on probability models, such as the finite mixture model for probability densities. The word model is usually used to represent the type of

constraints and geometric properties of the covariance matrices. In the family of model based clustering algorithms, one uses certain models for clusters and tries to optimize the fit between the data and the models. In the model-based clustering approach, the data are viewed as coming from a mixture of probability distributions, each of which represents a different cluster. It is assumed that the data are generated by a mixture of probability distributions in which each component represents a different cluster. Thus a particular clustering method can be expected to work well when the data conform to the model. It also leads to a way of automatically determining the number of clusters based on standard statistics, taking "noise" or outliers into account and thus yielding robust clustering methods. Expectation–Maximization (EM) is an algorithm that performs expectation-maximization analysis based on statistical modeling.

2.4 Clustering Algorithms:

Clustering starts with a dataset D containing a set of points $p \in D$. Density-based algorithms need to obtain a density estimate over the data space. DBSCAN estimates the density around a point using the concept of ε -neighborhood.

Definition 2.1:

ε –Neighborhood: The ε -neighborhood, $N\epsilon$ (p), of a data point p is the set of points within a specified radius ε around p. $N\epsilon(p) = \{q \mid d(p, q) < \epsilon\}$, where d is some distance measure and $\epsilon \in R+$. The point p is always in its own ε-neighborhood, i.e., $p \in N\epsilon(p)$ always holds.

Definition 2.2:

Point classes: A point $p \in D$ is classified as

- a core point if , $N\epsilon$ (p), has high density, i.e., $|N\epsilon(p)| \geq \text{minPts}$ where $\text{minPts} \in Z+$ is a user-specified density threshold,
- a border point if p is not a core point, but it is in the neighborhood of a core point $q \in D$, i.e., $p \in N\epsilon(p)$
- otherwise it is considered as a noise point,

Definition 2.3:

Directly density-reachable. A point q ∈ D is directly density-reachable from a point p ∈ D with respect to ε and minPts if, and only if,

- $|N\epsilon\ (p)| \geq$ minPts, and
- q ∈ $|N\epsilon\ (p)|$. That is, p is a core point and q is in its ε -neighborhood.

Definition 2.4:

Density-reachable: A point p is density-reachable from q if there exists in D an ordered sequence of points (p_1, p_2, ..., p_n) with q = p_1 and p = p_n such that p_i +1 directly density-reachable from p_i ∀ i ∈ {1, 2, ..., n − 1}.

Definition 2.5.

Density-connected: A point p ∈ D is density-connected to a point q ∈ D if there is a point o ∈ D such that both p and q are density-reachable from o.

Definition 2.6.

Cluster: A cluster C is a non-empty subset of D satisfying the following conditions:

- Maximality: If p ∈ C and q is density-reachable from p, then q ∈ C; and
- Connectivity: ∀ p, q ∈ C, p is density-connected to q.

2.4.1 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996[8]. It is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).

Density based clustering methods cluster data based on a local cluster criterion such as density connected points. Typically, density based algorithms can discover clusters of arbitrary shapes and are relatively noise tolerant. DBSCAN [8], the earliest density based clustering algorithm, introduces many concepts which are used by later density based clustering algorithms. It classifies points as core points if they have many data elements in their vicinity. Thereafter, a cluster can be represented by the set of core points it contains. The algorithm can identify clusters of arbitrarily shape opposed to K-Means and its variant [11] .

2.4.2 OPTICS

Ordering points to identify the clustering structure (OPTICS) is an algorithm for finding density-based clusters in spatial data. It was presented by Mihael Ankerst, Markus M. Breunig, Hans-peter-criegan and Jorg Sander [4]. Its basic idea is similar to DBSCAN, but it addresses one of DBSCAN's major weaknesses: the problem of detecting meaningful clusters in data of varying density [4]. OPTICS produces an augmented ordering of the elements in the dataset representing its clustering structure and has been shown to be quite insensitive to the input parameters[12] provided that the values of the parameters are large enough to get a 'good' result. OPTICS builds a reachability plot, in which valleys correspond to clusters[13,14]. The OPTICS plot is the plot of data elements, against their reachability distance, data elements ordered according to the time at which OPTICS stops considering them. The reachability distance of an element is determined by the distance to its nearest core point which has already been considered by OPTICS. Relative insensitivity to parameters (which enables it to identify clusters of varying densities) was the main motivation for me to choose OPTICS from among other density based algorithm[15]. Sometimes it is also understood as extended DBSCAN because it uses the parameters minimum points and the radius for the neighborhood points same as in DBSCAN.

In optics additional concepts called core-distance and reachability distance is used. All used distances are calculated using the Euclidean distance used for the neighborhood calculation.

The core-distance of a point $p \in D$ with respect to minPts and ε is defined as:

$$\text{core-dist}(p; \varepsilon, \text{minPts}) = \begin{cases} undefined & if\ |N_\varepsilon(p)| < \text{minPts and,} \\ minPts - dist(p) & otherwise \end{cases} \quad \ldots\ldots\ldots \quad (2.5)$$

where minPts-dist(p) is the distance from p to its minPts − 1 nearest neighbor, i.e., the minimal radius a neighborhood of size minPts centered at and including p would have[16].

The reachability-distance of a point p ∈ D to a point q ∈ D parameterized by ε and minPts is defined as:

$$\text{reachability-dist}(p, q; \varepsilon, \text{minPts}) = \begin{cases} undefined & if\ |N_\varepsilon(p)| < \text{minPts and,} \\ \max(core - dist(p), d(p,q)) & otherwise \end{cases} \quad \ldots\ldots\ldots \quad (2.6)$$

The reachability-distance of a core point p with respect to object q is the smallest neighborhood radius such that p would be directly density-reachable from q[16].

2.5 Work on Nepali Language:

Since the release of Nepali spell checker in 2005, various works on Nepali Natural language processing began. The same year, KU (Kathmandu University) along with MPP (Madan Puraskar Pustakalaya) completed an English to Nepali translation system दोभासे as an Acia-Pacific Development Information Program[6]. After that many research on Nepali like Part-of-speech (PoS) [8, 9,17] tagging, Stemming[2],Named Entity Recognition[18] etc has been done. Some text clustering for Nepali document has been done [19], there is still lack of comprehensive works on Nepali Document Clustering [2,4].

