Dissertation

Data Preprocessing for Improving Cluster Analysis and Its Application to Short Text Data

Graduate School of
Natural Science & Technology
Kanazawa University

Division of Electrical Engineering
and Computer Science

Student ID No.: 1223112012
Name: Tran Vu Anh
Chief advisor: Professor Kenji Satou

Date of Submission: July 3rd, 2015

Table of contents

CHAPTER I Dissertation introduction ... 1

1.1 Dissertation overview ... 1

1.2 Dissertation distribution ... 2

CHAPTER II Clustering and data preprocessing for clustering ... 3

2.1 Clustering ... 3

2.1.1 Background of clustering ... 3

2.1.2 Challenges in clustering ... 4

2.2 Data preprocessing ... 5

CHAPTER III Data preprocessing algorithm D-IMPACT ... 7

3.1 Gravity-based data preprocessing algorithm ... 7

3.2 Clustering algorithm IMPACT ... 8

3.2.1 IMPACT algorithm ... 8

3.2.2 Experiment result ... 15

3.3 Data preprocessing algorithm D-IMPACT ... 19

3.3.1 Movement of data points ... 20

3.3.1.1 Density ... 20

3.3.1.2 Attraction ... 21

3.3.1.3 Inertia value ... 22

3.3.1.4 Data point movement ... 22

3.3.2 D-IMPACT algorithm ... 23

3.3.2.1 Noisy points and outlier detection ... 24

3.3.2.2 Moving data points ... 25

3.3.2.3 Complexity ... 25

3.4 Experiment result ... 27

3.4.1 Datasets and method ... 27

3.4.1.1 Two-dimensional datasets ... 27

3.4.1.2 Practical datasets ... 28

3.4.1.3 Validating methods ... 29

3.4.2 Experiment results of 2D datasets 30
3.4.3 Experiment results of practical datasets 33
3.4.3.1 Iris, Wine, and GSE9712 datasets 33
3.4.3.2 Water treatment plant and Lung cancer datasets 238
3.5 Conclusion 40
CHAPTER IV Data preprocessing algorithm SCF 41
4.1 Clustering algorithms and data preprocessing methods for text clustering 41
4.1.1 Text clustering 41
4.1.2 Challenges of text clustering in short text data 42
4.1.3 Data preprocessing for text clustering 44
4.2 WordNet and semantic similarity 46
4.2.1 WordNet structure 46
4.2.2 Semantic similarity 48
4.3 Data preprocessing algorithm SCF 49
4.3.1 Phase 1: Word pruning and clustering. 50
4.3.2 Phase 2: keywords and Semantic related Conceptual Feature (SCF) matrix construction. 55
4.4 Experiment result 57
4.4.1 Datasets and text processing 57
4.4.2 Experiment result. 60
4.5 Conclusion. 62
CHAPTER V Conclusion 63
Supplementations 64
Bibliography 65

List of figures

Figure 1.1 A simple example of clustering. 1
Figure 2.1 An example of dendrogram. 4
Figure 2.2 Illustration of the effect of noise removal 6
Figure 3.1 Clusters are fractured after shrinking. 8
Figure 3.2 Flowchart of the IMPACT algorithm 9
Figure 3.3 Effect of denoising 10
Figure 3.4 Pseudo code of moving data points 14
Figure 3.5 Pseudo code of cluster identification 15
Figure 3.6 Illustrations of datasets from DS1 to DS8 17
Figure 3.7 Clustering results for DS1, DS2, DS3, DS4, DS6, and DS7 using IMPACT algorithm 18
Figure 3.8 Clustering results for DS8 using IMPACT 18
Figure 3.9 Outline of the D-IMPACT algorithm 23
Figure 3.10 An illustration of noisy points and outliers.. 24
Figure 3.11 Illustration of the effect of noise removal in D-IMPACT 25
Figure 3.12 Processing times of D-IMPACT and CLUES on test datasets. 26
Figure 3.13 Visualizations of 2D datasets. 27
Figure 3.14 Visualization of the dataset DM130 preprocessed by D-IMPACT and CLUES 31
Figure 3.15 Visualization of the dataset MultiCL preprocessed by D-IMPACT and CLUES 31
Figure 3.16 Visualization of two datasets t4.8k and t8.8k preprocessed by D-IMPACT 32
Figure 3.17 Visualizations of the dataset t4.8k preprocessed by CLUES using different values of *k* based on the size of the dataset.. 32
Figure 3.18 Visualization of the dataset Planet preprocessed by D-IMPACT and CLUES 33
Figure 3.19 Visualization of the Iris dataset before and after preprocessing by D-IMPACT. Visualization of the original dataset is shown in the bottom-left triangle.

Visualization of the dataset optimized by D-IMPACT is shown in the top-right triangle. 35
Figure 3.20 Visualization of the first four features of the Wine dataset before and after preprocessing by D-IMPACT. Visualization of the original dataset is shown in the bottom-left triangle. Visualization of the dataset preprocessed by D-IMPACT is shown in the top-right triangle. 36
Figure 3.21 Visualization of the Iris and Wine datasets preprocessed by CLUES. 37
Figure 3.22 Dendrograms of the clustering results on the WTP dataset. 39
Figure 4.1 Clustering search results for the word "Kanazawa". 42
Figure 4.2 The rapid increasing of the number of dimensions and sparseness when the number of texts increases. 43
Figure 4.3 D-IMPACT algorithm degenerated the performance of clustering on text data. 45
Figure 4.4 An illustration of WordNet's structure. 47
Figure 4.5 Flowchart of SCF algorithm. 50
Figure 4.6 Algorithm for removing words which have extreme high document-frequency. 52
Figure 4.7 Clustering algorithm for clustering words based on semantic similarity. 54
Figure 4.8 Algorithm for identifying keywords in each-document and selecting important representative words. 56
Figure 4.9 Histogram of semantic similarity between verb-verb using first k (k= 2 at the left, k = 3 at the right) concepts 59
Figure 4.10 Histogram of semantic similarity between noun-noun using first k (k= 2 at the left, k = 3 at the right) concepts. 59
Figure 4.11 Histogram of values of covariance matrix between important representative words in Enron dataset. 60
Figure 4.12 Comparison of clustering performances on UC Berkeley Enron and 20 newsgroups datasets. 62

List of tables

Table 3.1 Experiment datasets for IMPACT algorithm. 16
Table 3.2 Experiment datasets for D-IMPACT algorithm. 29
Table 3.3 Parameter sets of D-Impact for experiments.. 30
Table 3.4 The Index scores of clustering results using HAC on the original and preprocessed datasets of IRIS and Wine. The best scores are in bold 37
Table 3.5 Index scores of clustering results using k-means on original and preprocessed datasets of IRIS and Wine. The best scores are in bold.. 37
Table 3.6 Accuracy and precision values of noise and outlier detection on the Lung-cancer dataset 40
Table 4.1 Characteristics of UC Berkeley Enron and 20 newsgroups datasets.................. 58
Table 4.2 Result of feature reduction by word clustering and SCF algorithm. 61

Abbreviations

PCA = Principal Component Analysis	
HAC = Hierarchical Agglomerative Clustering	
RI = Rand Index	
aRI = adjusted Rand Index	
LSI = Latent Semantic Indexing	
WSD = Word Sense Disambiguation	

Acknowledgements

I want to express my gratitude to Prof. Kenji Satou for supervising this dissertation, and for numerous fruitful, serious, and enthusiastic discussions that initiated or improved many of the presented ideas and results. I am indebted to his guidance, patience, and mentorship.
I wish to thank Kanazawa University for providing me this scholarship. It created a change for me to explore and enjoy the life in Japan.
Finally I would like to thank all of my friends and my whole family, who supported me during the time I get my PhD. in Kanazawa University.

Abstract

Clustering is a process that divides data into groups (clusters) whose memberships are similar to each other than data objects belong to other groups This task is useful for manage, summarize, and understand the patterns underlying the data. Although it has a long history of development, there remain open problems, such as how to determine the number of clusters, the difficulty in identifying arbitrary shapes of clusters, and the curse of dimensionality. Preprocessing methods can help to solve those problems and hence improve the quality of clustering. In this study, we propose two data preprocessing algorithms called D-IMPACT and SCF algorithms. D-IMPACT algorithm has two phases. The first phase detects noisy and outlier data points based on the density, and then removes them. The second phase separates clusters by iteratively moving data points based on attraction and density. Our second work, the data preprocessing algorithm SCF, aims to reduce the number of dimensions without losing the semantic information stored in each feature for short text data. SCF algorithm has two phases: the first phase is doing pruning on to remove unnecessary words and replace semantically related words by a representative word. In the second phase, SCF algorithm transforms the data matrix into Semantic similarity Conceptual Feature (SCF) space, which presents the semantic similarity between keywords of each document and the concept underlying all the documents. Our experiment results show that D-IMPACT and SCF algorithms are able to improve the performance of the clustering algorithms performed on datasets preprocessed by them.

Keyword*:* data preprocessing, clustering, data point movement, feature reduction, semantic similarity.

CHAPTER I

Dissertation introduction

1.1 Dissertation overview

Clustering is a process that divides data into groups (clusters) whose memberships are similar to each other than data objects belong to other groups (**Figure 1.1**). Simply, clustering is the task that data objects are divided into groups whose memberships are similar to each other than data objects belong to other groups. Clustering is applied in many fields, such as: decision-making, machine-learning situations, document retrieval, image segmentation, bioinformatics, and finance. The discovered clusters can be used to explain the characteristics of underlying data, or server as the foundation for other data analysis techniques.

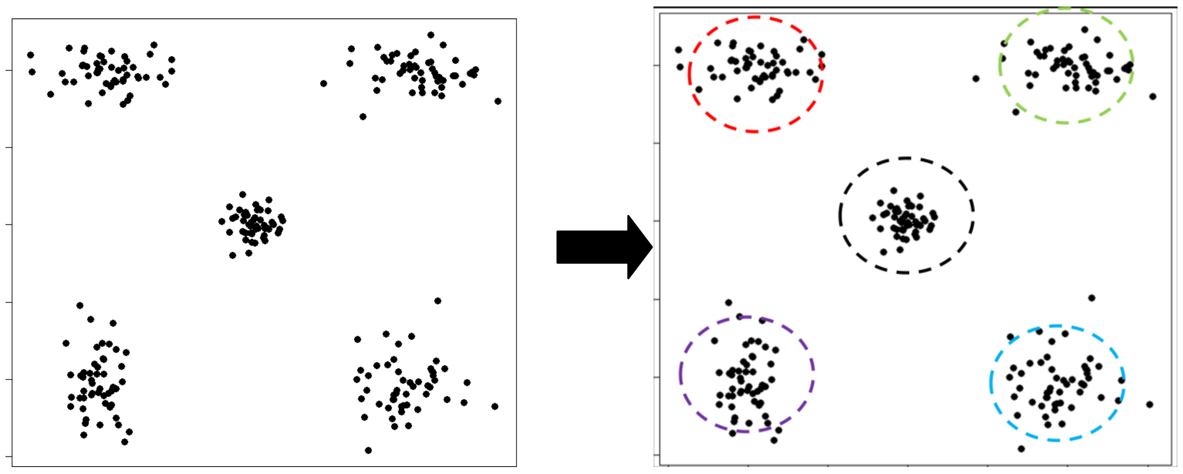


Figure 1.1 A simple example of clustering.

Although it has a long history of development, there remain open problems, such as how to determine the number of clusters, the difficulty in identifying arbitrary shapes of clusters, and the curse of dimensionality [1]. The majority of current algorithms perform well for only certain types of data [2]. Therefore, it is not easy to specify the algorithm and input parameters required to achieve the best result. In addition, it is difficult to evaluate the clustering performance, since most of the clustering validation indexes are specified for certain clustering objectives [3]. Finding an appropriate algorithm and parameters is very difficult and requires a sufficient amount of experiment results. The datasets measured from real systems usually contain outliers and noise, and are, there-fore, often unreliable [4] [5]. Such datasets can impact the quality of cluster analysis. However, if the data have been preprocessed appropriately, for example, clusters are well-separated, dense and have no noise, the performance of the clustering algorithms may improve.

In this study, we propose two data preprocessing algorithms called D-IMPACT and SCF algorithm. D-IMPACT algorithm iteratively moves data points based on attraction and density to detect and remove noise and outliers, and separate clusters. SCF algorithm reduces the number of dimension and improves the quality of term frequency matrix based on semantic similarity and clustering. Our experiment results show that these methods are able to produce new datasets such that the performance of the clustering algorithm is improved.

1.2 Dissertation distribution

Chapter I is to present our research and contribution. **Chapter II** briefly presents the background and challenges of clustering, then introduce the data preprocessing to overcome the challenges in clustering. **Chapter III** introduces our first data preprocessing algorithm D-IMPACT, which focuses on de-noising and separating clusters. C**hapter IV** presents SCF algorithm, a semantic-based data preprocessing method, to reduce the number of feature and improve the content presented in term frequency matrix. The dissertation is concluded in **Chapter V**.

CHAPTER II
Clustering and data preprocessing for clustering

This chapter briefly presents the background of clustering and its challenges. We then introduce data preprocessing methods in order to deal with challenges in clustering.

2.1 Clustering

2.1.1 Background of clustering

As introduced above, clustering task organizes data objects into groups whose members are similar in some way. A cluster is therefore a collection of objects which are similar between them and are dissimilar to the objects belonging to other clusters. Clustering is applied in various fields, e.g., marketing (categorizes the customer), biology (classify the gene expression data), geography (identify the similar zones appropriate for exploitation) and so on.

Clustering has more than 50 years of development. Many clustering algorithms were proposed with different schemas and concepts [1]. The taxonomy of clustering techniques is not unique due to the different viewpoints of comparisons. Some algorithms are the combination of different techniques and concepts, therefore they can be classified to various classes. We only introduce common representative algorithms for each clustering technique's class.

Partitioning clustering algorithm These clustering techniques attempt to break a dataset into *k* clusters by optimizing a given criterion. They usually repeat the iteration of finding the centroid of each cluster and assigning points to the centroids until the criterion is assumed maximized. *k*-means, *k*-medoids [1], and PAM [6] are simple examples of partitioning clustering algorithms.

Hierarchical clustering algorithm Hierarchical clustering algorithms start with each data point belonging to one of the disjoint clusters, then merge the two most similar clusters, or vice versa, start with the whole dataset then divide them to two most different cluster. Those processes continue until stop conditions are satisfied. The clustering result is outputted as a dendrogram (**Figure 2.1**). Some well-known algorithms for this class are Hierarchical Clustering (HC) [1] and CURE [7].

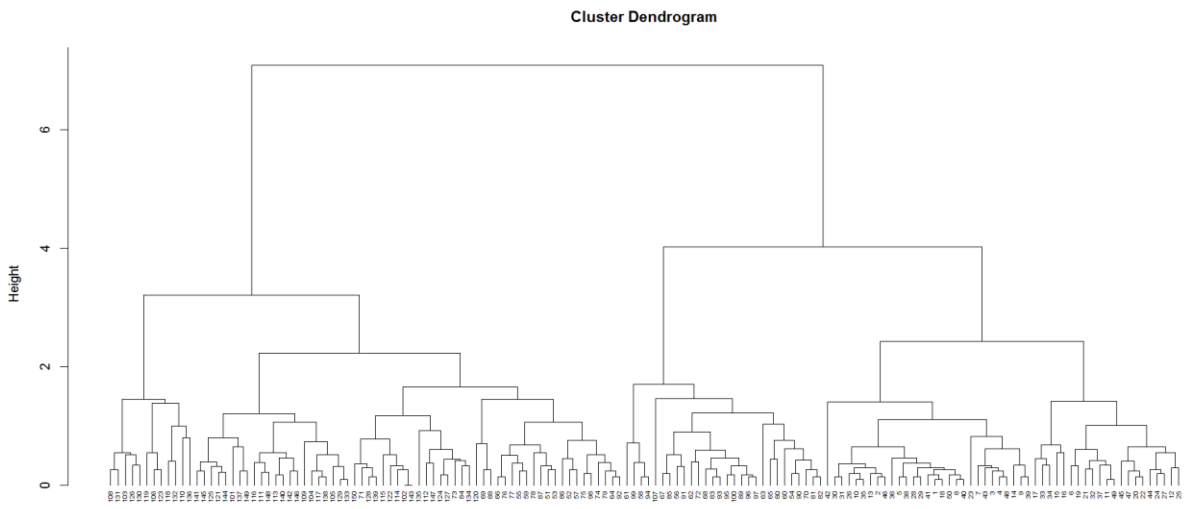


Figure 2.1 An example of dendrogram.

Density based clustering algorithm Density-based clustering algorithms attempt to find dense regions separated from other regions that satisfy certain criteria related to density. DBSCAN [8], the most popular Density based clustering algorithm, scans and finds all possible regions such that the size of the region is larger than *minPts* within the *Esp* radian. The *minPts* and *Esp* parameter then become two specific characteristics for the algorithms belong to this class.

Grid based clustering algorithm Grid-based clustering algorithms limit the search space into segments (e.g., cubes, cells, and regions) according to attribute space. This type of clustering algorithm is proposed with the hope to get rid of the curse of dimensionality problem. CLIQUE [9], STING [10] are clustering algorithm feature this type of Grid-based clustering algorithms.

2.1.2 Challenges in clustering

Even though with a long history of research and development, there are still several challenges existed for clustering:

The number of clusters. Most of clustering algorithms require a priori specification number of clusters. Others require a specific threshold or rely on a criterion to determine the number of clusters.

High dimensionality. The different between objects belong to same clusters and ones belong to other clusters decease as the increasing of the number of dimensions. This problem makes similarity function loss its usefulness on discriminating data objects.

Clustering validation. Various criteria or indexes are employed to validate the clustering result. However, since there is no "best" criterion or index, the choice of volition function can highly affect the clustering result.

Noise and outlier. The datasets measured from real systems usually contain outliers and noise, and are, there-fore, often unreliable. Such datasets can impact the quality of cluster analysis.

Such problems can impact the quality of cluster analysis. However, if the data have been preprocessed appropriately, for example, clusters are well-separated, dense and have no noise, the performance of the clustering algorithms may improve. Data preprocessing is often used to perform such tasks.

2.2 Data preprocessing

Real world data usually contain noises and outliers, are high dimensional, hence, strongly impact the performance of clustering. To deal with such problems, data preprocessing methods are employed to improve quality of data and therefore, improve the performance of clustering. The popular tasks of data preprocessing methods in clustering are:

Feature reduction. These methods represent the input space into a lower-space but still retain the variance of the data as possible. Principal Component Analysis (PCA) [11] is a well-known example for feature reduction methods. The concept of PCA is finding new principal components, which linear with variance of data, to reduce the number of dimension with minimal loss of information. Although PCA accounts for as much variance of the data as possible, clustering algorithms combined with PCA do not necessarily improve, and, in fact, often degrade, the cluster quality [12]. PCA essentially performs a linear transformation of the data based on the Euclidean distance between samples; thus, it cannot characterize an underlying nonlinear subspace.

Feature selection. Feature reduction methods represent the input space into a new space, and hence, cause the loss of features in original space. In contrast to feature reduction, feature selection methods try to select a subset of features in original space such that the clustering performed on this space can be improved. Many feature selection methods for clustering are summarized in [13].

Noise and outlier removal. Noises and outliers greatly affect the performance of clustering. Several clustering algorithms, i.e., single linkage hierarchal clustering, often miss-clusters outliers as clusters. Noises reduce the intercluster similarity, hence make clusters becomes not well-separated. A lot of methods were proposed in order to identify and remove noises and outliers to make the identification of clusters easier (**Figure 2.2**). A number of outlier removal methods are summarized in [14].