In 2014, S. Sarkar, A. Roy and B.S. Purkayastha presented a comparative analysis of K-Means, Particle Swarm Optimization (PSO) and hybrid PSO+K-Means algorithm for clustering of Nepali text documents and performed experimental evaluation by using intra cluster similarity and inter-cluster similarity [19]. Similarly A. Neupane published another paper that was used to create Nepali character dataset using semi-supervised clustering approach in the same year. Two algorithms EM and K-Means were used to create the database using extracted features from both handwritten and scanned Nepali text [5].

The very same year, C. Sitaula also proposed an algorithm which combines the advantage of classical vector space model to cluster the semantic texts and ideas from fuzzy logic. The algorithm treated text having similar context words as semantic texts. It used the concept of advanced enhanced vector space model obtained by adding TFIDF with fuzzy membership value and perform the cosine operation in order to calculate the semantic distance between texts [3].

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Dataset Preparation

The official written script for Nepali is Devnagari which is an abugida (alphasyllabary) which contains unicode code points from U+0900 to U+097F to represent Nepali characters and symbols. Abugida is also the most commonly used devanagari script. Only a subset of these code points are used in current version of Nepali language. The vowels(अं,अ) and consonants(ज्ञ,त्र,क्ष) do not have separate code points but they are considered as single object. The vowels consonants and numerals for Nepali language are as follows.

Vowels:	(अ,आ,इ,ई,उ,ऊ,ए,ऐ,ओ,औ,अं,अः,ऋ)
Consonants:	(क,ख,ग,घ,ङ,च,छ,ज,झ,ञ,ट,ठ,ड,ढ,ण,त,थ,द,ध,न,प,फ,ब,भ,म,य,र,ल,व,श,ष,स,ह,क्ष,त्र,ज्ञ)
Numerals:	(१,२,३,४,५,६,७,८,९,०).

Table 3.1: Nepali Characters

Different articles will be collected from various nepali news portals like www.ekantipur.com,www.setopati.com etc. A dataset upto 10000 sample has been used for the study.

3.2 Basic Framework for Nepali Document Clustering:

At first the raw data is collected which undergoes various preprocessing operations like symbols and number removal, tokenization, stop-words removal and stemming. Then the feature extraction techniques like TF-IDF with LSI is used. After extraction those features are passed to clustering algorithms (DBSCAN and OPTICS) and finally the performance of both the algorithms are evaluated using evaluation metrics (homogeneity, completeness, vmeasure and Silhouette's coefficient).

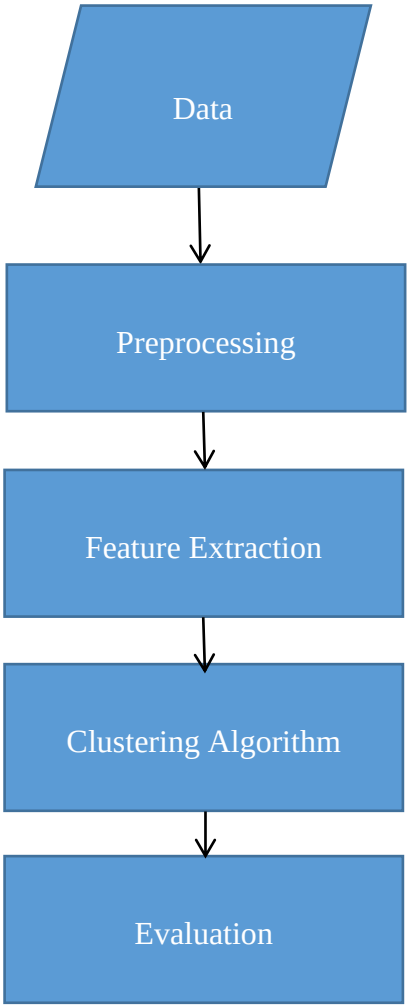

Figure 3.1**:** General Structure of Text Clustering

3.3 Preprocessing

Raw corpus taken from the news websites contains many unnecessary words like punctuations and repeating words which does not help much in clustering text. Taking out those words from the corpus may reduce the computation time and improve the result of cluster analysis. The following preprocessing steps will be used on the corpus:

3.3.1 Tokenization:

This step breaks each individual documents in the corpus, first into continuous bag of words(CBAG).In Nepali language punctuation marks like danda(।),double danda(॥),question mark(?) can be used to break down the sentences and space, comma colon and semicolon can be used to breakdown the words.

3.3.2 Special Symbol and HTML Tag Removal:

Punctuation marks and HTML tags, which do not have any significance to the corpus are removed.

3.3.3 Stop Word Removal:

Stop words are the words which have very high frequency in the corpus. They either do not contribute anything or their contribution is negligible in differentiating documents. So common stop words like म,आफु,ऊ,त्यो,तिमी, etc. are removed.

3.3.4 Stemming:

Stemming is the process of removing inflectional or derivational affixes from the document [9]. In devanagari language many compound words are formed by merging root word with affixes like रामको,रामबाट,रामले, etc. So stemming removes the affixes like को,बा,ले,and get root word राम.

3.3.5 Text Representation:

The text is represented in numeric form using TFIDF (term frequency-inverse document frequency) which is further processed by Latent Semantic Indexing (LSI) that reduces the dimension of the term document matrix and produces the low dimension matrix.

The formula for TFIDF is:

$$W_{i,j}=tf_{i,j}\times\log(N/df_i)\ldots\ldots\ldots\ldots \quad (3.1)$$

where, $W_{i,j}$ is the weight for term i in document j, N is the number of documents in the corpus, $tf_{i,j}$ is the term frequency of term i in document j and df_i is the document frequency of term i in the corpus[1]. TFIDF can tell whether a term is relevant or not with the topic of the document[20].

Since the computation of TFIDF for large set of documents is time and space consuming due to high dimensionality of TFIDF vector the dimensionality reduction is performed on TFIDF vector using LSI method.

LSI is a popular linear algebraic indexing method to produce low dimensional representation. The idea behind LSI is to take advantage of implicit higher order structure in the association of terms with documents (‘‘semantic structure”) in order to improve the detection of relevant documents, on the basis of terms found in queries [21].. For a term-document matrix $X = [x_1, x_2,\ldots. x_n] \in R^m$ with rank r, LSI decomposes X using SVD as follows:

$$X = U\Sigma V^T \quad (3.2)$$

where $\Sigma = \text{diag}(\sigma_1,\ldots\ldots., \sigma_r)$ and $\sigma_1 \geqslant \sigma_2 \geqslant \ldots\sigma_r$ are the singular values of X. $U = [u_1, \ldots, u_r]$ and u_i is the left singular vector. $V = [v_1\ldots\ldots, v_r]$ and v_i is the right singular vector. LSI uses the first k vectors in U as the transformation matrix to embed the original documents into a k-dimensional space.