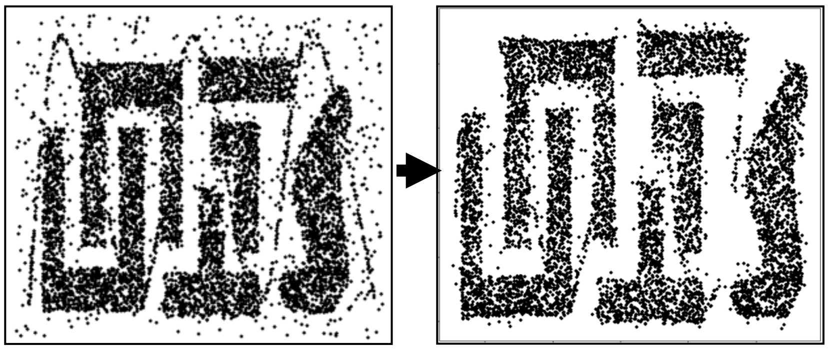


Figure 2.2 Illustration of the effect of noise removal

In this chapter, we introduce the background of clustering and data preprocessing for clustering. Several clustering and data preprocessing methods were introduced. In the next chapter, we will present our data preprocessing algorithm D-IMPACT in detailed.

CHAPTER III

Data preprocessing algorithm D-IMPACT

In this chapter, we describe the data preprocessing algorithm D-IMPACT based on concepts underlying the clustering algorithm IMPACT [15]. We aim to improve the accuracy and flexibility of the movement of data points in the IMPACT algorithm by applying the concept of density to various affinity functions. These improvements will be described in the subsequent subsections.

3.1 Gravity-based data preprocessing algorithm

Recent studies have focused on new categories of clustering algorithms which prioritize the application of data preprocessing. SHRINK, a data shrinking process, moves data points along the gradient of the density, generating condensed and widely separated clusters [16]. Following data shrinking, clusters are detected by finding the connected components of dense cells. The data shrinking and cluster detection steps are conducted on a sequence of grids with different cell sizes. The clusters detected at these cells are compared using a cluster-wise evaluation measurement, and the best clusters are then selected as the final result. In CLUES [17], each data point is transformed such that it moves a specific distance toward the center of a cluster. The direction and the associated size of each movement are determined by the median of the data point's *k* nearest neighbors. This process is repeated until a pre-defined convergence criterion is satisfied. The optimal number of neighbors is determined through optimization of commonly used index functions to evaluate the clustering result generated by the algorithm. The number of clusters and the final partition are determined automatically without any input parameters, apart from the convergence stop criteria.

These two shrinking algorithms share the following limitations:

- The process of shifting toward the median of neighbors can easily fracture the cluster (**Figure 3.1**).
- The direction of the movement vector is not appropriate in specific cases. For example, if the clusters are adjacent and differ highly in density, the median of the neighbors is likely to be located on another cluster.

We introduce IMPACT, a clustering algorithm based on the simulation of gravity system: moving data points under effect of attractive-force like values to form dense regions that can be easily identified as clusters. The data points movement in IMPACT algorithm can avoid the addressed problems. The next section will explain the algorithm in detailed.

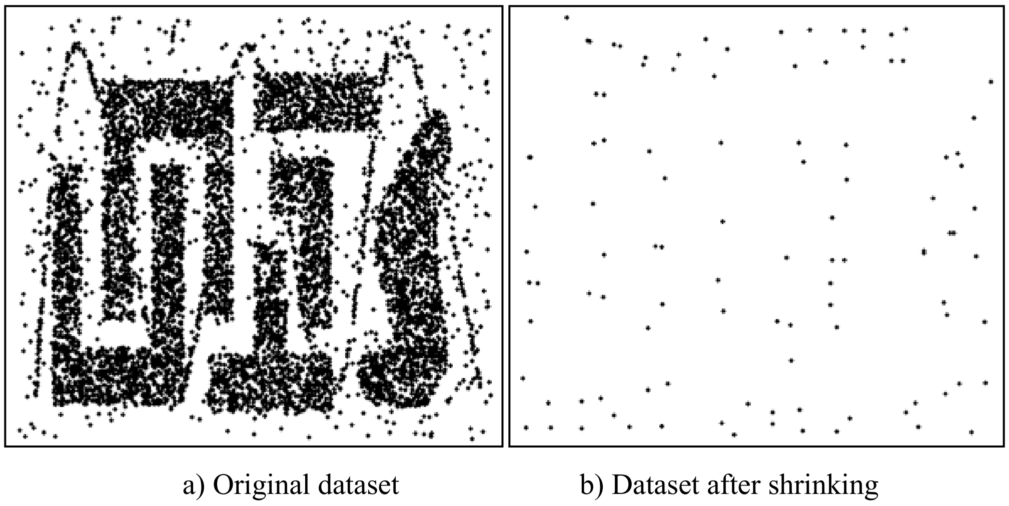


Figure 3.1 Clusters are fractured after shrinking.

3.2 Clustering algorithm IMPACT

3.2.1 IMPACT algorithm

The IMPACT algorithm is based on the idea of gradually moving all objects closer to similar objects according to the attraction between them until the dataset becomes self-partitioned. The algorithm has two phases. The first phase is for normalizing and denoising the input dataset. In the second phase, IMPACT iteratively moves the data points and identifies clusters. The flowchart of IMPACT algorithm is described in **Figure 3.2**.

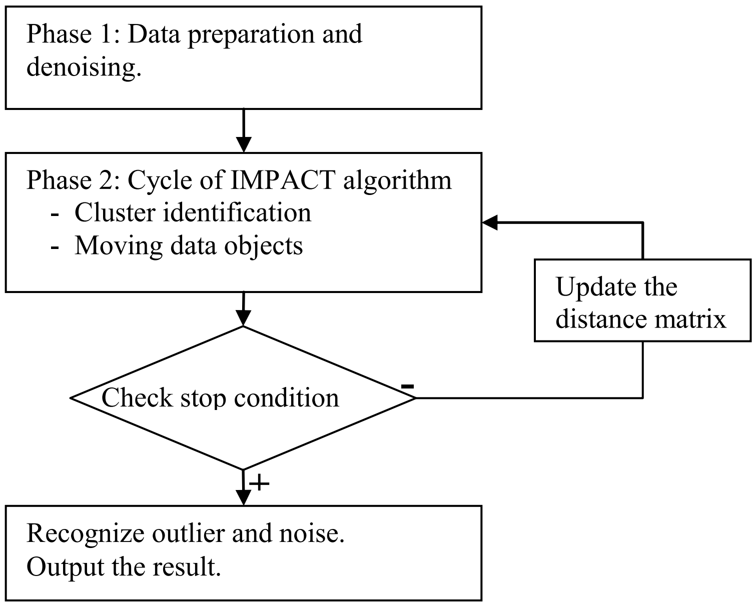

Phase 1: Data preparation and denoising.
Phase 2: Cycle of IMPACT algorithm
- Cluster identification
- Moving data objects
Update the distance matrix
Check stop condition
-
+
Recognize outlier and noise. Output the result.

Figure 3.2 Flowchart of the IMPACT algorithm

A) Phase 1: Normalizing and denoising the input dataset.

The first step in this phase is to read the input data and normalize the numerical attribute values into the range [0,1]. The objective of this process is to avoid attributes with a wide range of values dominating the clustering results. Each value in the dataset is modified as

$$x_{ij} = \frac{x_{ij} - \min\limits_{r=1..m}(x_{rj})}{\max\limits_{r=1..m}(x_{rj}) - \min\limits_{r=1..m}(x_{rj})}.$$

The distance matrix is computed from this normalized dataset. The threshold *Th* is then computed from the maximum value of the distance matrix (i.e., longest distance).
The second step of this phase is denoising. Since we identify clusters only by grouping data objects according to the threshold *Th*, in noisy datasets, clusters linking at the border region can affect the recognition of clusters. However, if we simply move the data objects, noise might be reduced and the border regions become clearer, as the points move closer to their centroids and the gaps between clusters widen. The denoising step is controlled by a denoise-level parameter, which is the number of steps of moving data objects. The noisier a dataset is, the bigger this value should be. **Figure 3.3** shows the effect of denoising.

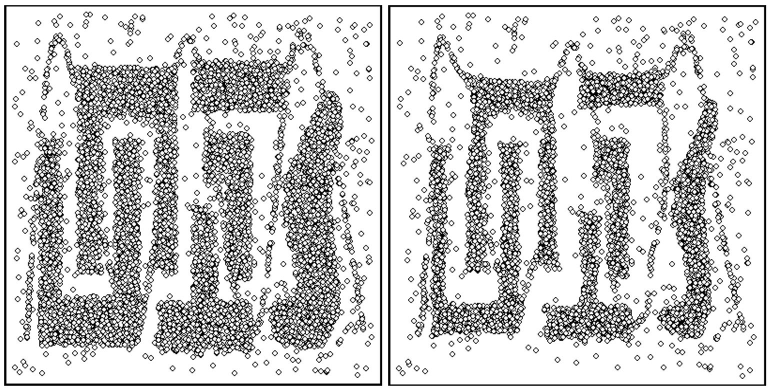

a) Before denoising b) After denoising

Figure 3.3 Effect of denoising

B) Phase 2: Repeating the cycle of identifying clusters and moving data objects until the stop condition is satisfied.

Moving data objects can improve the quality of identified clusters by increasing the similarities between similar data objects and the dissimilarities between clusters. Firstly, we describe how to compute the movement of a data object (*movement vector*).

Given a dataset $D = \{x \mid x \in R^n\}$ with *m* data objects (vectors), our objective is to group *m* data objects into clusters without specifying their number. We assume that each data object is attracted by others via a natural force called attraction as in a physical system. There are three steps to compute the movement vector of a data object:

Computing the attraction and attractive vector between all data objects

As in physics, objects attract each other and move closer under the effect of an attractive force among them (attraction).

Attraction *Attraction* is a quantity that represents the attractive force between two data objects x_i and x_j:

$$A_{ij} = \text{attraction}(x_i, x_j) = \frac{1}{\text{distance}(x_i, x_j)^p},$$

where p ($p > 0$) is a user specified parameter used to adjust the effect of attraction between two data objects.

Attractive vector *Attractive vector* is an *n*-dimensional vector representing the attractive force between a data object and another data object caused by the attraction between them. Attractive vector $av_{ij} = (av_{ij1}, av_{ij2} \dots av_{ijn})$ of x_i for x_j is computed as

$$av_{ijk} = \frac{x_{jk} - x_{ik}}{\sum_{r=1}^{n} | x_{jr} - x_{ir} |} \times A_{ij} \quad (k = 1..n)$$

Computing the movement vector for each data object

The attractive forces shift objects, as represented by movement vector. The direction of the movement vector v_i of x_i is the summation of all attractive vectors of all other data objects to x_i:

$$v_i = \sum_{j=1}^{m} av_{ij}$$

Computing the *Inertia* of each cluster and the *Scale* value for each movement vector.

The length of the movement vector should be calculated carefully. For the sake of higher clustering accuracy, the distance of movement (magnitude of the movement vector) should not be too long. However, if the distance of the movement is too short, the clustering process will be slow. In addition, after data objects form a cluster, they do not need to move so much. Based on these considerations, the movement is adjusted by two values:

Inertia If x_i belongs to a cluster C_j, its movement vector v_i is adjusted as

$$v_i = v_i \times (1 - I_j),$$

where I_j is the *Inertia* of cluster C_j. The *Inertia* avoids clusters from moving too quickly and incorrectly merging.

Scale Because the threshold value *Th* is used during the clustering step, the appropriate magnitude of each movement vector should be no greater than *Th*. Scale, a value used to adjust the length of movement vector, is computed as

$$Scale = \frac{Th}{\max_{i=1..m}(| \overrightarrow{v_i} |)}.$$

After adjustment, all movement vectors are guaranteed not to cross the scanning field of the nearest objects. It is therefore clear that the movement of data objects retains the global and local structure of the cluster: the *Inertia* ensures clusters do not merge easily, while computation of all the attractions affecting one data object retains the global balance.

Finally, data objects are moved as

$$x_i = x_i + Scale \times v_i \,.$$

The movement increases the similarity between close objects and the dissimilarity between groups by increasing the distance of their borders. **Figure 3.4** summarizes the steps to move data objects.

Next, clusters are identified. Cluster identification is the process by which indistinguishable data objects are grouped together. The pseudo code in **Figure 3.5** presents the steps in this process. If the distance between two objects is less than *Th*, they are linked and form a group. The threshold *Th* used in the grouping step is computed as

$$Th = q \times maxDistance \,,$$

$$maxDistance = \max(\mathrm{distance}(x_i, x_j)) \ \forall x_i, x_j \,,$$

where q is a parameter specified by the user to compute *Th*, the threshold value to determine whether two data objects are indistinguishable. For example, if $q = 0.05$, we can say that two data objects are indistinguishable if their difference is 5% less than the distance between the most different pair. Although all data objects are assigned to groups, not all groups can be considered as clusters. A group G is a cluster if it satisfies the condition

$$|G| \geq min\_ClusterSize \,,$$

where *min_ClusterSize* is a threshold used to eliminate small groups.

The iterative process stops when it meets the stop condition. The stop condition of the IMPACT algorithm can be satisfied in many ways, and not just when all data objects are clustered. Below are common stop conditions that are used for different objectives.

A given percentage of data objects have been clustered When all or most data objects are clustered, we can stop the cycle and deal with unclustered objects later.

The magnitude of the longest movement vector is sufficiently small (e.g., less than *Th* or a user specified parameter) Data objects in dense regions are usually clustered quickly, while noisy objects and outliers are not attracted greatly by clusters.

This concept is employed by IMPACT to detect outliers and noise effectively. After detecting all clusters, outliers, and noise, the final clustering result is output.

Hence, we can see that the IMPACT algorithm works by iterating the grouping and moving of data objects until the dataset is self-partitioned. To evaluate the performance of IMPACT algorithm, we tested it on datasets with different characteristics. The results will be presented in the next section.

```
Input
x: data points
p: input parameter
Output:
x: data points after moving
Algorithm:
compute attraction matrix:
        $A_{ij} = \text{attraction}(x_i, x_j) = \frac{1}{\text{distance}(x_i, x_j)^p}$
compute attractive vectors:
        $av_{ijk} = \frac{x_{jk} - x_{ik}}{\sum_{r=1}^{n} |x_{jr} - x_{ir}|} \times A_{ij} \quad (k = 1.n)$
        $av_{ij} = (av_{ij1}, av_{ij2} \ldots av_{ijn})$
compute movement vector:
        $v_i = \sum_{j=1}^{m} av_{ij}$
compute Inertia for each cluster:
        $I_j = \frac{|C_j|}{largestClusterSize}$
for each movement vector $x_i$
    if $x_i \in C_j$ then
        adjust $x_i$'s movement vector as:
            $x_i \in C_j \Rightarrow v_i = v_i \times (1 - I_j)$
        compute the Scale value:
            $Scale = \frac{Th}{\max_{i=1..m}(|\overrightarrow{v_i}|)}$
    end if
end for
move all data points:
        $x_i = x_i + Scale \times v_i$
end;
```

Figure 3.4 Pseudo code of moving data points

```
Input
$x$: data points
$Th$, $min\_ClusterSize$: input parameter
Output:
$C$: set of clusters
Algorithm:
$l$ = 0;   $V$ = ∅;
for each $x_i \notin V$ then
   $S = x_i$;
   $G$ = ∅;
   while $S \neq \emptyset$ do
      Randomly take $x_z$ out of $S$;
      $G = G \cup \{x_z\}$
      $V = V \cup \{x_z\}$
      for each $x_j \notin V$ do
         if distance($x_z$,$x_j$)<$Th$
            $S = S \cup \{x_j\}$
         end if
      end for
   end while
   if $|G| \geq min\_ClusterSize$ then
      $l$++;
      $C_l = G$
   end if
end for
```

Figure 3.5 Pseudo code of cluster identification

3.2.2 Experiment result

In this section, we evaluate the performance of IMPACT and demonstrate its effectiveness for different types of data distributions. We use six synthetic datasets, two datasets used in the paper presenting the Chameleon algorithm, two datasets from the UCI Machine Learning Repository, and one text dataset. We firstly introduce these eleven datasets used in this experiment.

The six synthetic datasets are denoted DS1 to DS5, and DS8. DS1 and DS2 are hierarchical datasets. DS3 and DS4 contain clusters with different densities and sizes. DS5 includes many disjointed clusters (142 clusters) to demonstrate that IMPACT works well with a large number of clusters. DS6 and DS7 [18] are extremely difficult to cluster: they contain clusters with different shapes, noise and outliers. DS6 has a chain connecting all clusters (i.e., the single-link effect), while DS7 contains clusters with different arbitrary shapes. The gaps between clusters are small and filled with noise. DS8 is a simple hard clustering dataset that includes 100 points generated randomly, and three 10 point clusters placed in three corners of the dataset. DS8 does not have any "natural" clusters, so the clustering results could differ depending on the cluster validity. DS8 is suitable for testing the validity of the clustering algorithm. **Figure 3.6** presents all datasets from DS1 to DS8 in two-dimensional plots. Wine and Iris are commonly used datasets taken from the UCI Machine Learning Repository [19]. R8- [20] is a sub collection of the Reuters-21578 dataset. These datasets characterize different problems in clustering. **Table 3.1** gives the sizes of all datasets and the numbers of desired clusters (the correct clustering results) for them.

Table 3.1 Experiment datasets for IMPACT algorithm

Dataset	Size of datasets	Number of clusters
DS1	250	2
DS2	800	3
DS3	1934	4
DS4	4343	6
DS5	8026	142
DS6	8000	6
DS7	8000	8
DS8	130	3
Iris	150 (four features)	3
Wine	178 (13 features)	3
R8-	445 documents	8

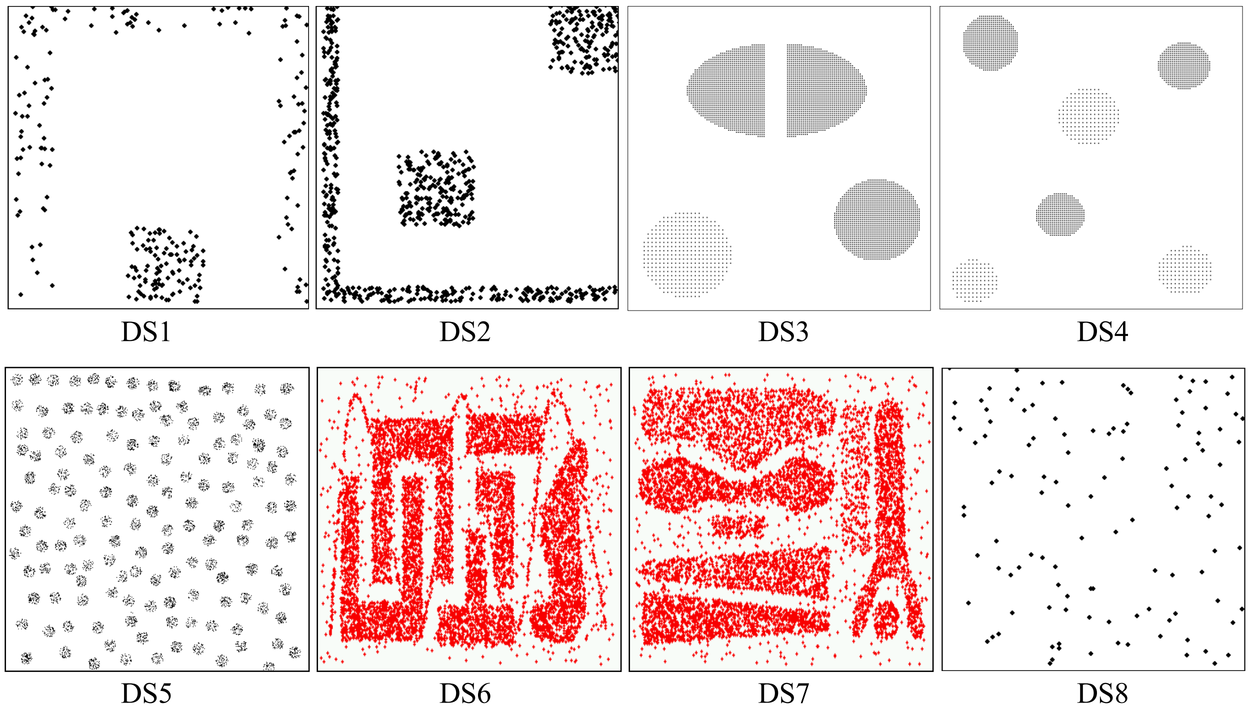


Figure 3.6 Illustrations of datasets from DS1 to DS8

The results of IMPACT are presented and analysed in below. More detailed results and comparisons with other algorithms can be found on the literature of IMPACT algorithm.