3.4 Clustering:

3.4.1 DBSCAN

The DBSCAN algorithm can be abstracted into the following steps:

- Find the points in the ε (eps) neighborhood of every point, and identify the core points with more than minPts neighbors.
- Find the connected components of core points on the neighbor graph, ignoring all non-core points.
- Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor, otherwise assign it to noise.

3.4.2 OPTICS

OPTICS produces an augmented ordering of the elements in the dataset representing its clustering structure and has been shown to be quite insensitive to the input parameters[12] provided that the values of the parameters are large enough to get a 'good' result. OPTICS builds a reachability plot, in which valleys correspond to clusters. The OPTICS plot is the plot of data elements, against their reachability distance, data elements ordered according to the time at which OPTICS stops considering them. The reachability distance of an element is determined by the distance to its nearest core point which has already been considered by OPTICS. Relative insensitivity to parameters (which enables it to identify clusters of varying densities) was the main motivation for us to choose OPTICS from among other density based algorithm.

The OPTICS algorithm is as follows:

OPTICS (SetOfObjects, ε, MinPts, OrderedFile)

```
OrderedFile.open();

FOR i FROM 1 TO SetOfObjects.size DO

    Object := SetOfObjects.get(i);

    IF NOT Object.Processed THEN
```

ExpandClusterOrder(SetOfObjects, Object, ε, MinPts, OrderedFile)

OrderedFile.close(); END; // OPTICS

Algorithm for ExpandCluster

```
ExpandClusterOrder(SetOfObjects, Object, ε, MinPts, OrderedFile)
neighbors := SetOfObjects.neighbors(Object, ε);
Object.Processed := TRUE;
Object.reachability_distance := UNDEFINED;
Object.setCoreDistance(neighbors, ε, MinPts);
OrderedFile.write(Object);
IF Object.core_distance <> UNDEFINED THEN
    OrderSeeds.update(neighbors, Object);
    WHILE NOT OrderSeeds.empty() DO
        currentObject:=OrderSeeds.next();
        neighbors:=SetOfObjects.neighbors(currentObject, ε);
        currentObject.Processed := TRUE;
        currentObject.setCoreDistance(neighbors,ε,MinPts);
        OrderedFile.write(currentObject);
        IF currentObject.core_distance<>UNDEFINED THEN
        OrderSeeds.update(neighbors.currentObject);
```

```
        END //ExpandClusterOrder
```

Algorithm for Ordering Seeds:

```
OrderSeeds::update(neighbors, CenterObject);
c_dist := CenterObject.core_distance;
FORALL Object FROM neighbors DO
    IF NOT Object.Processed THEN
        new_r_dist:=max(c_dist,CenterObject.dist(Object));
        IF Object.reachability_distance=UNDEFINED THEN
            Object.reachabililty_distance:=new_r_dist
            Insert(Object,new_r_dist)
        ELSE//Object already in ordered seeds
            IF new_r_dist<Object.reachability_distance THEN
                Object .reachability_distance:=new_r_dist
                Decrease(Object.reachability_distane)
        END
```

ExtractDBSCAN-Clustering (ClusterOrderedObjs,ε', MinPts)

```
// Precondition: ε' ≤ generating dist ε for ClusterOrderedObjs
 ClusterId := NOISE;
FOR i FROM 1 TO ClusterOrderedObjs.size DO
        Object := ClusterOrderedObjs.get(i);
        IF Object.reachability_distance > ε' THEN // UNDEFINED > ε
                IF Object.core_distance ≤ ε' THEN
                        ClusterId := nextId(ClusterId);
                        Object.clusterId := ClusterId;
                ELSE Object.clusterId := NOISE;
        ELSE // Object.reachability_distance ≤ ε'
                Object.clusterId := ClusterId;
END; // ExtractDBSCAN-Clustering
```

3.5 Performance Evaluation Parameters

For the purposes of the following discussion, a data set comprising N data points, and two partitions of these, a set of classes, C = $\{c_i | i = 1, \dots, n\}$ and a set of clusters, , K = $\{k_i | i = 1, \dots, m\}$ has been assumed.

3.5.1 Homogeneity

The result of a clustering operation satisfies homogeneity if each of the clusters contain data points from a single class only. The determination of how close a given clustering is to this ideal

is done by examining the conditional entropy of the class distribution given the proposed clustering. In a perfectly homogeneous case, H(C|K) = 0. However this in not the case in almost all situations. Usually, the size of this value is dependent on the size of the dataset and the distribution of class sizes. Hence, instead of taking the raw conditional entropy, this value is normalized by the maximum reduction in entropy the clustering information could provide, specifically, H(C).

H(C|K) = H(C) and is maximal when the clustering provides no new information. H(C|K) = 0 when each cluster contains only members of a single class and the clustering is perfectly homogeneous. In this degenerate case (H(C|K) = 0), when there is only a single class, homogeneity is defined as 1. So following the notion of 1 being desirable and 0 being undesirable, the homogeneity is defined as[22]:

$$H=\begin{cases} 1 & if\ \mathrm{H(C|K)} = 0, \\ 1 - H(C|K)/H(C) & otherwise \end{cases} \quad \ldots\ldots\ldots\ldots \quad (3.2)$$

where,

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{a_{c_k}}{N} log \frac{a_{c_k}}{\sum_{c=1}^{|C|} a_{c_k}} \quad \ldots\ldots\ldots\ldots \quad (3.3)$$

$$H(C) = -\sum_{c=1}^{|C|}\frac{\sum_{k=1}^{|K|} a_{c_k}}{n} log \frac{\sum_{k=1}^{|K|} a_{c_k}}{n} \quad \ldots\ldots\ldots\ldots \quad (3.4)$$

3.5.2 Completeness

Completeness is a metric symmetrical to homogeneity. The result of a clustering operation satisfies completeness if all the data points that are members of a given class are elements of the same cluster. In the perfectly complete case, H(K|C) = 0 and in the worst case scenario, each class is represented by every cluster with a distribution equal to the distribution of cluster sizes, i.e., H(K|C) = H(K) and is maximal. When there is a single cluster, completeness is defined as 1. The completeness is defined as[22]:

$$C=\begin{cases} 1 & if\ \mathrm{H(K|C)} = 0, \\ 1 - H(K|C)/H(K) & otherwise \end{cases} \quad \ldots\ldots\ldots \quad (3.5)$$

Where,

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{a_{c_k}}{N} log\frac{a_{c_k}}{\sum_{k=1}^{|K|} a_{c_k}} \quad \ldots\ldots\ldots \quad (3.6)$$

$$H(C) = -\sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|} a_{c_k}}{n} log\frac{\sum_{c=1}^{|C|} a_{c_k}}{n} \quad \ldots\ldots\ldots \quad (3.7)$$

3.5.3 V-measure:

V-measure is the weighted harmonic mean of homogeneity and completeness, given by [22]:

$$\mathrm{V}_\beta = \frac{(1+\beta)hc}{(\beta h) + c} \quad (3.8)$$

If β > 1 completeness is weighted more strongly in the calculation. Conversely, if β < 1, homogeneity is weighted more strongly. The β has been set to 1 so,

$$\mathrm{V}_\beta = \frac{2hc}{h + c} \quad (3.9)$$

These measures can be applied to any clustering analysis irrespective of number of data points (n-invariance), number of classes or number of clusters [22].

3.5.4 Silhouette Coefficient

The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If

many points have a low or negative value, then the clustering configuration may have too many or too few clusters.

Given a data point i and clusters k let a(i) be the average distance between i and all other data within the same cluster. a(i) can then be interpreted as a measure of how well i is assigned to its cluster (smaller values are better). The average dissimilarity of point i and cluster c can be defined as the average of the distance from i to all points in c.

Let b(i) be the lowest average distance of i to all points in any other cluster, of which i is not a member. The cluster with this lowest average dissimilarity is defined as the "neighbouring cluster" of i as it is the next best fit cluster for point i. Silhouette coefficient of point i can now be defined as [23] :

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.4}$$

CHAPTER 4

IMPLEMENTATION

4.1 Programming Language and Frameworks

The algorithms has been implemented using Python and Scikit Learn in sypder IDE and jupyter lab along with pip package manager. All the algorithms have been implemented using Python (version 3.7) language, spyder (IDE) and Jupyter plugin for Python with pip as package manager. For efficiency and ease, several APIs from Scikit-Learn [24] library and Pandas data framework were used in this research. The study is carried out in a Dell n5559 with 2.5 core i7-6650 processor and 8GB RAM.

4.2 Preprocessing

4.2.1 Parser

The Parser is implemented to remove invalid characters like punctuation marks, Hyper Text Markup language (HTML) characters and any devanagari characters or symbols that do not add any meaningful contributions to the clustering process. It removes characters which are not specified in a collection of valid characters where valid characters are vowels ('अआइईउऊऋएऐओऔ), consonants('कखगघङचछजझञटठडढणतथदधनपफबभमयरलवशषसह') and dependent characters('ा ि ी ु ू ृ े ै ो ौ ं ः ँ ्') and symbols .

4.2.2 Tokenizer

The Tokenizer splits the output given by Parser algorithm using spaces as delimiters. It breaks individual documents to sentences and then to words. Sentences are separated by devnagari danda (।), devnagari double danda (॥) and question mark(?) whereas words are separated by space, comma, colon and semicolon. The tokenizer produces the list of tokens which will be further processed by stop-words removal procedure as well as Stemmer algorithm.

4.2.3 Stop-Word Remover

The Stop-Word Remover removes stop-words from the tokenized data. The list of stop-words are listed in appendix A.1. Removal of stop words is performed as:

- Get the list of stopwords from the stopwords file
- If data contains the word from the stopwords list remove that word from data

4.2.4 Stemmer

The rules for stemmer are listed in appendix A.2. Since a token may contain multiple suffixes applying rules randomly or by ascending order may produce incorrect removal of suffixes. Thus, stemming is performed as follows:

- Read the rules from rule file as specified in appendix A.2. and orders them by descending order of length.
- Based on the rules specified in the list, remove all the suffixes from the word and extract the root word only.

4.2.5 Cluster Analysis:

The cluster is analysed by measuring the time taken to complete the clustering of data and performance metrics like homogeneity, completeness, v-measure and silhouette coefficient.

CHAPTER 5

RESULT ANALYSIS AND DISCUSSION

The dataset mentioned in section 3.1 was clustered using two algorithms DBSCAN and OPTICS with various sample data sizes. The performance of algorithms was studied using four measures: Homogeneity, Completeness, V-Measure and Silhouette Coefficient. Similarly, the time taken by the algorithms was also studied using TFIDF with LSI. The parameter epsilon(ε) has set to 0.55 and minPts has been set to 10 for both DBSCAN and OPTICS algorithm after tuning. The major advantage of using LSI is its ability to transform the text representation into a lower dimensional form which helps to improve the execution time of clustering algorithm. Performance Analysis with TFIDF with LSI are shown in Tables below for both the algorithms:

Data size	Homogenity	Completeness	Vmeasure	Silhouette
2000	0.160969774	0.109250698	0.130160828	0.013862155
3000	0.243002626	0.08941403	0.130726567	0.02859803
4000	0.294304899	0.075048765	0.119599296	0.036303206
5000	0.543635695	0.188257878	0.279668264	0.070009519
6000	0.553278873	0.201509974	0.295423579	0.054442381
7000	0.558169934	0.212702205	0.308025079	0.031666692
8000	0.631318542	0.285581533	0.393266228	0.052464531
9000	0.594126833	0.319237596	0.415316419	0.047820564
10000	0.4781437	0.297075564	0.366463569	0.02209046

Table 5.1: Cluster Analysis of DBSCAN Algorithm

Data size	Homogenity	Completeness	Vmeasure	Silhouette
2000	0.153630668	0.10839689	0.127109429	0.007158962
3000	0.234345949	0.090214538	0.130277174	0.017309833
4000	0.283584799	0.075800047	0.119625195	0.022451462
5000	0.5332833	0.189665482	0.279813555	0.060810945
6000	0.545059456	0.203109368	0.295940376	0.045877948
7000	0.548729035	0.213076888	0.306958693	0.024339528
8000	0.622184885	0.285770273	0.391653581	0.047873814
9000	0.589490953	0.321275943	0.415889648	0.044045098
10000	0.472487414	0.298351358	0.365750314	0.014042888