Cluster identifying ability To demonstrate the ability of the IMPACT algorithm to identify clusters, we performed clustering on DS1, DS2, DS3, DS4, DS5, DS6, and DS7, which are datasets with different cluster types. The results are shown in **Figure 3.7**. Clustering results obtained from DS1 and DS2 datasets with IMPACT demonstrate that IMPACT can cluster hierarchical datasets effectively. Figure 17 shows results obtained from DS3 and DS4 using IMPACT with default parameters. The clustering results show that IMPACT is not affected by the size and density of clusters. In case of DS 5, IMPACT identified correctly all 142 clusters of DS5. The clustering results of the last two datasets DS6 and DS7, shown in Figure 19, are similar to the results reported in the literature [21]: most clusters are the same but in the case of DS7, IMPACT breaks the marked cluster into two smaller clusters owing to the presence of some low-density regions within. The results demonstrate that IMPACT is quite effective in finding clusters of arbitrary shape, density, and orientation.

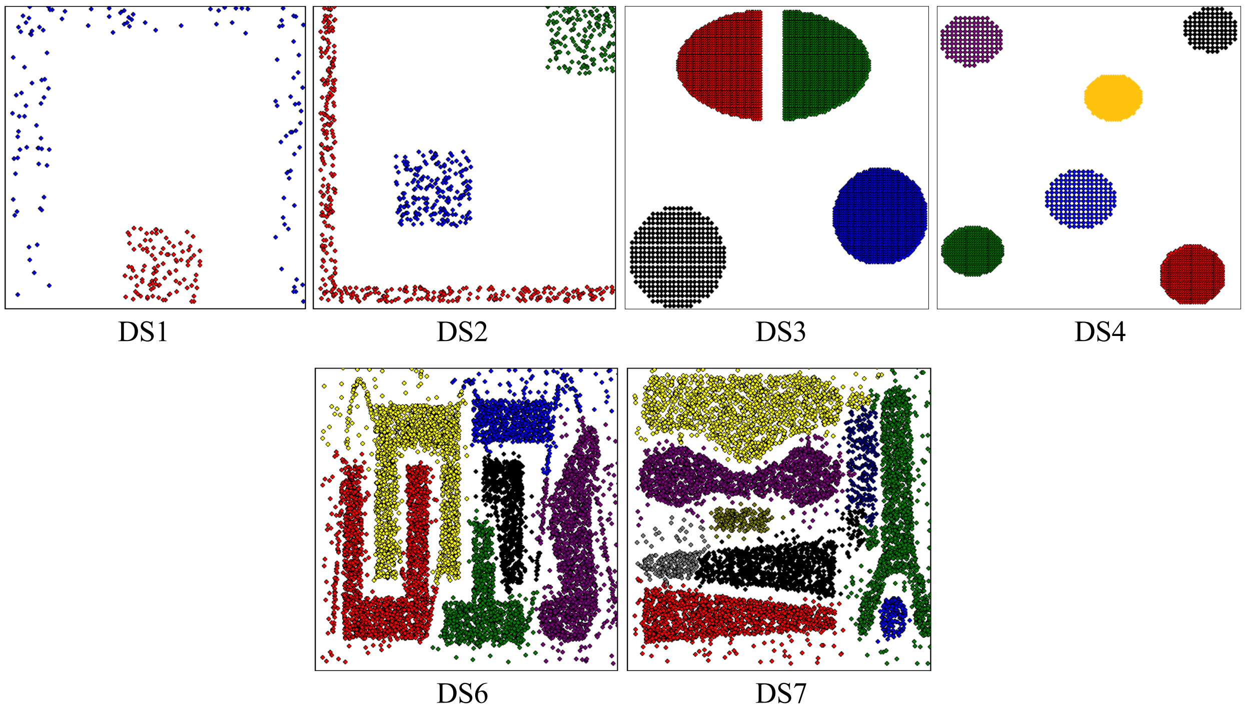


Figure 3.7 Clustering results for DS1, DS2, DS3, DS4, DS6, and DS7 using IMPACT algorithm

Parameter sensitivity of the IMPACT algorithm One of the most critical clustering problems is sensitivity to input parameters. To obtain accurate clustering results, we usually need to estimate the best value of the parameters for the given dataset. The IMPACT algorithm is designed to overcome this problem. We ran IMPACT with default parameters ($p = 2$, $q = 0.05$, *min_ClusterSize* = 20%) on DS8. Clustering results obtained by running IMPACT with different parameter values of p and q are shown in **Figure 3.8**. It is seen how with different values of p and q, IMPACT produces the same results with a dataset with not well-separated clusters. This result suggests that the IMPACT algorithm is not parameter sensitive.

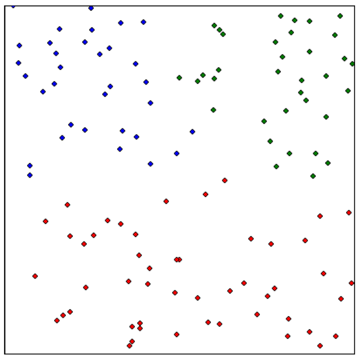


Figure 3.8 Clustering results for DS8 using IMPACT

Practical datasets The IMPACT algorithm not only works effectively with two-dimensional datasets but also produces accurate results when dealing with practical datasets. We used two common datasets from the UCI Machine Learning Repository (Wine and Iris) and one text dataset (R8-) for validation. In the case of the Wine and Iris datasets, the IMPACT algorithm found the correct number of clusters in most tests and archive highest Rand index scores [22] in most of cases. The text dataset R8- needs to be preprocessed before clustering. We used a Perl program to stem nouns and verbs to generate the dictionary and feature vector from R8-. However, because of the high dimension of feature vectors, IMPACT failed to identify the clusters in the dataset. To avoid this problem, we applied PCA to reduce the number of features, and then employed IMPACT for clustering. Even IMPACT could detect seven clusters only, but its result is remarkable because (1) IMPACT did not require the correct number of clusters and (2) the $F_{measure}$ score [22] for IMPACT (0.87) is higher than other results in the literature [23] and [24].

In all the experiments described above, our algorithm was able to identify clusters accurately for most of the datasets and was insensitive to the choice of parameters. However, there are several limitations existed for IMPACT algorithm:

- The datasets are not completely denoised.
- In several cases, small parts of clusters are merged.
- IMPACT takes long processing time to cluster the data.

In this study, we propose a data preprocessing algorithm named D-IMPACT (Density-IMPACT) [25] to overcome the limitation of gravity-based preprocessing algorithms by utilizing the idea of IMPACT algorithm and the concept of density [8]. An advantage of our algorithm is its flexibility in relation to various types of data; it is possible to select an affinity function suitable for the characteristic of the dataset. This flexibility improves the quality of cluster analysis even if the dataset is high-dimensional and non-linearly distributed, or includes noisy samples.

3.3 Data preprocessing algorithm D-IMPACT

In this section, we describe the data preprocessing algorithm D-IMPACT based on concepts underlying the IMPACT algorithm. We aim to improve the accuracy and flexibility of the movement of data points in the IMPACT algorithm by applying the concept of density to various affinity functions. These improvements will be described in the subsequent subsections.

3.3.1 Movement of data points

The main difference between the data movement in D-IMPACT and IMPACT algorithms is that the movement of data points can be varied by the density functions, the attraction functions, and an *inertia* value. This helps D-IMPACT detect different types of clusters and avoid many common clustering problems. In this subsection, we describe the scheme to move data points in D-IMPACT. We assume that the dataset has *m* samples and each sample is characterized by *n* features. We also denote the feature vector of the i^{th} sample by x_i.

3.3.1.1 Density

We use two formulae to compute the density of a data point based on its neighbors, which are defined as data points located within a radius Φ. This density is calculated with and without considering the distance from the data point to its neighbors. We define the density δ_i for the data point x_i as

$$\delta_i = \mathrm{den}(x_i),$$

where den(xi) is one of following density functions:

$$\mathrm{den}_1(x_i) = |NN(x_i)|,$$

$$\mathrm{den}_2(x_i) = \left(\frac{|NN(x_i)|}{\sum_{x_j \in NN(x_i)} \mathrm{distance}(x_i, x_j)} \right),$$

where $NN(x_i)$ is the set of neighbors of x_i and $|NN(x_i)|$ is the number of neighbors. Unlike the density function den_1, the density function den_2 considers not only the number of neighbors, but also the distance between them to avoid issues relating to the choice of threshold value, Φ. In a practical application, we scale the density to avoid scale differences arising from the use of specific datasets as follows:

$$\delta_i = \frac{\delta_i}{\max_{j=1..m}(\delta_j)}.$$

3.3.1.2 Attraction

In our D-IMPACT algorithm, the data points attract each other and one other closer. We define the attraction of data point x_i caused by x_j as

$$A_{ij} = \text{attraction}(x_i,x_j) = \begin{cases} 0 & if \quad \text{distance}(x_i,x_j) < \Phi \\ \text{aff}(x_i,x_j) & if \quad \text{distance}(x_i,x_j) \geq \Phi \end{cases},$$

where aff(x_i,x_j) is a function used to compute the affinity between two data points x_i and x_j. This quantity ignores the affinity between neighbors. The affinity can be computed using the following formulae:

$$\text{aff}_1(x_i,x_j) = \frac{1}{\text{distance}(x_i,x_j)^p},$$

$$\text{aff}_2(x_i,x_j) = \delta_j \times \frac{1}{\text{distance}(x_i,x_j)^p},$$

$$\text{aff}_3(x_i,x_j) = \frac{\min(\delta_i,\delta_j)}{\max(\delta_i,\delta_j)} \times \frac{1}{\text{distance}(x_i,x_j)^p},$$

$$\text{aff}_4(x_i,x_j) = \delta_j \times \frac{\min(\delta_i,\delta_j)}{\max(\delta_i,\delta_j)} \times \frac{1}{\text{distance}(x_i,x_j)^p}.$$

These four formulae have been adopted to improve the quality of the movement process in specific cases. The function aff$_1$, used in IMPACT, considers the distance between two data points only. The function aff$_2$ considers the effect of density on the attraction; highly-aggregated data points cause stronger attraction between one another than sparsely-scattered data points. This technique can improve the accuracy of the movement process. The function aff$_3$ considers the difference between the densities of two data points; two data points attract each other more strongly if their densities are similar. This can be used in the case where clusters are adjacent but have differing densities. The function aff$_4$ is a combination of aff$_2$ and aff$_3$. The parameter p is used to adjust the effect of the distance to the affinity. Attraction is the key value affecting the computation of the movement vectors. For each specific problem in clustering, an appropriate attraction computation can help D-IMPACT to correctly separate clusters.

Under the effect of attraction, two data points will move toward each other. This movement is represented by an *n*-dimensional vector called the affinity vector. We denote a_{ij} as the affinity vector of data point x_i caused by data point x_j. The k^{th} element of a_{ij} is defined as

$$a_{ijk} = \frac{x_{jk} - x_{ik}}{\sum_{r=1}^{n} | x_{jr} - x_{ir} |} \times A_{ij} \quad (k = 1..n).$$

The affinity vector is a component used to calculate the movement vector.

3.3.1.3 Inertia value

To shrink clusters, D-IMPACT moves the data points at the border region of original clusters to the centroid of the cluster. Highly aggregated data points, usually located around the centroid of the cluster, should not move too far. In contrast, sparsely-scattered data points at the border region should move to the centroid quickly. Hence, we introduce an inertia value to adjust the magnitude of each movement vector. We define the inertia value I_i of data point x_i based on its density[1] by

$$I_i = 1 - \delta_i .$$

3.3.1.4 Data point movement

D-IMPACT moves a data point based on its corresponding movement vector. The movement vector v_i of data point x_i is the summation of all affinity vectors that affect the data point x_i

$$v_i = \sum_{j=1}^{m} a_{ij},$$

where a_{ij} is the affinity vector. The movement vectors are then adjusted by the inertia value and scaled by *s,* which is a scaling value used to ensure the magnitude does not exceed a value *Φ*, as in the IMPACT algorithm. This scaling value is given by

$$s = \frac{\Phi}{\max_{i=1..m}(||v_i||)}.$$

[1] In the case of sparse datasets, neighbor detection based on a scanning radius usually fails. Therefore, most of data points will have a density equal to 1. Hence, we replace the formula used to compute the inertia value with $I_i = 1 - \delta_i/2$.

Finally, each data point is moved using

$$x_{i(k)} = x_{i(k-1)} + s \times I_i \times v_i,$$

where $x_{i(k\text{-}1)}$ is the coordinate of data point x_i in the previous iteration, and $x_{i(k)}$ is the coordinate of data point x_i in this iteration. We propose the algorithm D-IMPACT based on this scheme of moving data points.

3.3.2 D-IMPACT algorithm

D-IMPACT has two phases. The first phase detects noisy and outlier data points, and then removes them. The second separates clusters by iteratively moving data points based on attraction and density functions. **Figure 3.9** shows the flow chart of the D-IMPACT algorithm. The algorithm will be explained in detailed in next sections.

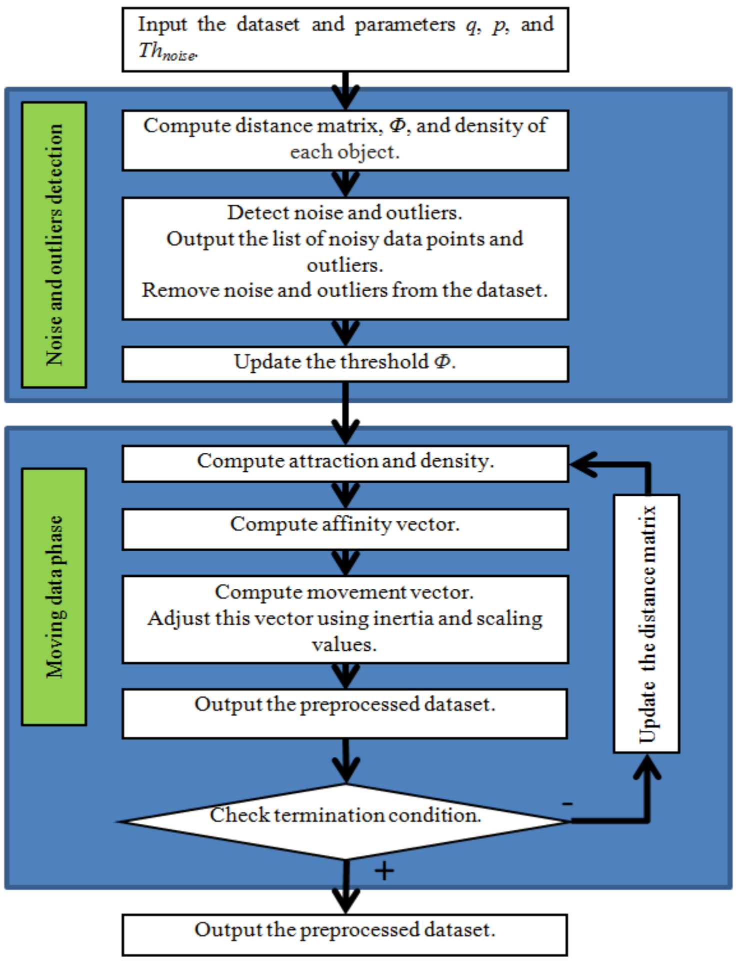

Input the dataset and parameters q, p, and Th_{noise}.
Noise and outliers detection
Compute distance matrix, Φ, and density of each object.
Detect noise and outliers.
Output the list of noisy data points and outliers.
Remove noise and outliers from the dataset.
Update the threshold Φ.
Moving data phase
Compute attraction and density.
Compute affinity vector.
Compute movement vector.
Adjust this vector using inertia and scaling values.
Output the preprocessed dataset.
Update the distance matrix
Check termination condition.
-
+
Output the preprocessed dataset.

Figure 3.9 Outline of the D-IMPACT algorithm

3.3.2.1 Noisy points and outlier detection

First, the distance matrix is calculated. The density of each data point is then calculated by one of the formulae defined in the previous subsection. The threshold used to identify neighbors is computed based on the maximum distance and the input parameter q, and is given by

$$\Phi = q \times maxDistance,$$

where *maxDistance* is the largest distance between two data points in the dataset.

The next step is noise and outlier detection. An outlier is a data point significantly distant from the clusters. We refer to data points which are close to clusters but do not belong to them to as noisy points, or noise, in this manuscript. Both of these data point types are usually located in sparsely-scattered areas, that is, low-density regions. Hence, we can detect them based on density and the distance to clusters. We consider a data point as noisy if its density is less than a threshold Th_{noise}, and it has at least one neighbor which is noisy or a cluster-point (with the latter defined as a data point whose density is larger than Th_{noise}). An outlier is a point with a density less than Th_{noise} that has no neighbor which is noisy or a cluster-point. **Figure 3.10** gives an example of noise and outlier detection.

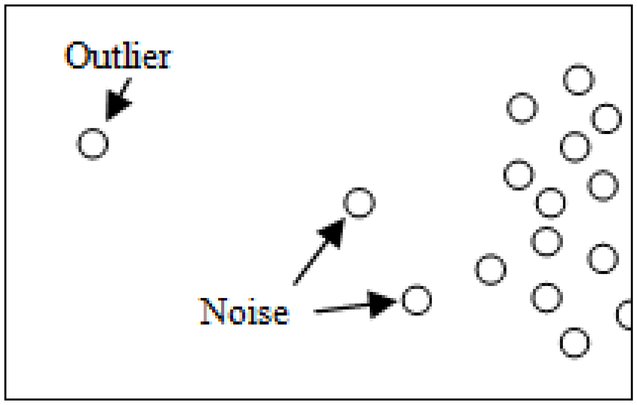


Figure 3.10 An illustration of noisy points and outliers.

Both outliers and noisy points are output and then removed from the dataset. The effectiveness of this removal is shown in **Figure 3.11**. The value Φ is then recalculated as the dataset has been changed by the removal of noise and outliers. When this phase is complete, the movement phase commences.

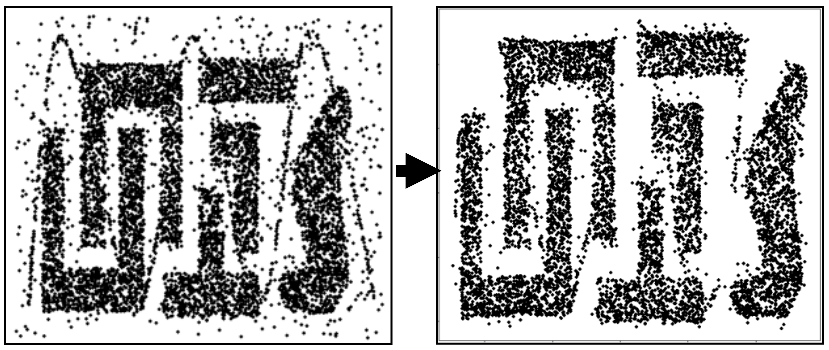


Figure 3.11 Illustration of the effect of noise removal in D-IMPACT.

3.3.2.2 Moving data points

In this phase, the data points are iteratively moved until the stop criterion is met. The distances and the densities are calculated first, after which, we compute the components used to determine the movement vectors: attraction, affinity vector, and the inertia value. We then employ the movement method described in the previous section to move the data points. The movement shrinks the clusters to increase their separation from one another. This process is repeated until the stop condition is satisfied. In D-IMPACT, we adopt various stop criteria as follows:

- Stop after a fixed number of iterations controlled by the parameter n_{iter}.
- Stop based on the average of the densities of all data points.
- Stop when the magnitudes of movement vectors have decreased significantly compared to the previous iteration.

When this phase is complete, the preprocessed dataset is output. The new dataset contains separated and shrunk clusters, with noise and outliers removed.

3.3.2.3 Complexity

D-IMPACT is a computationally-efficient algorithm. The cost of computing m^2 affinity vectors is O(m^2n). The complexity of the computation of movement vectors is O(mn). Therefore, the overall cost of an iteration is O(m^2n). We see, based on our experiments, that the number of iterations is usually small and does not have significant impact on the overall complexity. Therefore, the overall complexity of D-IMPACT is O(m^2n).

We measured the real processing time of D-IMPACT on 10 synthetic datasets. For each dataset, the data points were randomly located (uniformly distributed). The sizes of the datasets varied from 1000 to 5000 samples. These datasets are included in the supplement. We compared D-IMPACT with CLUES using these datasets. D-IMPACT was employed with the parameter n_{iter} set to 5. For CLUES, the number of neighbors was set to 5% of the number of samples and the parameter *itmax* was set to 5. The experiments were executed using a workstation with a T6400 Core 2 Duo central processing unit running at 2.00 GHz with 4 GB of random access memory. **Figure 3.12** shows the advantage in speed of D-IMPACT in relation to CLUES.

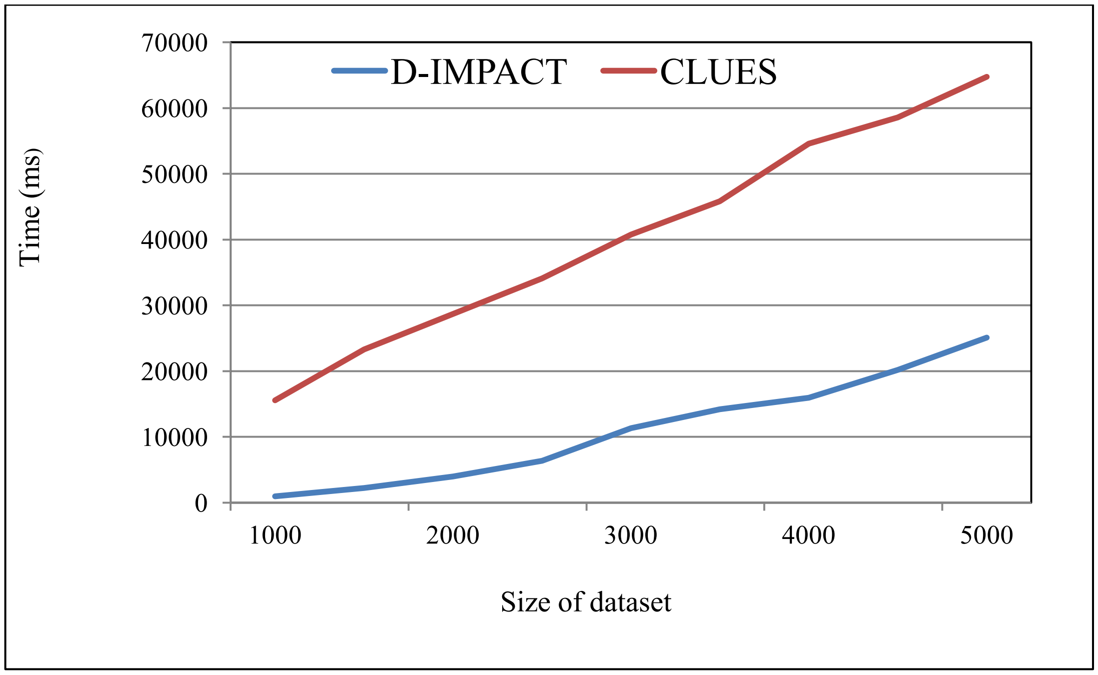


Figure 3.12 Processing times of D-IMPACT and CLUES on test datasets.

3.4 Experiment result

In this section, we compare the effectiveness of D-IMPACT and the shrinking function of CLUES (in short, CLUES) on different types of datasets.

3.4.1 Datasets and method

3.4.1.1 Two-dimensional datasets

To validate the effectiveness of D-IMPACT, we used different types of datasets: two dimensional (2D) datasets taken from the Machine Learning Repository (UCI), and a microarray dataset. **Figure 3.13** shows the 2D datasets used.

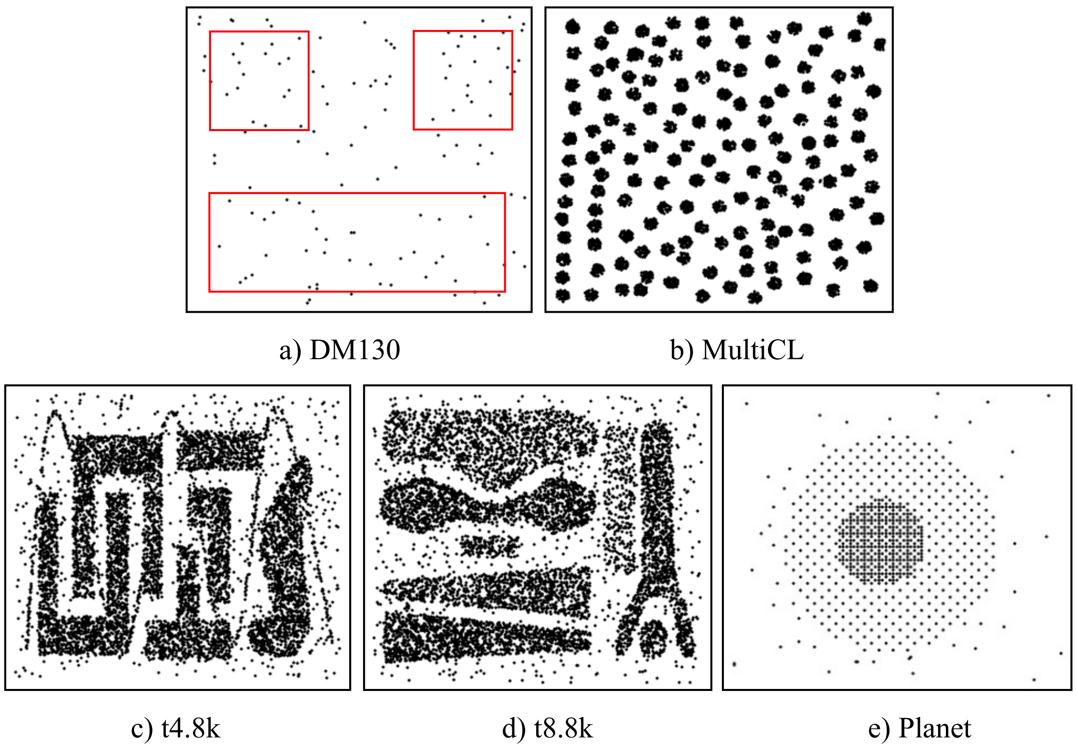


Figure 3.13 Visualizations of 2D datasets.

The 2D datasets are DM130, t4.4k, t8.8k, multiCL, and Planet. They contain clusters with different shapes, densities and distributions, as well as noisy samples. The DM130 dataset has 130 data points: 100 points are generated randomly (uniformly distributed), and then three clusters, where each cluster comprises ten data points, are added to the top-left, top-right and bottom-middle area of the dataset (marked by red rectangles in **Figure 3.13a**). The MultiCL dataset has a large number of clusters (143 clusters) scattered equally. Two datasets, t4.8k and t8.8k [18], used in the analysis of the clustering algorithm Chameleon [21], are well-known datasets for clustering. Both contain clusters of various shapes and are covered by noisy samples. Clusters are chained by the single-link effect in the t4.8k dataset. The clusters of the Planet dataset are adjacent, but differ in density. These datasets encompass common problems in clustering.

3.4.1.2 Practical datasets

The practical datasets are more complex than the 2D datasets, i.e., the high dimensionality can greatly impact the usefulness of the distance function. We used the Wine, Iris, Water-treatment plant (WTP), and Lung-cancer (LC) datasets from UCI, as well as the dataset GSE9712 from the Gene Expression Omnibus [26] to test D-IMPACT and CLUES on high-dimensional datasets. The datasets are summarized in **Table 3.2**. The Iris dataset contains three classes (Iris Setosa, Iris Versicolour, Iris Virginica), each with 50 samples. One class is linearly separable from the other two; the latter are not linearly separable from each other. The Wine dataset (178 samples, 13 attributes), which are the results of chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars, include three overlapping clusters. The WTP dataset (527 samples, 38 attributes) includes the record of the daily measures from sensors in an urban waste water treatment plant. It is an imbalanced dataset - several clusters have only 1-4 members, corresponding to the days that have abnormal situations. The lung-cancer (LC) dataset (32 samples, 56 attributes) describes 3 types of pathological lung cancers. Since the Wine, WTP, and LC datasets have attributes within different ranges, we perform scaling to avoid the domination of wide-range attributes. The last dataset we use is a gene expression dataset, GSE9712, which contains expression values of 22283 genes from 12 radio-resistant and radio-sensitive tumors.

Table 3.2 Experiment datasets for D-IMPACT algorithm.

Dataset	Size of datasets	Number of features	Number of clusters
DM130	130	2	3
MultiCL	8026	2	143
t4.8k	8000	2	8
t8.8k	8000	2	8
Planet	719	2	2
Iris	150	4	3
Wine	178	13	3
WTP	527	38	13
LC	32	56	3
GSE9712	12	22283	4

3.4.1.3 Validating methods

For a fair comparison, we employed CLUES implemented in R [27] and varied the number of neighbors k (from 5% to 20% of the number of samples) for different datasets. For D-IMPACT, we used the default parameter set (q = 0.01, p = 2, aff$_1$, den$_1$, Th_{noise} = 0, n_{iter} = 2) with some modifications. The complete parameter set is described in **Table 3.3**. We compared the differences between the preprocessed datasets and the original datasets using 2D plots. However, it is difficult to visualize the high-dimensional datasets using only 2D plots. For this reason, we compared the two algorithms by using a plot showing several combinations of features. Further, to evaluate the quality of the preprocessing, we compared the clustering results for the datasets preprocessed by D-IMPACT and CLUES. We used two evaluation measures, the Rand Index and adjusted Rand Index (aRI) [22]. Hierarchical agglomerative clustering (HAC) was used as the clustering method [1]. We used the Wine, Iris, and GSE9712 datasets to validate the clustering results, and the WTP and LC datasets to validate the ability of D-IMPACT to separate outliers from clusters.

Table 3.3 Parameter sets of D-IMPACT for experiments.

Dataset	Parameter set
DM130	$p = 4$, $n_{iter} = 3$
MultiCL	den_2, aff_2
t4.8k	$q = 0.03$, $Th_{noise} = 0.1$, $n_{iter} = 1$
t8.8k	$q = 0.03$, $Th_{noise} = 0.1$, $n_{iter} = 1$
Planet	$q = 0.05$, p=4, den_2, aff_3, $n_{iter} = 4$
Iris	n_{iter} =5
Wine	$p = 4$, $Scale$ = true, $I_i = 1 - De_i/2$
WTP	$Scale$ = true, aff_2
LC	$Scale$ = true
GSE9712	$I_i = 1 - De_i/2$

3.4.2 Experiment results of 2D datasets

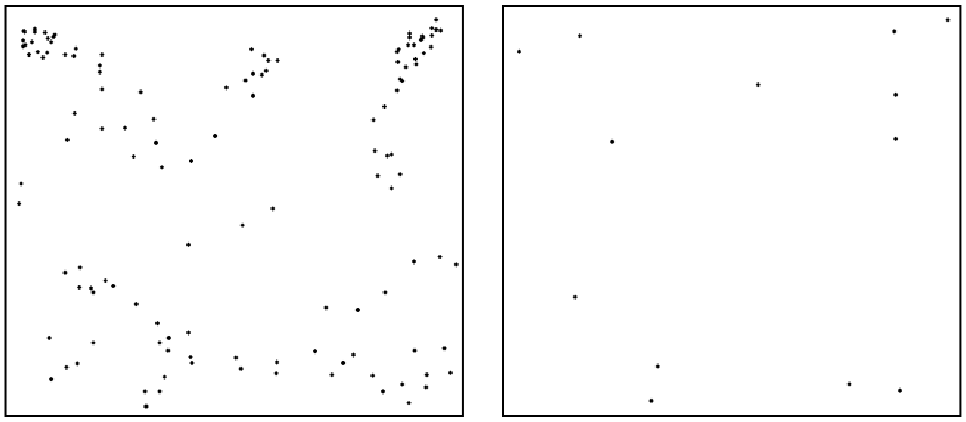
The results of D-IMPACT and CLUES on 2D datasets DM130, MultiCL, t4.8k, t8.8k, and Planet are displayed and analyzed in this section.

Clusters in the dataset DM130 are difficult to recognize since they are not dense or well separated. Therefore, we set the p to 4 and run D-IMPACT for longer (n_{iter} = 3). The D-IMPACT algorithm shrinks the clusters correctly and retains structures of the original dataset (**Figure 3.13a** and **Figure 3.14a**). CLUES, with the number of neighbors k varied from 10 to 30, degenerated the clusters into a number of overlapped points and caused a loss of the global structure (**Figure 3.14b**).

The shrinking process may merge clusters incorrectly since clusters in the dataset MultiCL are dense and closely located. Hence, we used the density function den_2 and the affinity function aff_2, which emphasizes the density, to preserve the clusters. The result is shown in **Figure 3.15**. D-IMPACT correctly shrunk the clusters (**Figure 3.15a**), yet CLUES merged some clusters incorrectly due to issues relating to the choice of k (**Figure 3.15b**).

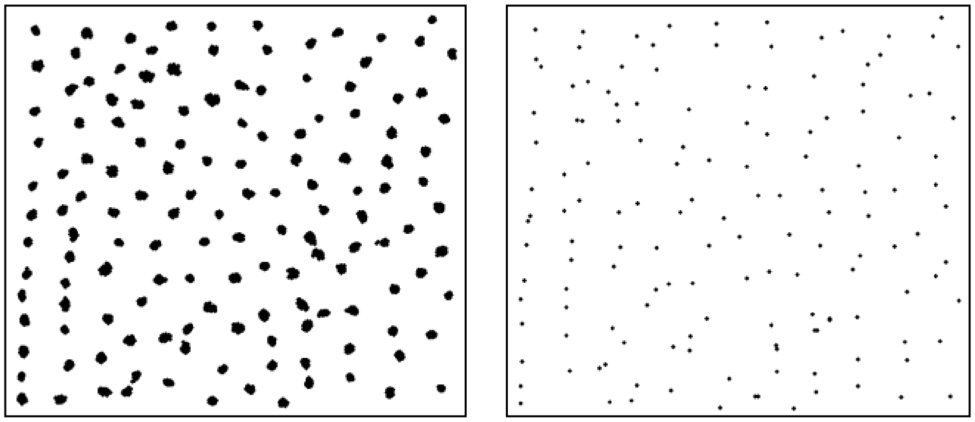
In relation to the two datasets t4.8k and t8.8k, D-IMPACT and CLUES are expected to remove noise and shrink clusters. We set $q = 0.03$ and $Th_{noise} = 0.1$ to detect carefully noise and outliers. The results of D-IMPACT are shown in **Figure 3.16**; the majority of noise was removed, and clusters were shrunk and separated. We then tested CLUES on the t4.8k dataset. Since the clusters in t4.8k are heavily covered by noise, we tested CLUES on the dataset whose noise was removed by D-IMPACT for a fair comparison. The value k is varied to test the parameter sensitivity of CLUES. **Figure 3.17** shows different results due to this parameter sensitivity.

To separate adjacent clusters in the dataset Planet, we used the function aff$_3$, which considers the density difference. The parameter q is set to 0.05, since the data points are located near each other. We used den$_2$ and $p = 4$ to emphasize the distance and density. The results are shown in **Figure 3.18**. As shown, D-IMPACT clearly outperformed CLUES.



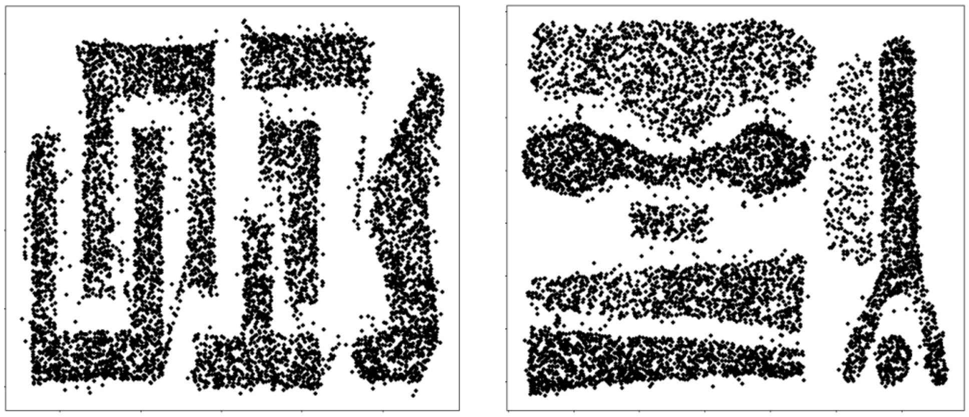
a) D-IMPACT b) CLUES

Figure 3.14 Visualization of the dataset DM130 preprocessed by D-IMPACT and CLUES.



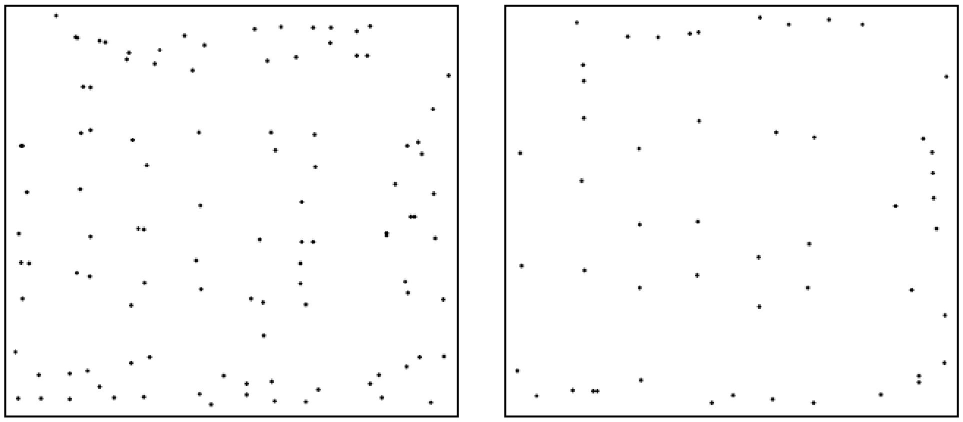
a) D-IMPACT b) CLUES

Figure 3.15 Visualization of the dataset MultiCL preprocessed by D-IMPACT and CLUES.