Table 5.2: Cluster Analysis of OPTICS Algorithm

Data size	Time in secs
2000	0.918977022
3000	1.74348855
4000	3.661858797
5000	5.186458826
6000	6.857482672
7000	9.51994133
8000	12.31933904
9000	16.26611924
10000	19.06142664

Table 5.3: Time Computation of DBSCAN

Data size	Time in secs
2000	2.625620842
3000	4.958814144
4000	8.723777533
5000	13.73911858
6000	17.30082417
7000	23.89435983
8000	29.19334745
9000	37.07195163
10000	53.20847564

Table 5.4: Time computation of OPTICS

Cluster Analysis comparison of DBSCAN and OPTICS algorithm is shown in chart below:

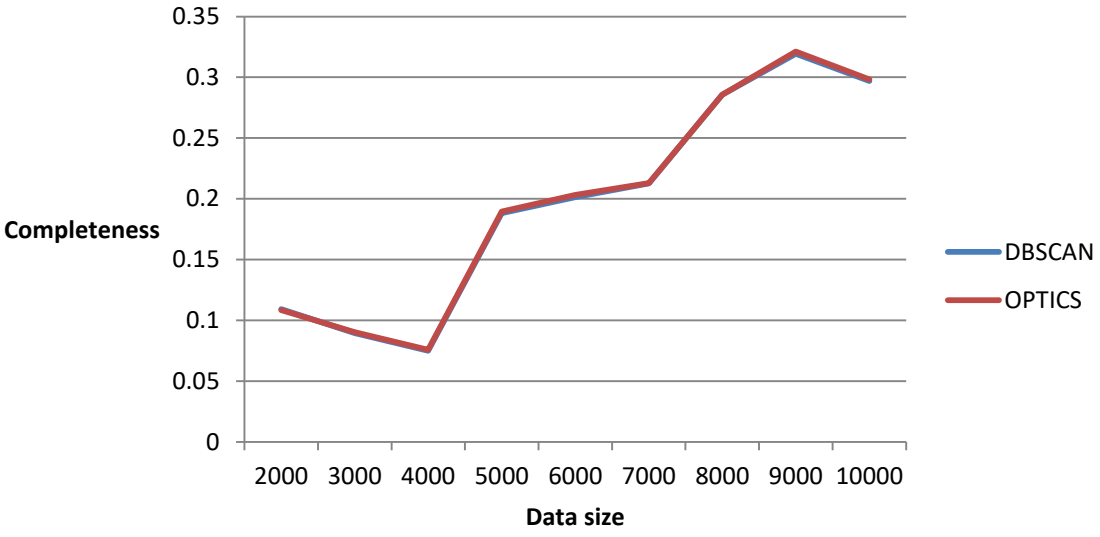

Figure 5.1**:** Completeness comparison

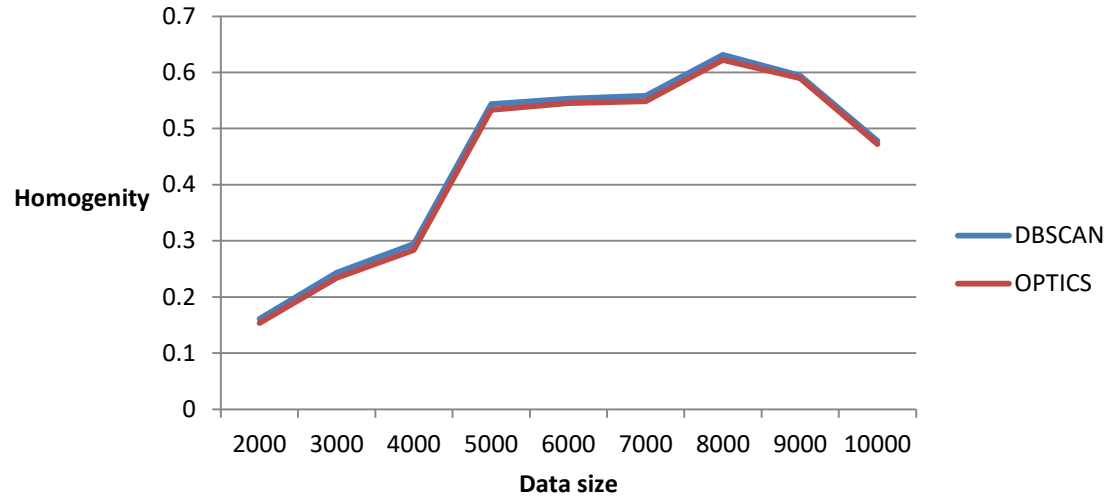

Figure 5.2: Homogenity comparison

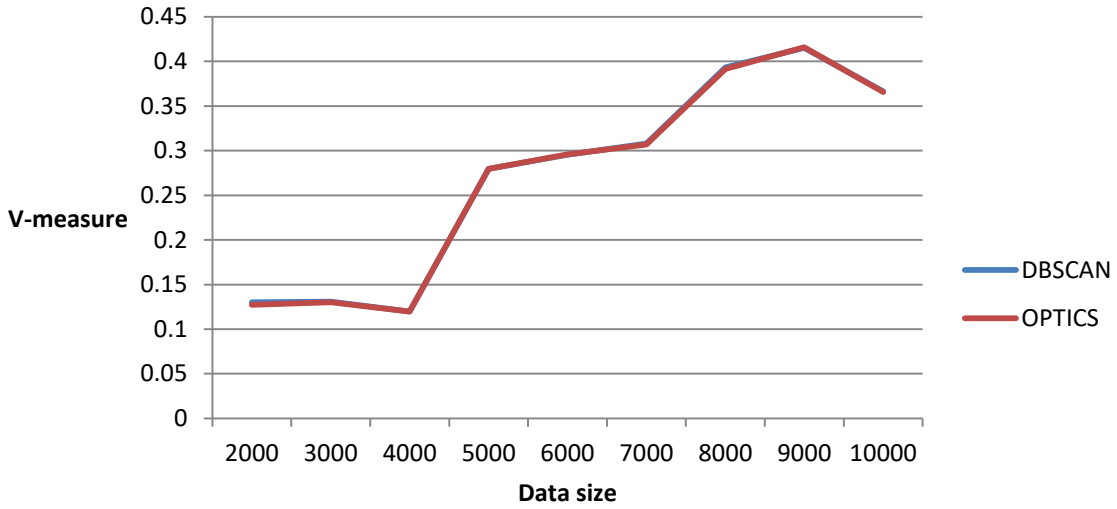

Figure 5.3: V-measure comparison

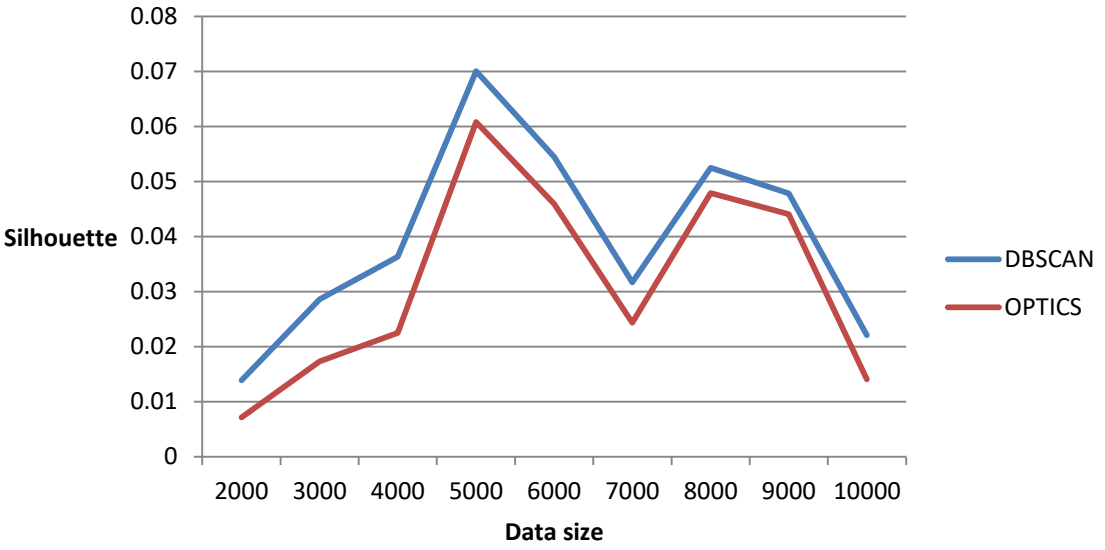

Figure 2.4: Silhouette coefficient comparison

Timing comparison of DBSCAN and OPTICS algorithm is shown in chart below:

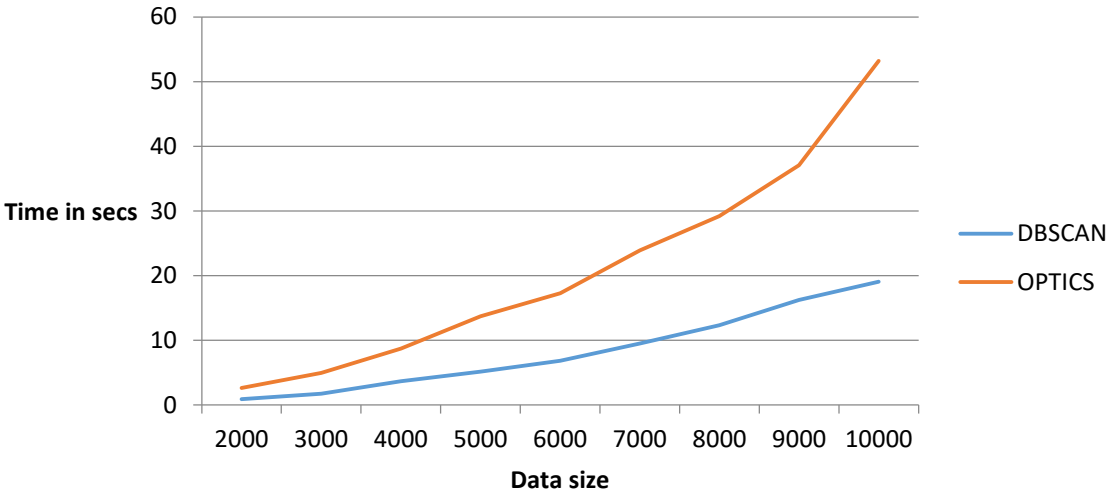

Figure 5.5: Timing comparison of DBSCAN and OPTICS

CHAPTER 6

CONCLUSION AND FUTURE RECOMMENDATION

6.1 Conclusion:

Extensive study has been done in this field for English language but study in clustering documents for Nepali language is still lacking. This thesis is an attempt to close the gap in this area. The summary of cluster quality analysis after applying OPTICS and DBSCAN for the dataset is listed in table 5.1, 5.2, 5.3 and 5.4. Based on the homogeneity, completeness, v-measure and silhouette coefficient DBSCAN performed slightly better than OPTICS algorithm. The time taken by DBSCAN algorithm is also less in compared to OPTICS algorithm.