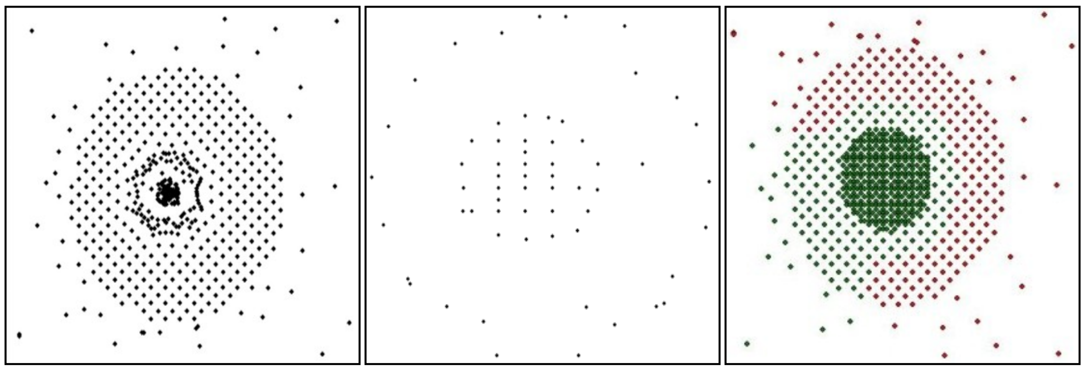
a) t4.8k b) t8.8k

Figure 3.16 Visualization of two datasets t4.8k and t8.8k preprocessed by D-IMPACT.



a) $k = 80$ (1%) b) $k = 160$ (2%)

Figure 3.17 Visualizations of the dataset t4.8k preprocessed by CLUES using different values of *k* based on the size of the dataset.



a) Preprocessed by D-IMPACT. Two clusters are separated.

b) Preprocessed by CLUES

c) Clustering result using HAC on the dataset in b), indicating that CLUES shrinks clusters incorrectly

Figure 3.18 Visualization of the dataset Planet preprocessed by D-IMPACT and CLUES.

3.4.3 Experiment results of practical datasets

3.4.3.1 Iris, Wine, and GSE9712 datasets

To avoid the domination of wide-range features, we scaled several datasets (*Scale* = true). In the case of Wine, we had to modify the inertia value and use $p = 4$ to emphasize the importance of nearest neighbors. We used HAC to cluster the original and preprocessed Iris and Wine datasets, and then validated the clustering results with aRI. A higher Rand Index score indicates a better clustering result. The Iris dataset was also preprocessed using a PCA-based de-noising technique. However, the distance matrices before and after applying PCA are nearly the same (using 2, 3, or 4 principal components (PCs)). Therefore, the clustering results of HAC for the dataset preprocessed by PCA are at most the same result as that of the original dataset, which depends on the number of PCs used (aRI score ranged from 0.566 to 0.759). **Table 3.4** shows the aRI scores of clustering results of HAC on original datasets and datasets preprocessed by D-IMPACT and CLUES. The effectiveness was dependent on the datasets. In the case of Iris, D-IMPACT greatly improved the dataset, particularly as compared with CLUES. However, for the Wine dataset, CLUES achieved the better result. This is due to the overlapped clusters in the Wine dataset are undistinguishable using affinity function. The GSE9712 dataset is high-dimensional and has a small number of samples. Due to the curse of dimensionality and the noise included in microarray data, it is very difficult to distinguish clusters based on the distance matrix. We performed D-IMPACT and CLUES on this dataset to improve the distance matrix, and then applied the clustering algorithm HAC. D-IMPACT clearly outperformed CLUES since CLUES greatly decreased the quality of the cluster analysis.

We also performed *k*-means clustering [1] on these datasets. We performed 100 different initializations for each dataset. The clustering results also favored D-IMPACT. **Table 3.5** shows the best and average scores (in brackets) of the experiments. In addition, using Welch's two sample *t*-test, the stability of the clustering result on D-IMPACT increased; the *p*-values between two experiments (100 runs of *k*-means for each experiment) of the original dataset, CLUES, and D-IMPACT were 0.490, 0.365 and 0.746, respectively. Since the *p*-value of the *t*-test is the confidence of the alternative "the two vectors have different means", a higher *p*-value indicates more stable clustering results.

To clearly show the effectiveness of the two algorithms, we visualized the Iris and Wine datasets preprocessed by D-IMPACT and CLUES as shown in **Figures 3.19** and **3.20**. Since Wine has 13 features (i.e. 78 subplots are required to visualize all the combinations of the 13 features), we only visualize the combinations for the first four features, using 2D plots (**Figure 3.20**). D-IMPACT successfully separated two adjacent clusters (blue and red) in the Iris dataset. D-IMPACT also distinguished overlapping clusters in the Wine dataset. We marked the separation created by D-IMPACT with red-dashed ovals in **Figure 3.20**. This shows that D-IMPACT worked well with overlapped clusters. CLUES degenerated the dataset into a number of overlapped points. This caused the loss of cluster structures and reduced the stability of clusters in the dataset (**Figure 3.21**). Therefore, the use of *k*-means created different clustering results during the experiment.

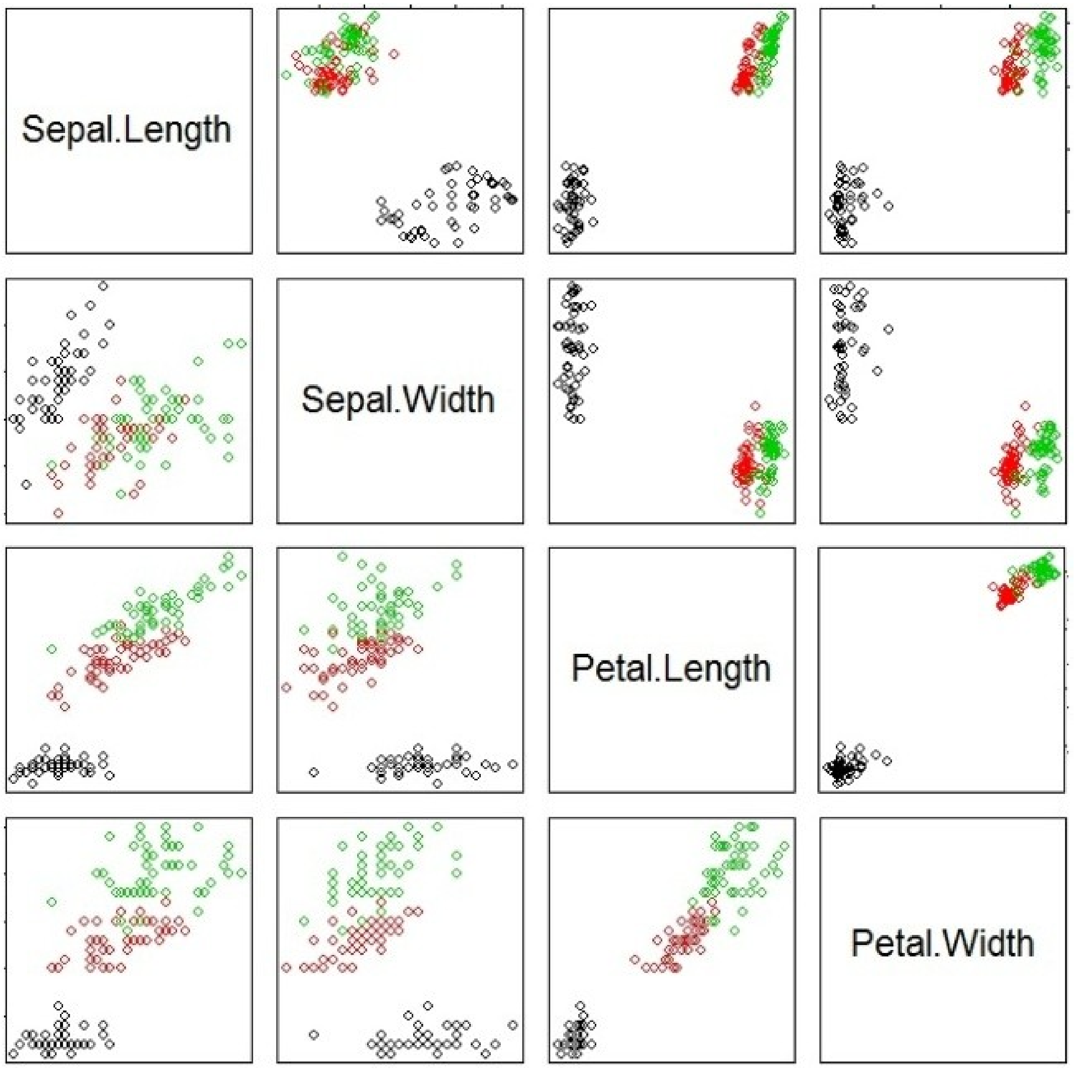


Figure 3.19 Visualization of the Iris dataset before and after preprocessing by D-IMPACT. Visualization of the original dataset is shown in the bottom-left triangle. Visualization of the dataset optimized by D-IMPACT is shown in the top-right triangle.

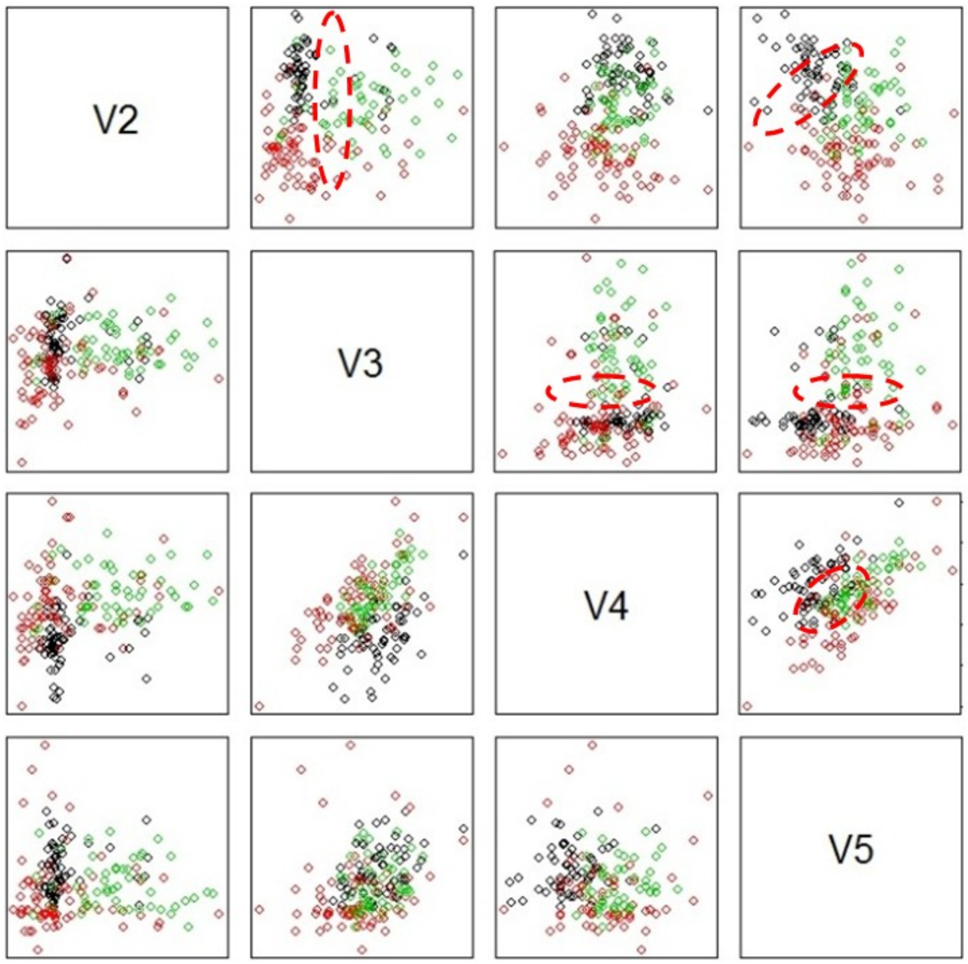
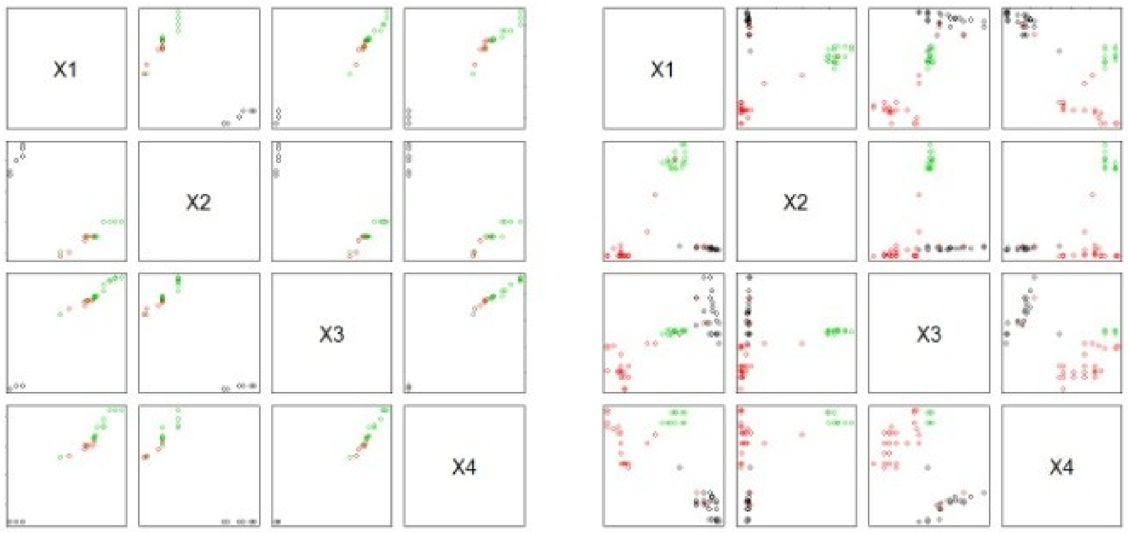


Figure 3.20 Visualization of the first four features of the Wine dataset before and after preprocessing by D-IMPACT. Visualization of the original dataset is shown in the bottom-left triangle. Visualization of the dataset preprocessed by D-IMPACT is shown in the top-right triangle.



a) Iris b) Wine

Figure 3.21 Visualization of the Iris and Wine datasets preprocessed by CLUES.

Table 3.4 Index scores of clustering results using HAC[1] on the original and preprocessed datasets of IRIS and Wine. The best scores are in bold.

Dataset	Preprocessing algorithm		
	None	CLUES	D-IMPACT
Iris	0.759	0.732	**0.835**
Wine	0.810	**0.899**	0.884
GSE9712	0.330	0.139	**0.330**

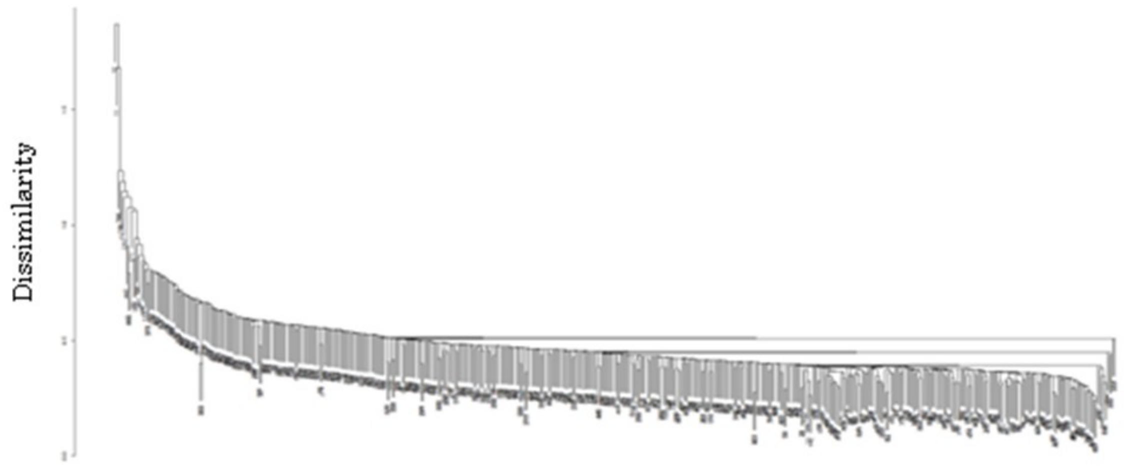
Table 3.5 Index scores of clustering results using *k*-means on original and preprocessed datasets of iris and wine. The best scores are in bold.

Dataset	Preprocessing algorithm		
	None	CLUES	D-IMPACT
Iris	0.730 (0.682)	0.757 (0.677)	**0.757** (**0.686**)
Wine	0.899 (**0.859**)	**0.915** (0.814)	0.899 (0.852)
GSE9712	0.403 (0.212)	0.139 (0.224)	**0.403** (**0.329**)

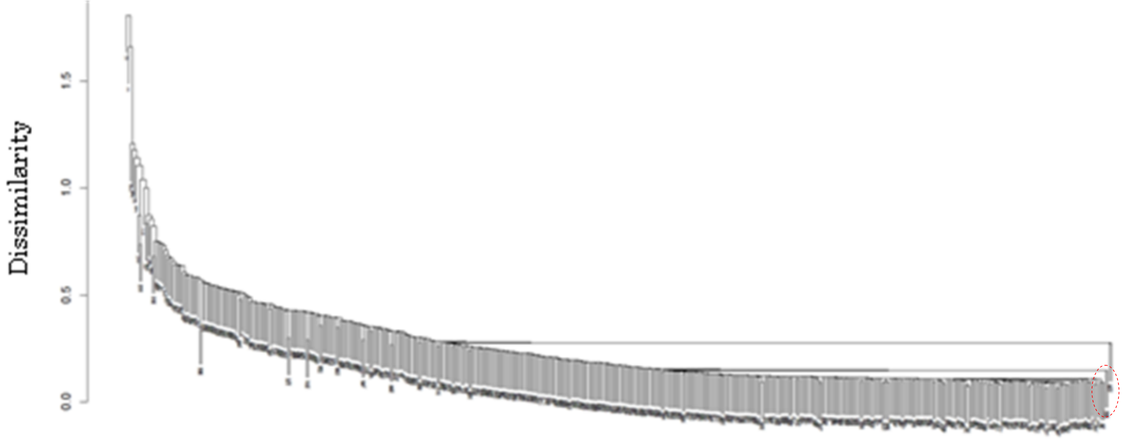
[1] We used the linkage that achieved the best result on the original dataset to perform clustering on the preprocessed dataset. These were average linkage for Iris, complete linkage for Wine dataset, and single linkage for GSE9712

3.4.3.2 Water treatment plant and Lung cancer datasets

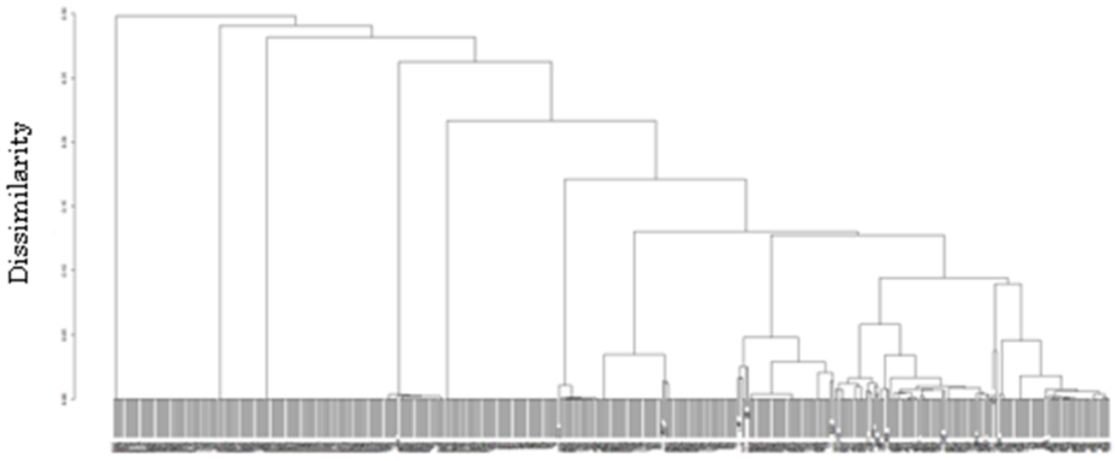
To validate the outlier separability, we tested CLUES and D-IMPACT on the WTP and LC datasets. The WTP dataset has small clusters (1-4 samples for each cluster). Using aff_2, we can reduce the effect of the affinity to these minor clusters. We show the dendrogram of HAC clustering results (using single-linkage) on the original and preprocessed dataset of WTP in **Figure 3.22**. In the dataset preprocessed by D-IMPACT, several minor clusters are more distinct than the major clusters (**Figure 3.22b**). In addition, the quality of the dataset was improved after preprocessing by D-IMPACT; the clustering result using *k*-means (100 runs) on the dataset preprocessed by D-IMPACT achieved average aRI = 0.217, while the clustering result on the original dataset had average aRI = 0.120. CLUES merged minor clusters during shrinking and, therefore, the clustering result was bad (average aRI = 0.114). To compare the outlier detection capability of D-IMPACT and CLUES, we calculated the Rand Index scores for only minor clusters. The resulting dataset preprocessed by D-IMPACT achieved Rand Index = 0.912, while CLUES had Rand Index = 0.824. In addition, in the clustering result on the dataset preprocessed by D-IMPACT, 8 out of 9 minor clusters were correctly detected. In contrast, no minor cluster was correctly detected when using CLUES.



a) Dendrogram of the original Water-treatment dataset.



b) Dendrogram of the Water-treatment dataset after being preprocessed by D-IMPACT.



c) Dendrogram of the Water-treatment dataset after being preprocessed by CLUES.

Figure 3.22 Dendrograms of the clustering results on the WTP dataset.

The lung cancer (LC) dataset was used by R. Visakh and B. Lakshmipathi to validate the outlier detection ability of an algorithm focusing on a constraint based cluster ensemble using spectral clustering, called CCE [28]. The dataset has no obvious noise or outliers. We detected some noise and outlier points by considering the distance to the nearest neighbor and the average distance to the *k*-nearest neighbors ($k = 6$) of 32 samples in the LC dataset. We generated a list of candidates for noise and outliers: sample numbers 18, 19, 23, 26, and 29. We then performed HAC with different linkages on the original and preprocessed LC datasets to detect noise and outliers based on the dendrogram. These results were then compared with the reported result of CCE. This was done by calculating the accuracy and precision values. The results in **Table 3.6** clearly show that D-IMPACT outperformed CCE. It also shows the effectiveness of D-IMPACT in relation to outlier detection.

Table 3.6 Accuracy and precision values of noise and outlier detection on the Lung-cancer dataset

Preprocessing algorithm	Linkage	Accuracy	Precision
None	Single	0.718	0.5
None	Average	0.343	0.556
None	Complete	0.125	0.222
D-IMPACT	Single	0.781	0.667
D-IMPACT	Average	**0.812**	**1**
D-IMPACT	Complete	**0.812**	**1**
CCE	N/A	0.75	0.6

3.5 Conclusion

In this study, we proposed a data preprocessing algorithm named D-IMPACT inspired by the IMPACT clustering algorithm. D-IMPACT moves data points based on attraction and density to create a new dataset where noisy points and outliers are removed, and clusters are separated. The experiment results with different types of datasets clearly demonstrated the effectiveness of D-IMPACT. The clustering algorithm employed on the datasets preprocessed by D-IMPACT detected clusters and outliers more accurately.

CHAPTER IV
Data preprocessing algorithm SCF

In this chapter, we describe the data preprocessing algorithm SCF which aims to reduce the number of dimensions without losing the semantic information stored in each feature. The new space produced by SCF will contains the semantic similarity between the keywords of each documents and the concept underlying the corpus.

4.1 Clustering algorithms and data preprocessing methods for text clustering

4.1.1 Text clustering

With the rapid growth of information exchange, a large number of documents are created in everyday, such as emails, news, forum post, social network posts, etc. To help people deal with document overload, many systems apply clustering to help people manage, organize, and organize text data more effective. Here are several examples:

Organizing search result. Searching a word on internet can return thousands or more results. In order to help the user be able to quickly capture the overview of searching results, Carrot [29] applies clustering technique on the searching results to classify them into topics. **Figure 4.1** shows the topics obtained by clustering the researching result for the words "Kanazawa".

Recommender system. To help user can keep tracking of the topic, many recommender system apply the basic of clustering to suggest document related to current topic [30] [31] [32]. For example, at the end the news, there are other news are list under the category "related article" to help the reader can quickly find the related news.

Managing document collection. Clustering is a useful technique to manage a large number of documents during daily works, e.g., emails, messages. Documents will be automatically classified into small meaningful groups, which are convenient for users [33] [34].

To cluster text data, the classic clustering algorithms presented in previous section (e.g., k-means, HC, DB-SCAN) are employed [35]. In [36] [37] [38], researchers extend the classic clustering algorithms in order to improve the effectiveness of clustering on text data. Recently, a number of semantic-based clustering techniques are developed [39] [40] [41]. However, there still exist several challenges for text clustering, which will be presented in the next section.

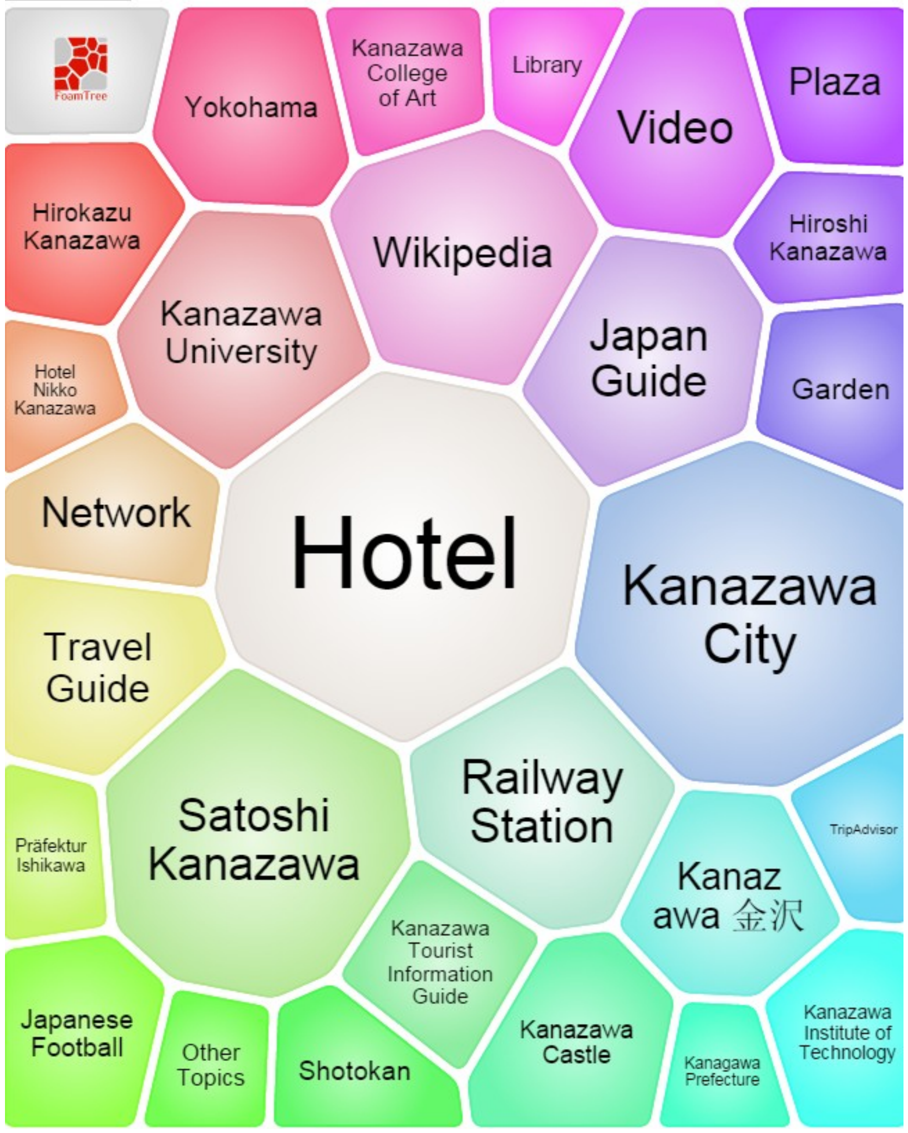


Figure 4.1 Clustering search results for the word "Kanazawa".

4.1.2 Challenges of text clustering in short text data

As we mention in section 3.1.1, there are still several challenges existed for clustering, i.e, the number of clusters, high dimensionality. In case of text data, due to text processing technique, high dimensionality and sparseness becomes a critical problem for text clustering. For the convenience of document similarity computation, text data are usually transformed into a vector space matrix (term frequency matrix) by BOW (bag of words) approach. First, all words (terms) appeared in the corpus (set of documents) are summarized, and then the occurrence of each word are counted for each document. Hence, the term frequency matrix contains a huge number of features due to the diversity of language (high dimensionality). In addition, since a word only appears in several documents, the number of occurrence of it in other document is zero. This make most of the values in the term frequency matrix are zero (sparseness). These problems greatly affect the quality and increase the processing time of clustering. In case of short text, i.e., emails, news, messages, these problems become more critical due to length of the text. **Figure 4.2** shows the increasing of the number of dimensions and sparseness (percentage of zero in the matrix) when the number of short texts increases.

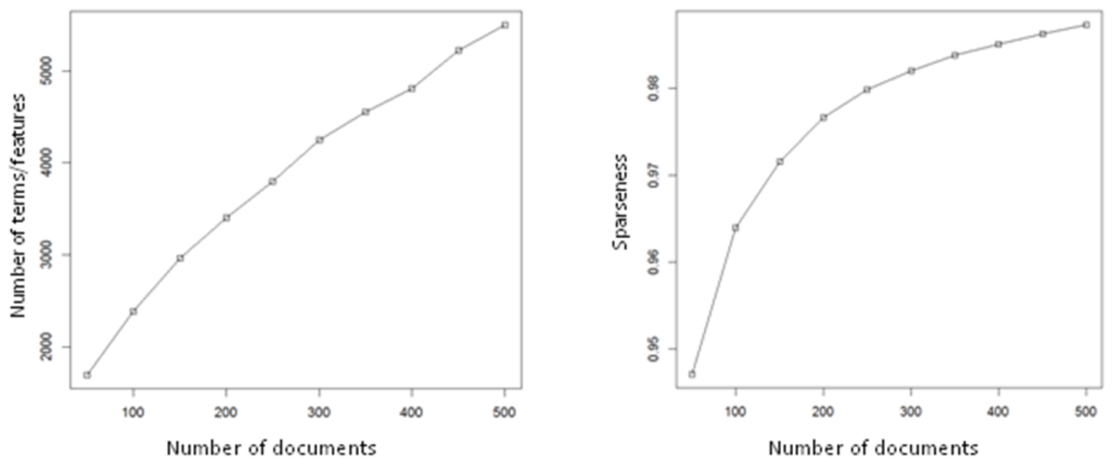


Figure 4.2 The rapid increasing of the number of dimensions and sparseness when the number of texts increases.

In addition to high dimensionality and sparseness, the BOW approach also ignores the semantic relationship between the words. Synonyms words, for example, "car" and "automobile", are considered as two independent features in term frequency matrix and hence, increase the number of dimensions to present the texts in corpus and impact the content presented in the term frequency matrix.

A specific problem for clustering in short text data is the quality of data. Comparing to other kinds of documents, i.e., article, official document, book, the short text is less strict in grammar and may contains a lot of misspellings. In addition, short texts may contain pattern repeated in many texts but not contribute to the content of the text. For examples, quotes from sequence of emails or news, salutation, are repeated in many documents. The poor quality of short text data leads to the poor performance of clustering on them.

One of solutions for problems above is employing data preprocessing before doing clustering in order to improve the quality of clustering results. Next section will present the basic of data processing and introduce briefly several data preprocessing algorithms.

4.1.3 Data preprocessing for text clustering

The clustering algorithms often classify data into groups based on the similarity between them. However, as we presented in previous section, high dimensionality problem makes the similarities between documents become less distinguishable, hence greatly affects the quality of clustering. To handle this problem, data preprocessing methods are often used in order to improve the quality of term frequency matrix, which then will be used to compute the similarities between documents. Besides the preprocessing methods introduced in section 2.2, we briefly introduce several popular data preprocessing techniques for text clustering

Latent Semantic Indexing (LSI) [42]**.** As we introduced in section 2.2, PCA method represents a matrix into a lower dimensional space such that the distance between two matrices is minimized. The concept of PCA is finding new principal components, which linear with variance of data, to reduce the number of dimensions with minimal loss of information. LSI applies PCA technique to project term frequency matrix into "latent" semantic space, which can reveal underlying topics in the documents. However, similar to PCA, the new dimensions produced by LSI are just a linear transformation from original term frequency matrix, so they may not correspond to meaningful topic underlying the documents. In addition, these methods are heavy computation, so they are inefficient to be employed on large datasets.

Semantic-based approaches. The methods introduced in section 2.2 share a same limitation: they ignore the meaning of words in the documents (semantic information). All words are treated independently without considering their semantic relationships to other words. Recently, many researches utilized WordNet [43], a thesaurus for English, to do clustering with considering the semantic relationship between words [39] [40] [41]. However, these researches are quite complex, heavy computation, and cannot completely solve the problem of high dimensionality.

In addition, we also test D-IMPACT algorithm on text data. Since D-IMPACT cannot reduce the number of features, and the topics of the documents are not well presented in the term frequency matrix, D-IMPACT failed to improve the quality of clustering. **Figure 4.3** show the result of clustering on dataset preprocessed by D-IMPACT. The test was done on Enron dataset, one of datasets used for the experiment described in section 4.4.

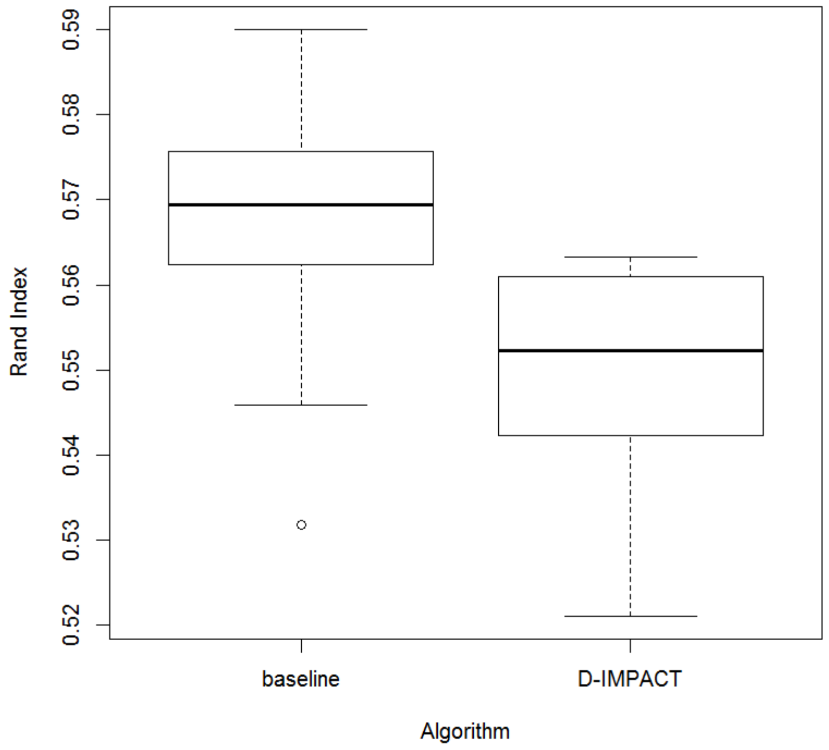


Figure 4.3 D-IMPACT algorithm degenerated the performance of clustering on text data

In the next section, we introduce WordNet, a lexical database in English and the method to measure the semantic similarity based on the structure of WordNet. This function plays an important role in our research.

4.2 WordNet and semantic similarity

WordNet is a large lexical database of English and then extended to other languages. Nouns, verbs, adjectives and adverbs are organized into sets of cognitive synonyms (synsets), which are linked by semantic relationship. This network makes WordNet becomes a useful tool for computational linguistics and natural language processing. For example, researchers in [44] [45] employed WordNet for word sense disambiguation. In [39] [40] [41], WordNet is used as a part of document clustering.

4.2.1 WordNet structure

Information in WordNet is organized around sets of cognitive synonyms called synsets and the relationship between them. WordNet 3.0 contains 155287 unique strings (words) organized into 117659 synsets, which has 206941 relationship links (word-sense pairs) between them. Each synset expresses a distinct concept, and contain the information below:

- List of simple words belongs to the synset.
- Basic definition of the synset.
- One or more examples to illustrate the usage of the synset.
- Semantic relationship to other synsets.

To give and easier understanding of WordNet structure, we illustrate the structure of Wordnet in **Figure 4.4**. The senses of “Province” are presented by two different synsets (polynym). The first synset (entity→physical_entity→object→location→region→district→ administrative_district→state) contains two lemmas: “state” and “province”. The definition of this synset is “the territory occupied by one of the constituent administrative districts of a nation”. In addition, an example of the usage of this synset is also given: "his state is in the deep south". Since the word “province” is a polynym (words has different senses), there is another synset presents the word (entity→abstraction→attribute→state→situation→environment→sphere→province). This synset also has two lemmas “province” and “responsibility”, with the definition is “the proper sphere or extent of your activities”. A simple usage example is also given: "it was his province to take care of himself".