6.1 Future Recommendation:

In this dissertation the data size has been limited to 10,000 samples. Even after the reduction in dimensionality of the distance matrix using LSI, the time taken for tuning the ε and minPts for the algorithms is huge. Sometimes the virtual memory is consumed due to extensive memory requirement. Future studies can be focused on the speeding up computation time of both the algorithms. Also, the parameters can be further tuned and the LSI dimensions can be increased to test the dataset during clustering.

References

[1]J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 2nd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, ISBN: 978-1-55860-901-3, 1-55860-901-6

[2]C. Aggarwal and C. Zhai, “A survey of text clustering algorithms,” pp. 77–128, Aug. 2012.

[3]C. Sitaula, “Semantic text clustering using enhanced vector space model using nepali language,” International Journal on Natural Language Computing (IJNLC), vol. 3, no. 3, pp. 83–92, 2014.

[4]H. Kriegel, P. Kröger, J. Sander and A. Zimek, "Density-based clustering", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231-240, 2011. Available: 10.1002/widm.30.

[5]A. Huang, “Similarity measures for text document clustering,” Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, pp. 49–56, 2008.

[6]A. Neupane, “Development of nepali character database for character recognition based on clustering,” International Journal of Computer Applications, vol. 107, no. 11, pp. 42– 46, 2014, Full text available.

[7]B. K. Bal and P. Shrestha, “A morphological analyzer and a stemmer for Nepali,” PAN Localization, Working Papers, vol. 2007, pp. 324–31, 2004

[8]E. Ikonomakis, D. Tasoulis and M. Vrahatis, "Density Based Text Clustering", *Lecture Series on Computer and Computational Sciences*, vol. 1, pp. 1-4, 2006.

[9]T. B. Shahi, T. N. Dhamala, and B. Balami, “Support vector machines based part of speech tagging for Nepali text,” International Journal of Computer Applications, vol. 70, no. 24, pp. 38–42, 2013.

[10]T. B. Shahi and A. K. Pant, “Nepali news classification using naïve bayes, support vector machines and neural networks,” in 2018 International Conference on Communication information and Computing Technology (ICCICT), 2018, pp. 1–5. DOI: 10.1109/ ICCICT.2018.8325883o improve retrieval and support browsing

[11]A. Ram, S. Jalal, A. Jalal and M. Kumar, "A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases", International Journal of Computer Applications, vol. 3, no. 6, pp. 1-4, 2010. Available: 10.5120/739-1038.

[12]J.Stefan and H. Kriegel, “Visually Mining Through Cluster Hierarchies”, SIAM DM Conference, 2002.

[13]D. P and S. Roy, "OPTICS on Text Data: Experiments and Test Results", IBM India Research Lab, IIT Delhi, Hauz Khas, New Delhi, India.

[14]K. Santhisree and D. Damodaram, "OPTICS on Sequential Data: Experiments and Test Results", International Journal of Computer Applications, vol. 11, no. 5, pp. 1-4, 2010. Available: 10.5120/1582-2119.

[15]Y. Zhao and B. Shen, "Optimization and Application of OPTICS Algorithm on Text Clustering", Journal of Convergence Information Technology, vol. 8, no. 11, pp. 375-383, 2013. Available: 10.4156/jcit.vol8.issue11.43.

[16]M. Ankerst, M. Breunig, H. Kriegel and J. Sander, "OPTICS", *ACM SIGMOD Record*, vol. 28, no. 2, pp. 49-60, 1999. Available: 10.1145/304181.304187.

[17]A. Paul, B. Syam Purkayastha, and S. Sarkar, “Hidden markov model based part of speech tagging for nepali language,” pp. 149–156, Sep. 2015

[18]A.Dey, A.Paul,and B.S.Purkayastha, “Named entity recognition for nepali language: A semi hybrid approach,” *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 3, no. 8, pp. 21–25, Feb. 2014

[19]S. Sarkar, A. Roy, and B S. Purkayastha, “A comparative analysis of particle swarm optimization and k-means algorithm for text clustering using nepali wordnet,” International Journal on Natural Language Computing, vol. 3, pp. 83–92, Jun. 2014.

[20]W. Zhang, T. Yoshida, and X. Tang, “A comparative study of tf*idf, lsi and multi-words for text classification,” Expert Systems with Applications, vol. 38, no. 3, pp. 2758–2765, Mar. 2011, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2010.08.066

[21]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.

[22]A. Rosenberg and J. Hirschberg, “V-measure: A conditional entropy-based external cluster evaluation measure,” in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, Jan. 2007, pp. 410–420

[23]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, ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[24]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

Appendix

A.1 Stopwords:

अक्सर अगाडि अगाडी अघि अझै अठार अथवा अनि अनुसार अन्तर्गत अन्य अन्यत्र अन्यथा अब अरु अरुलाई अरू अर्को अर्थात अर्थात् अलग अलि अवस्था अहिले आए आएका आएको आज आजको आठ आत्मआदि आदिलाई आफनो आफू आफूलाई आफै आफैँ आफ्नै आफ्नो आयो उ उक्त उदाहरण उनको उनलाई उनले नि उनी उनीहरुको उन्नाइस उप उसको उसलाई उसले हालाई ऊ एउटा एउटै एक एकदम एघार ओठऔ औं कता कति कतै कम कमसेकम कसरि कसरी कसै कसैको कसैलाई कसैले कसैसँग कस्तो कहाँबाट कहिलेकाहीं का काम कारण कि किन किनभने कुन कुनै कुन्नी कुरा कृपया के केहि केही को कोहि कोहिपनि कोही कोहीपनि क्रमशः गएगएको गएरगयौ गरि गरी गरे गरेका गरेको गरेर गरौं गर्छ गर्छन् गर्छु गर्दा गर्दै गर्न गर्नु गर्नुपर्छ गर्ने गैर घर चार चाले चाहनुहुन्छ चाहन्छु चाहिं चाहिए चाहिंले चाहीं चाहेको चाहेर चोटी चौथो चौध छ छन छन् छु छू छैन छैनन् छौ छौं जता जतातै जना जनाको जनालाई जनाले जब जबकि जबकी जसको सबाट जसमा जसरी जसलाई जसले जस्ता जस्तै जस्तो जस्तोसुकै जहाँ जान जाने जाहिर जुन जुनै जे जो जोपनि जोपनी झैं ठाउँमा ठीक ठूलो ततता तत्काल तथा तथापि तथापी तदनुसार तपाइ तपाई तपाईको तब तर तर्फ तल तसरी तापनि तापनी तिन तिनि तिनिहरुलाई तिनी तिनीहरु तिनीहरुको तिनीहरू तिनीहरूको तिनै तिमी तिर तिरको ती तीन तुरन्त तुरुन्त तुरुन्तै तेश्रो तेस्कारण तेस्रो तेह्र तैपनि तैपनी त्यत्तिकै त्यत्तिकैमा त्यस त्यसकारण त्यसको त्यसले त्यसैले त्यसो त्यस्तै त्यस्तो त्यहाँ त्यहिँ त्यही त्यहीं त्यहीं त्यो त्सपछि त्सैले थप थरि थरी थाहा थिए थिएँ थिएन थियो दर्ता दश दिए दिएको दिन दिनुभएको दिनुहुन्छ दुइ दुइवटा दुई देखि देखिन्छ देखियो देखे देखेको देखेर दोश्री दोश्रो दोस्रो द्वारा धन्न धेरै धौ न नगर्नु नगर्नू नजिकै नत्र नत्रभने नभई नभएको नभनेर नयाँ नि निकै निम्ति निम्न निम्नानुसार निर्दिष्ट नै नौ पक्का पक्कै पछाडि डी पछि पछिल्लो पछी पटक पनि पन्ध्र पर्छ पथ्र्यो पर्दैन पर्ने पर्नेमा पर्याप्त पहिले पहिलो पहिल्यै पाँच पांच पाचौँ पाँचौं पिच्छे पूर्व पो प्रति प्रतेक प्रत्यक प्राय प्लस फरक फेरि फेरी बढी बताए बने बरु बाट बारे बाहिर बाहेक बाह्र बिच बिचमा बिरुद्ध बिशेष बिस बीच बीचमा बीस भए भएँ भएका भएकालाई भएको भएन भएर भन भने भनेको भनेर भन् भन्छन् भन्छु भन्दा भन्दै भन्नुभयो भन्ने भन्या भयेन भयो भर भरि