In WordNet, there are several kinds of relationship between synsets. We briefly introduce several common relationships:

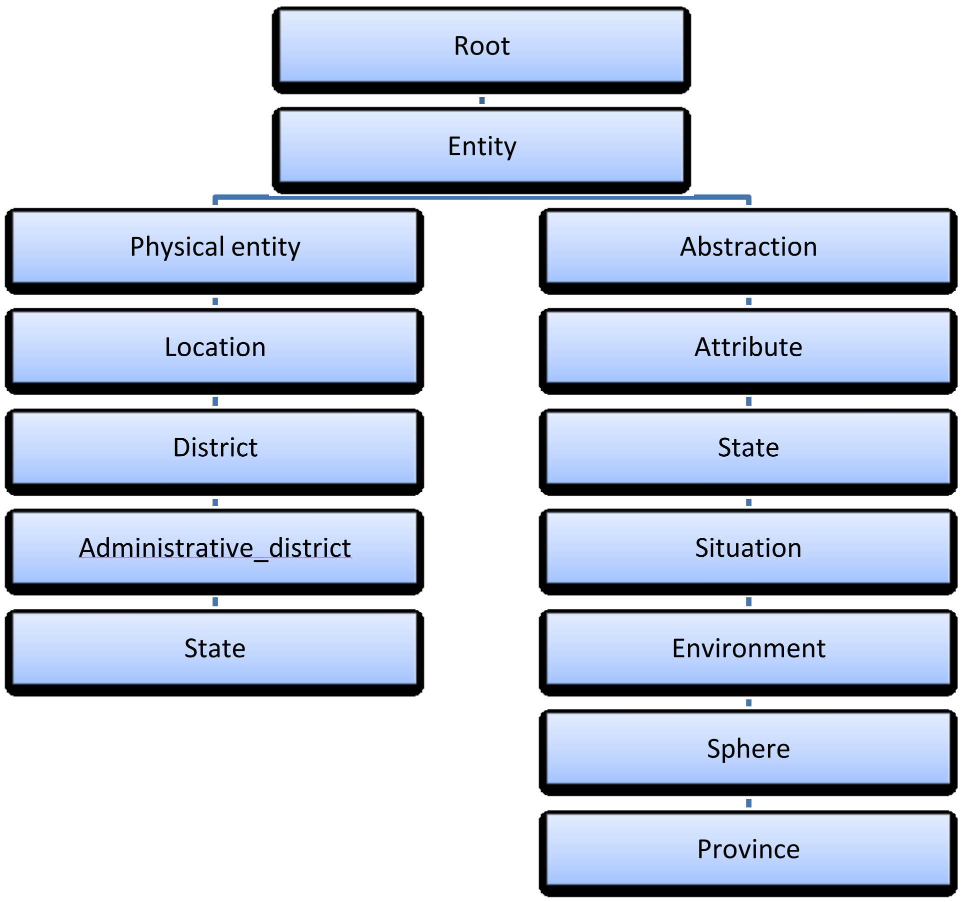


Figure 4.4 An illustration of WordNet's structure

Hypernym/hyponym ("is-a" relationship). X is the hypernym of Y if X is the more general direct link from Y. In contrast, Y is the hyponym of X. For an example, in **Figure 4.4**, "location" is hypernym of "district", and in contrast, "district" is the hyponym of "location".

Holonyms/meronyms ("a-part-of relationship"). X is the holonym of Y if X is the item contained in Y. In addition, X is the meronym of Y if X is one of the components or substances that make up Y.

Hence, synsets are linked based on the sense relationship between them, providing a hierarchical structure for computing semantic similarity. Next section will present the basic of semantic similarity and our method to compute the semantic similarity between two words.

4.2.2 Semantic similarity

To date, semantic similarity plays an important role to improve the quality of many data mining techniques [46], such as information retrieval, text classification, text clustering, and so on. In [47], a number of semantic similarity computation methods are summarized and categorized into four classes:

Path-based measures. The idea of this approach is calculating semantic between two synsets based on length of the path linking the synsets and the position of the concepts in the taxonomy. Examples for this approach are Wu & Palmer's measure [48], and Leakcock& Chodorow's measure [49].

Information content-based measures. This approach calculates the semantic similarity based on the information shared between two synsets, i.e., the common synsets shared by them. Examples of Information Content-based Measures are Resnik's measure [50], Lin's measure [51], and Jiang's measure [52].

Feature-based measure. Unlike methods above, this approach does not rely on the hierarchical structure of WordNet. To calculate the semantic similarity, the information stored in the synset, i.e, the definition or the set of "gloss". The example for this approach is Tversky's model [53].

Hybrid measure. This approach combine all ideas above, in addition utilizes other information in WordNet, i.e, synset relationship, to calculate the semantic similarity. The example for this category is Zhou's measure [54].

In this research, we simply apply path length measure implemented in package WordNet interface on Python (NLTK package) to calculate the semantic similarity [55]. The score is inversely proportional to the number of nodes along the shortest path between the synsets. The shortest possible path occurs when the two synsets are the same, in which case the length is 1. Thus, the maximum semantic similarity is 1.

However, the most important thing is to calculate the semantic similarity between words. Due to polynym problem (one word can be presented by different senses according to their definition in the context), it is necessary to identify the most appropriate sense of a word according to the context of it in document (word sense disambiguation process). However, this approach has several limitations. First, the quality of WSD is not reliable. According to [56], the accuracies of WSD methods are all below than 0.6, which is far from a reliable result. Second, most of WSD methods are heavy-computation, while we are aiming for a low-complexity method. Third, WSD may increase the number of dimensions, making the high dimensionality become more critical.

Another approach is using first k-synsets approach. The idea is using only first k-synsets for each word to calculate the semantic similarity between them. We denote synset(t, k) = {s_1, …, s_k} as the set of first k-synset for word t, and term_ss(x,y) as the semantic similarity between two synsets x and y; the semantic similarity $ss(t_i, t_j)$ between two words t_i, t_j is computed as:

$$\mathrm{ss}(t_i, t_j) = \max(\mathrm{term\_ss}(x, y) | x \in \mathrm{synset}(t_i, k), y \in \mathrm{synset}(t_j, k))$$

Based on the formula, the semantic similarity between two words t_i, t_j is the maximum value of semantic similarity between all pairs of synsets belonging to two set of synsets synset(t_i, k) and synset(t_j, k). In this research, we employ first k approach to calculate the semantic similarity between two words.

Based on the semantic similarity, we proposed SCF, a data processing algorithm, to transform term frequency matrix into a lower dimension space without losing the semantic information. Then, in order to improve the performance of clustering, SCF algorithm produces Semantic-related Conceptual feature space to represent the texts. The experiment result shows that SCF can greatly decrease the number of dimensions and improve the performance of clustering

4.3 Data preprocessing algorithm SCF

This chapter is to present an algorithm to reduce the number of dimensions by doing semantic-based features clustering and then extract key words and important semantic conceptual features in order to improve the quality of clustering. SCF algorithm has two phases: the first phase is doing pruning on to remove unnecessary words and replace semantically related words by a representative word. Next, in the second phase, we find keywords for each document and select important words for a better presentation of the content underlying the corpus. Finally, conceptual features are constructed to present the semantic similarity between keywords of each document and the concept identified based on covariance matrix. The flowchart of SCF algorithm is summarized in **Figure 4.5**. In this section, the scheme of SCF algorithm will be explained step by step.

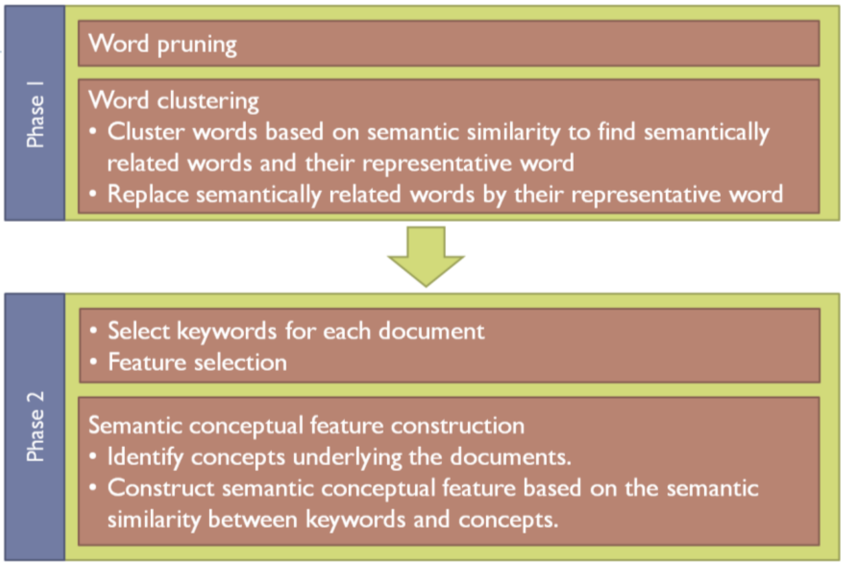

Phase 1
Word pruning
Word clustering
• Cluster words based on semantic similarity to find semantically related words and their representative word
• Replace semantically related words by their representative word
Phase 2
• Select keywords for each document
• Feature selection
Semantic conceptual feature construction
• Identify concepts underlying the documents.
• Construct semantic conceptual feature based on the semantic similarity between keywords and concepts.

Figure 4.5 Flowchart of SCF algorithm

4.3.1 Phase 1: Word pruning and clustering.

In this phase, unnecessary words are removed and replace semantically related words by a representative word in order to reduce the number of dimensions.

As we mentioned, short text data may contain a lot of misspellings and jargons, which will impact the quality of data. These words usually occur in a small number of documents, hence increase the sparseness of term frequency matrix. Words not included in WordNet do not have any synset in WordNet, hence the semantic similarities between it and other words are zero. Hence, it will not contribute to the semantic similarity of the document to any concept identified in the last step, and therefore, does not impact the final result. Therefore, we discards all words not included in WordNet in this step.

The next step is removing extreme-high document frequency words. Short text, i.e., emails, news, and forum posts may contain a lot of patterns (greetings, patterns, so on), signatures, and quotes, which are repeated texts and are irrelevant to the document topic. Words appeared in these texts add noise to the similarity between documents, hence degenerate the quality of clustering. To automatically detect those extreme high document frequency words, we perform clustering on document frequency values of all words to identify the cluster of extreme value. The algorithm is described in **Figure 4.6**. All words belong to the cluster whose mean's value is extreme high are discarded.

Next, we do clustering on the remaining words based on the semantic similarity between them. From the clustering result, clusters can reveal the groups of semantically related words and the centroid of the clusters is considered as the representative words for all the words belonging to that cluster. The clustering technique employed on this step should be able to deal with the problems following:

Computation complexity Due to the diversity of language, the number of words to be clustered can vary from thousands to hundreds of thousands. Hence, the algorithm should be simple and low complexity to reduce the processing time.

The number of clusters The algorithm should automatically identify the appropriate number of clusters.

Polynym As we mentioned, a word in WordNet is represented by several senses located in different locations on the ontology. Hence, a word may belong to different clusters (soft clustering problem).

```
Input:
BOW: bag of words
df: document frequency of al words in BOW
synset(w): set of synsets describing the word t in WordNet
Output:
BOW
Algorithm:
for each w in BOW
     if synset(w) = ∅
          Bow = Bow/{w}
initial_means = {min(df),average(df),max(df)}
C = {c1,c2,c3} = k-means(df, initial_means)
For each w in BOW
     if w ∈ c3 then
          Bow = BOW/{w}
```

Figure 4.6 Algorithm for removing words which have extreme high document-frequency.

We employ the algorithm described in **Figure 4.7** to clustering the remaining words based on the semantic similarity between them. The algorithm has low complexity O(n), can detect the centroid of clusters and allows one word to belong to different clusters. Hence, it can satisfy all the requirements we have mentioned.

Then, we create representative word frequency matrix (RWFM). The feature space of this matrix is representative words (centroids of the clusters) identified in the previous step. To calculate the frequency of representative words rw_j in the document d_i, we use the following formula:

$$RWFM_{ij} = \sum_{x \in cluster_j} (\mathrm{ss}(c_j, x) \times \mathrm{tf}(x, d_i))$$

From the formula, we can see that a frequency of a representative word can present the frequency and the semantic relationship of all the words belonging to the group of semantically related words to the representative word, and it means the original term frequency matrix is transformed into new feature vector space that can retain the information of frequency and semantic relationship.

```
Input:
ss: semantic similarity function
Th: threshold value to define semantically related words
tf(x, d): the number of occurrences of word x in document d
Output:
centroid: set of representative words
cluster: groups of semantically related words
Algorithm:
centroid = {}
k = 0
For each $t \in BOW$
    isolated = true
    for each $c_i \in centroid$
        if ss($c_i$, t) $\geq th$
            isolated = false
            $cluster_i = cluster_i \cup t$
            if $\sum_{x \in cluster_i} ss(t, x) > \left(\sum_{x \in cluster_i} ss(c_i, x)\right)$
                $c_i = t$
        end if
    end for
    if isolated
        $k = k + 1$
        $c_k = t$
        $cluster_k = \{t\}$
    end if
end for
```

```
for each $c_i \in centroid$
    for $c_j \in centroid$
        if $i \neq j$ & $(c_i = c_j)$
            $cluster_i = cluster_i \cup cluster_j$
            $centroid = centroid \setminus \{c_j\}$
            $cluster = cluster \setminus cluster_j$
        end if
    end for
end for
```

Figure 4.7 Clustering algorithm for clustering words based on semantic similarity

4.3.2 Phase 2: keywords and Semantic related Conceptual Feature (SCF) matrix construction

It is not necessary to use all the words or representative words to describe the main topics in the document. Actually, the main topics can be identified via several words, for examples, keywords in a scientific article, or tags in a news article. In this research, we define such words are keywords. To automatically identify the keywords, we firstly apply TF–IDF [57] (term frequency–inverse document frequency) to weight the representative words frequency matrix RWFM. Then, for each documents d_i, we apply k-means clustering (k=2, initial means are the minimum and maximum values) on the weighted term frequency of them to cluster representative words occurred in document d_i. All the representative words belonging to the cluster with higher mean of frequencies are considered as the keywords for documents d_i. Next we do feature selection to select important representative words. Denote $kwdf(d) = \{ kwdf_1,\ldots, kwdf_m\}$, with m is the number of representative words occur in document d and $kwdf_i$ is number of document in which the representative word rw_i is a keyword, as the set of important values of representative words, we perform k-means clustering (k=2, initial means are the minimum and maximum values) on the these values to find important representative words (representative words that belong to the cluster with higher mean of import values of representative words). The representative words which are not keywords in many documents (representative words that belong to the cluster with higher lower mean of import values of representative words) will not contribute much to the general topics in the corpus, and therefore, should be discarded. The method for identifying keywords in each-document and selecting important representative words is in described in **Figure 4.8**.

After identifying keywords and selecting important representative features, we discover the concepts (main topics) underlying the documents. The concept should contain representative words which are co-occurrence in many documents. Since we discarded all the representative words which are the keywords in several documents, all the remaining representative words have high document frequency. Hence, to find the concept, we only have to find the groups of representative words which highly co-occur (concepts). To find these groups we employ the algorithm described in **Figure 4.7** on the covariance matrix of important representative words to find the groups of high co-occurrence important representative words, which are the concepts underlying the corpus.

```
Input:
RWTF: representative words term frequency matrix
RW: set of representative words
rw(d): set of representative words occur in document d
Output:
$kw_i$: set of keywords for document $d_i$
IF: set of important feature
Algorithm:
RWTF = TF-IDF(RWTF)
for each $d_i$ in corpus
    $kw_i$ = {}
    initial_mean = {min(rw($d_i$)), max(rw($d_i$))}
    C = ($c_1$,$c_2$) = k-means(rw($d_i$), initial_mean, k = 2)
    for each t ∈rw($d_i$)
        if t ∈ $c_2$
            $kw_i = kw_i \cup t$
        end if
    end for
end for
for each $r_i$ ∈ RW
    $v_i = |\{d_j| \ r_i \in kw_j\}|$
initial_mean = {min(v), max(v)}
Cl = ($cl_1$,$cl_2$) = k-means(v, initial_mean, k = 2)
IF = {}
for each $r_i$ ∈ RW
    if $r_i$ ∈ $c_2$
        $IF = IF \cup r_i$
    end if
end for
```

Figure 4.8 Algorithm for identifying keywords in each-document and selecting important representative words

Finally, the Semantic related Conceptual Feature (SCF) matrix will be constructed based on the keywords of each document and the concepts underlying the corpus. We denote SCF_{ij} as the semantic similarity between the keywords of document d_i and the concept cf_j, and will be computed by the following formula:

$$SCF_{ij} = \frac{\sum_{k \in kw(d_i), c \in cf_j,} \mathrm{ss}(k, c)}{\max\left(|kw(d_i)|, l\right)}$$

where ss(k,c) is the semantic similarity between two words k and c, l is a threshold value to restrict the document has too small number of keywords.

The final matrix *SCF* only presents the semantic similarity between keywords of each document (which are representative word that important to present the content underlying the document and corpus) and the concepts underlying the corpus. Therefore, SCF should improve the performance of clustering.

4.4 Experiment result

In this chapter, we evaluate the performance of the proposed algorithm SCF and compare it other method.

4.4.1 Datasets and text processing

In our study, we use two short text corpuses to evaluate the proposed algorithm: Enron and 20 newsgroups corpus. These datasets are widely used for experiment in text mining research, such as text classification and text clustering. The Enron dataset contain about 619,446 emails belonging to158 users from Enron Corporation [58]. In order to employ this dataset for evaluating the performance of clustering algorithm, we use UC Berkeley Enron dataset [59], a subset of Enron dataset labeled by human. This subset contains 1702 emails and classify into 8 classes. However, since the class 7 and class 8 contain empty emails only, we only employ 1546 emails from class 1 to class 6 in this experiment. Since the emails in UC Berkeley Enron dataset are labeled by more than one people, some of them are classified into more than one class. We fix the label of a these emails based on the majority among the classified labels. The second dataset, 20 newsgroups dataset [60], contain approximately 20000 newsgroups belonging to 20 different classes. Some of the newsgroups are highly related to each other (i.e., baseball and hockey, autos and motorcycles), while others are distinguishable (i.e., religion and science). **Table 4.1** summaries the characteristic of these two datasets.

Table 4.1 Characteristics of UC Berkeley Enron and 20 newsgroups datasets

Name	**No. documents**	**No. classes**	**Size of clusters**	**No. nouns**	**No. verbs**
Enron	1546	6	727, 36, 92, 474, 74,143	5489	2212
20 newsgroups	19918	20	~500 for each	50399	12255

To do the text processing, we used NLTK package on Python [55] for tokenizing and doing POS-tagging and then selected all nouns and verbs. We used the first *k* synsets approach described in section 4.2.2 to calculate the semantic similarity. The value *k* is set to 2 due to following reasons: the experiment results in [61] show that *k*=1 or *k* = all (all synsets are picked for each word) do not improve the quality of clustering result. In [62], researcher indicated several reasons to select *k* not greater than three. To identify the most appropriate value for *k*, we carefully investigated the case of *k*=2 and *k*=3. For monitoring the difference between two cases, we calculated the similarities between noun-noun and verb-verb collected from a subset of 20 newsgroups dataset, which has greater number of topics to ensure the rich of context and diversity of language. This subset contains 50 first newsgroups from each group. The distributions of similarity matrices in two cases are shown in **Figures 4.9** and **4.10**. The two similar distributions show that the both choices do not affect much to the semantic relationship between words. Hence we focused only the number of highly related noun-noun and verb-verb pairs. In case of *k*=2, 9118 pairs of verb-verb senses and 16980 pairs of noun-noun senses have value equal or greater than 0.5 (hypernym/hyponym relationship). In case of *k*=3, the number rapidly increases: there are 13762 (150.93% compared to the case *k*=2) pairs of verb-verb and 24028 (141.51% compared to the case *k*=2) pairs of noun-noun senses have value equal or greater than 0.5. Hence, using *k*=3 may lead to inappropriate word representation and causes the loss of information. For the safety of doing word clustering, we applied first *k*-synsets approach with the value of *k* equal to two (for each word, first two senses are chosen) in this research.