भरी भा भित्र भित्री भीत्र म मध्य मध्ये मलाई मा मात्र मात्रै माथि माथी मुख्य मुनि मुन्तिर मेरो मैले यति यथोचित यदि यद्ध्यपि यद्यपि यस यसका यसको यसपछि यसबाहेक यसमा यसरी यसले यसो यस्तै यस्तो यहाँ यहाँसम्म यही या यी यो र रही रहेका रहेको रहेछ राखे राख्छ राम्रो रुपमा रूप रे लगभग लगायत लाई लाख लागि लागेको ले वटा वरीपरी वा वाट वापत वास्तवमा शायद सक्छ सक्ने सँग संग सँगको सँगसँगै सँगै संगै सङ्ग सङ्गको सट्टा सत्र सधै सबै सबैको सबैलाई समय समेत सम्भव सम्म सय सरह सहित सहितै सही साँच्चै सात साथ साथै सायद सारा सुनेको सुनेर सुरु सुरुको सुरुमै सो सोचेको सोचेर सोही सोह्र स्थित स्पष्ट हजार हरे हरेक हामी हामीले हाम्रा हाम्रो हुँदैन हुन हुनत हुनु हुने हुनेछ हुन् हुन्छ हुन्थ्यो हैन हो होइन होकि होला

A.2 Stemmer rules:

काल

सामान्य भूत

ेँ िस् यो ी ौँ ौं ौ े िन्

सामान्य वर्तमान

छु छस् छ छे छौँ छौं छौ छन् छिन्

सामान्य भविष्यत्

नेछु नेछस् नेछ नेछिन् नेछौँ नेछौं नेछौ नेछन् नेछिन्

अपूर्ण भूत/वर्तमान/भविष्यत्

दै ँदै िरहेको िरहेका िरहेकी

पूर्ण भूत/वर्तमान/भविष्यत्

ेको ेका ेकी

अज्ञात भूत

ेछु ेछौँ ेछौं िछस् ेछौ ेछ ेछन् िछ

अभ्यस्त भूत

थेँ थें थ्यौँ थ्यौं थिस् थ्यौ थ्यो थे थी थिन् नुहुन्थ्यो

वचन

हरू

विभक्ति

ले बाट द्वारा लाई देखि को का कि रो रा री नो ना नी मा

A.4 Parser.py:

```
  def parse_valid_chars(self, value):
        value = Parser.remove_zwnj(value)
         value = Parser.remove_zwj(value)
        return EMPTY_STRING.join(c if c in  self.whitelist else SPACE for c in value)
```

A.5 Tokenizer.py

```
def tokenize(self, value):
         value = self.parser.parse_valid_chars(value)
        tokens = value.split(SPACE)     # remove empty elements from tokens
        return list(filter(None, tokens))
```

A.6 Stopwords_remover.py

```
        def remove_stopwords(df):    //df represents a dataframe
        stop_words_txt = 'Data/StopWords.txt'
        stop_words_file = open(stop_words_txt, 'r', encoding='utf-8')
        stop_words = stop_words_file.read().splitlines()
        print('removing stop words')
```

```
        df.data = df.data.map(lambda tokens:  [t for t in tokens if len(t) > 2 and t not in
        stop_words])

        print('stop words removed from data')

         return df
```

A.7 Stemmer.py

```
class Stemmer:
    def __init__(self):
        rule_file_name = 'Data/StemmerRules.txt'
        rule_file = open(rule_file_name, 'r', encoding='utf-8')
        self.rules = rule_file.read().split('\n')
        # remove blank lines and comments
        self.rules = [x for x in [y.lstrip() for y in self.rules]
                if len(x) > 0 and not x.startswith('#')]
        # split lines to individual rules
        self.rules = [x for row in self.rules for x in row.split(' ')]
        self.rules.sort(key=len, reverse=True)

    def remove_suffix(self, word, suffix):
        if not word.endswith(suffix):
            return word
        return word[:len(word) - len(suffix)]

    def find_root(self, word):
        for suffix in self.rules:
            word = self.remove_suffix(word, suffix)
        return word
```

```
if __name__ == '__main__':
    s = Stemmer()
    print(s.rules)
```

A.6 Timing analysis function

```
def calculatetime():

        data_sizes = list(range(2000, 10_001, 1000))

        for data_size in data_sizes:

            df = Analysis.read_data(data_size)

            true_k = len(df.category.unique())

            X = Analysis.tfidf(df)

            X = Analysis.lsi(X)

            models = [Analysis.dbscan_model(eps=0.55, minPts=10),

                    Analysis.optics_model(eps=0.55, minPts=10))]

            labels = df.target

            for model in models:

                t0 = time()

                model.fit(X)
```

```
        duration = time() - t0
        print(data_size, duration)
```

A.7 Cluster analysis function:

```
def clusteranalysis():
    data_sizes = list(range(2000, 10001, 1000))
    for data_size in data_sizes:
      df = Analysis.read_data(data_size)
      true_k = len(df.category.unique())
      X = Analysis.tfidf(df)
      X = Analysis.lsi(X)
      models = [Analysis.dbscan_model(eps=0.55, minPts=10),
            Analysis.optics_model(eps=0.55, minPts=10))]
      labels = df.target
      for model in models:
        model.fit(X)
        homogeneity = metrics.homogeneity_score(labels, model.labels_)
        completeness = metrics.completeness_score(labels, model.labels_)
```

```
v_measure = metrics.v_measure_score(labels, model.labels_)

silhouette = metrics.silhouette_score(X, model.labels_)

print(data_size, homogeneity, completeness, v_measure, silhouette)
```