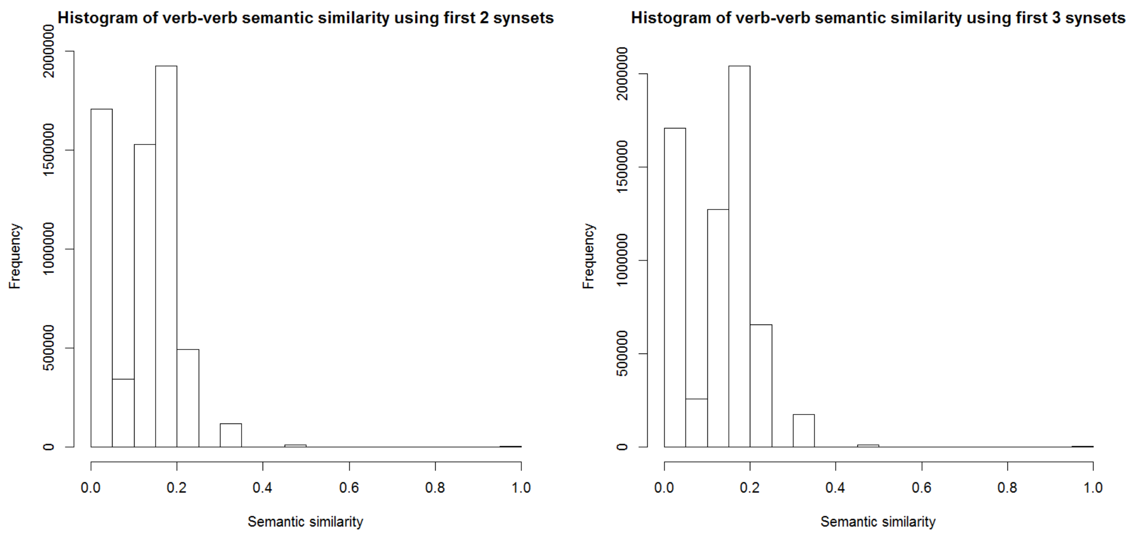


Figure 4.9 Histogram of semantic similarity between verb-verb using first *k* (*k*= 2 at the left, *k* = 3 at the right) concepts

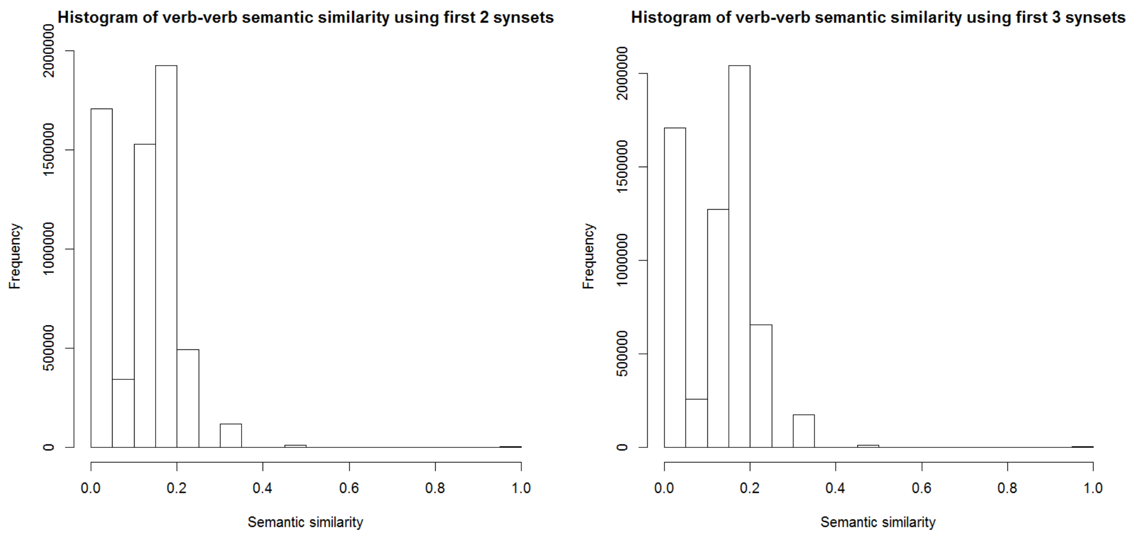


Figure 4.10 Histogram of semantic similarity between noun-noun using first *k* (*k*= 2 at the left, *k* = 3 at the right) concepts

4.4.2 Experiment result

We employed SCF algorithm on UC Berkeley Enron and 20 newsgroups datasets. For doing word clustering, the algorithm described in Figure 23 is applied with the value of the threshold *th* is set to 0.5 (hypernym/hyponym relationship). To find the concepts underlying the corpus, the algorithm described in **Figure 4.7** is employed with the value of the threshold *th* is set to 0.7. The reason we set this parameter is because of by analyzing the histogram of covariance matrix, we realized that most of the pairs of important representative words have co-occurrence value less than 0.7, hence we choose 0.7 as the value for the threshold *th* to identify high co-occurrence pairs.

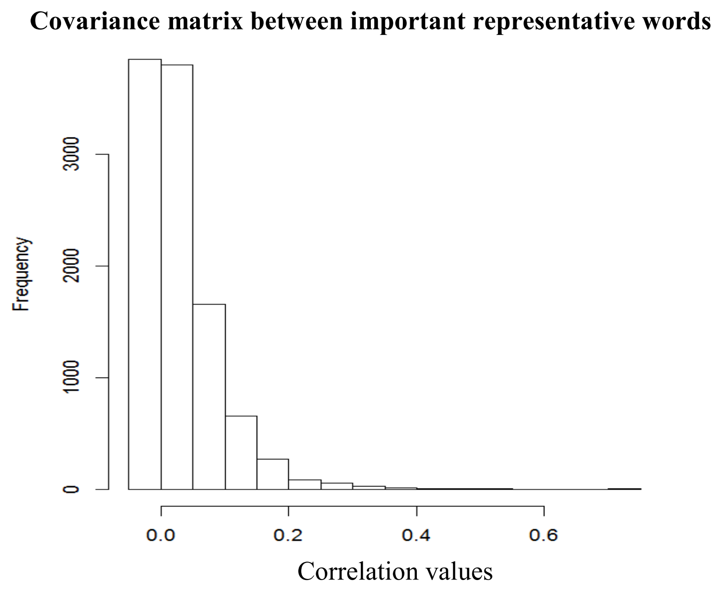


Figure 4.11 Histogram of values of covariance matrix between important representative words in Enron dataset.

The result of number reduction is showed in the **Table 4.2**. Both two methods word clustering and SCF algorithm can greatly reduce the number of feature. In case of UC Berkeley Enron dataset, by the effect of word clustering, number of representative verbs was reduced to 748 (66.18% of features are reduced) and the number of representative nouns was reduced to 1964 (64.21% of features are reduced). And then, SCF algorithm transforms representative words frequency matrix to low space: the space has 258 SCF features (68 concepts of important representative verbs and 190 concepts of important representative nouns), means 96.52% of features are reduced. In case of 20 newsgroups, word clustering decreases the number of representative verbs to 1613 (86.84% of features are reduced) and the number of representative nouns to 7006 (86.10% of features are reduced). SCF algorithm then transforms the space of 8619 representative words to 801 SCF features (209 concepts of important representative verbs and 592 concepts of important representative nouns), means 98.72% of features are reduced.

Table 4.2 Result of feature reduction by word clustering and SCF algorithm

Name	Words	Baseline	Phase 1	Phase 2	% reduction	
UC Berkeley Enron	Nouns	5489	1964	168	97.122	96.652
	Verbs	2212	748	90	95.931	
20 newsgroups	Nouns	50399	7006	592	98.825	98.722
	Verbs	12255	1613	209	98.294	

The results of clustering performed on the matrices produced by word clustering and SCF algorithm are showed in **Figure 4.12**. We employed *k*-means (and giving the correct number of clusters in both cases of UC Berkeley Enron and 20 newsgroups datasets) 10 times for each experiment on four matrices: the baseline (contains nouns and verbs after doing stopword removal), term frequency transformed by PCA (with the number of pc varied from 2 to 30), the RWFM (by word clustering), and SCF matrix. The clustering results are then evaluated by Rand Index. The results show that both word clustering can improve the performance of clustering compared to using the original term frequency matrix. In case of UC Berkeley Enron dataset, PCA degenerate the performance of clustering, because most of the topics in this dataset are highly related to a main theme: business. PCA improved the quality of clustering in case of 20 newsgroups dataset, however the improvement is smaller than the result of SFC algorithm.

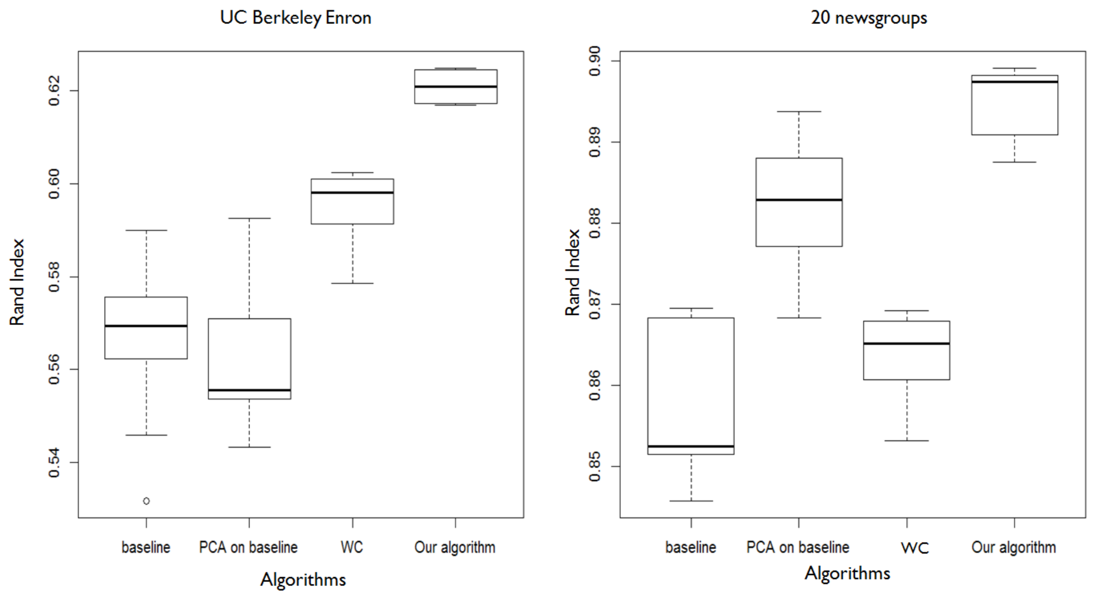


Figure 4.12 Comparisons of clustering performances on UC Berkeley Enron and 20 newsgroups datasets.

4.5 Conclusion

In this research, we proposed a data preprocessing algorithm named SCF to reduce the number of dimensions without losing the semantic information stored in each feature. The new space produced by SCF algorithm presents the semantic similarity between the keywords of each documents and the concept underlying the corpus, hence can present the content of the topics underlying the corpus clearer. The experiment results show that SCF algorithm can create a new space of a small number of features but can improve the performance of clustering result preformed on SCF matrix.

In the future, we would like to validate the effect of different semantic similarity functions, i.e., Wu & Palmer’s measure, Leakcock& Chodorow’s measure, Jiang’s measure, Tversky’s model, or Zhou’s measure, to find the best measure for word clustering and improve SCF algorithm.

CHAPTER V Conclusion

In this literature, we introduced to data preprocessing method named D-IMPACT and SCF. D-IMPACT focuses on removing noises/outliers and separating clusters based on moving data points. SCF algorithm focuses on reducing the number of features by doing word clustering and feature selection. Finally, semantic related conceptual feature matrix will be constructed based on the semantic similarity between keywords of each documents and the concepts underlying corpus. The experiment results clearly show effectiveness of both D-IMPACT and SCF algorithm.

In the future, we can improve the algorithm D-IMPACT by employing new formulas to compute the density, attraction and vectors in data objects moving phase. This can help D-IMPACT processes different types of dataset effectively.

Similar to D-IMPACT, we would like to validate the effect of different semantic similarity functions, i.e., Wu & Palmer's measure, Leakcock& Chodorow's measure, Jiang's measure, Tversky's model, or Zhou's measure, to find the best measure for SCF algorithm.

Supplementations

Currently, the algorithm D-IMPACT are implemented in C++ without GUI but can be used easily in command mode. The manual file contains the guide and several examples of using D-IMPACT to cluster different types of datasets. The program can be downloaded at https://sourceforge.net/projects/dimpactpreproce/.

Some of the datasets used in this literature are our synthesis data. Collected datasets are cited and included the download link.

For the source code of text processing and SCF algorithm in chapter IV, please contact us via email.

Bibliography

[1] Berkhin, P. "Survey of clustering data mining techniques." Technical report, Accrue Software, San Jose, CA, 2002.

[2] Murty, M.N., Jain, A.K., Flynn, P.J. "Data clustering: A Review." *ACM Computing Surveys*, **31(3)**, 1999, 264-323.

[3] Halkidi, M., Batistakis, Y., Vazirgiannis, M. "On Clustering Validation Techniques." *Journal of Intelligent Information Systems*, **17**, 2001, 107-145.

[4] Golub, T.R., et. al. "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring." *Science*, **286 (15)**, 1999, 531–537.

[5] Quinn, A. "Survey of Techniques for Preprocessing in High Dimensional Data Clustering," *Proceedings of the Cybernetic and Informatics Eurodays*, 2000.

[6] Kaufman, L., Rousseeuw, P.J. *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley & Sons, 1990.

[7] Guha, S., Rastogi, R., Shim, K. "CURE: An Efficient Clustering Algorithm for Large Databases." *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 1998, 73-84.

[8] Ester, M., Kriegel, H.P., Sander, J., Xu, X. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, 1996, 226–231.

[9] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications." *Proceedings of the ACM SIGMOD Conference*, 1998, 94-105.

[10] Wang, W., Yang, J., Muntz, R. "STING: A Statistical Information Grid Approach to Spatial Data Mining", *Proceedings of the 23rd Conference on VLDB*, 1997, 186-195.

[11] Abdi, H., Williams, L.J. "Principal component analysis." *Wiley Interdisciplinary Reviews: Computational Statistics*, **2(4)**, 2010, 433–459.

[12] Yeung, K.Y., Ruzzo, L.W. "An empirical study on principal component analysis for clustering gene expression data." *Bioinformatics*, **17**, 2001, 763-774.

[13] Alelyani, S., Tang, J., and Liu, H. "Feature Selection for Clustering: A Review." *Data Clustering: Algorithms and Applications*, **29**, 2013.

[14] Hodge, V.J., Austin, J. "A survey of outlier detection methodologies." *Artificial Intelligence Review*, **22(2)**, 2004, 85-126.

[15] Tran, V.A., et al. "IMPACT: A Novel Clustering Algorithm based on Attraction." *Journal of Computers*, **7(3)**, 2012, 653-665.

[16] Shi, Y., Song, Y., Zhang, A. “A shrinking-based clustering approach for multidimensional data.” *IEEE Trans. Knowl. Data Eng.*, **17(10)**, 2005, 1389 – 1403.

[17] Chang, F., Qiu, W., Zamar, R.H. ”CLUES: A Non-Parametric Clustering Method Based on Local Shrinking." *Computational Statistics & Data Analysis*, **52(1)**, 2007, 286-298.

[18] Karypis Lab Datasets. http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/chameleon-data.tar.gz.

[19] The UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets.

[20] Data Sets for Short-text Experimental Works.

https://sites.google.com/site/merrecalde/resources.

[21] Karypis, G., Han, E. H., Kumar, V. “CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling.” *Computer*, **32(8)**, 1999, 68-75.

[22] Hubert, L., and Arabie, P. “Comparing Partitions.” *Journal of Classification*, **2(1)**, Springer, 1985, 193–218.

[23] Ingaramo, D., Errecalde, M., and Rosso, P. "A general bio-inspired method to improve the short-text clustering task." *Computational Linguistics and Intelligent Text Processing*, Springer Berlin Heidelberg, 2010, 661-672.

[24] Errecalde, M., Ingaramo, D., and Rosso, P. “ITSA*: An Effective Iterative Method for Short-Text Clustering Tasks”, *Proceedings of IEA/AIE*, 2010, 550-559.

[25] Tran, V.A., et al. “D-IMPACT: A Data Preprocessing Algorithm to Improve the Performance of Clustering.” *Journal of Software Engineering and Applications*, **7**, 2014, 639-654.

[26] Radioresistant and radiosensitive tumors and cell lines. http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE9712.

[27] Chang, F., Qiu, W., Zamar, R.H., Lazarus, R., Wang, X. "clues: An R Package for Nonparametric Clustering Based on Local Shrinking." *Journal of Statistical Software*, **33(4)**, 2010, 1-16.

[28] Visakh, R., and Lakshmipathi, B. "Constraint based Cluster Ensemble to Detect Outliers in Medical Datasets," *International Journal of Computer Applications*, **45(15)**, 2012, 9-15.

[29] Osiński, S., and Weiss, D. "Carrot2: Design of a flexible and efficient web information retrieval framework." *Advances in Web Intelligence*, Springer Berlin Heidelberg, 2005, 439-444.

[30] Sarwar, B.M., et al. "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering." *Proceedings of the fifth international conference on computer and information technology*, **1**, 2002.

[31] Radev, D.R., Fan, W., and Zhang, Z. "Webinessence: A personalized web-based multi-document summarization and recommendation system." *Ann Arbor* 1001, 2001.

[32] Pazzani, M.J., and Billsus, D. "Content-based recommendation systems." *The adaptive web*, Springer Berlin Heidelberg, 2007, 325-341.

[33] Lee, H., et al. "iVisClustering: An interactive visual document clustering via topic modeling." *Computer Graphics Forum*, **31(3 pt3)**, Blackwell Publishing Ltd, 2012.

[34] Lin, D. and Pantel, P. "Concept discovery from text." *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, 2002.

[35] Aggarwal, C.C., and Zhai, C. X. "A survey of text clustering algorithms." *Mining Text Data*, Springer US, 2012, 77-128.

[36] Rose, J.D., and Mukherjee, S. "An Approach for Text Clustering Using Modified K-means Algorithm." *International Conference on Software Technology and Engineering* (*ICSTE 2012*), ASME Press, 2012.

[37] Spanakis, G., Siolas, S., and Stafylopatis, A. "Exploiting Wikipedia knowledge for conceptual hierarchical clustering of documents." *The Computer Journal*, **55(3)**, 2012, 299-312.

[38] Sakai, T., Tamura, K., and Kitakami, H. "Extracting Attractive Local-Area Topics in Georeferenced Documents using a New Density-based Spatial Clustering Algorithm." *IAENG International Journal of Computer Science*, **41(3)**, 2014.

[39] Amine, A., Elberrichi, Z., and Simonet, M. "Evaluation of text clustering methods using wordnet." *Int. Arab J. Inf. Technol*, **7(4)**, 2010, 349-357.

[40] Bouras, C., and Tsogkas, V. "A clustering technique for news articles using WordNet." *Knowledge-Based Systems*, **36**, 2012, 115-128.

[41] Dang, Q., et al. "WordNet-based suffix tree clustering algorithm." *2013 International Conference on Information Science and Computer Applications (ISCA 2013)*, Atlantis Press, 2013.

[42] Deerwester, S. "Improving Information Retrieval with Latent Semantic Indexing". Proceedings of the 51st Annual Meeting of the American Society for Information Science **25**, 1988, 36–40.

[43] Miller, G.A. "WordNet: a lexical database for English." *Communications of the ACM*, **38(11)**, 1995, 39-41.

[44] Dhungana, U.R., et al. "Word Sense Disambiguation using WSD specific WordNet of polysemy words." *Semantic Computing (ICSC), 2015 IEEE International Conference on*. IEEE, 2015, 148-152.

[45] Chen, J., and Liu, J. "Combining ConceptNet and WordNet for Word Sense Disambiguation." *IJCNLP*, 2011, 686-694.

[46] Varelas, G., et al. "Semantic similarity methods in wordNet and their application to information retrieval on the web." *Proceedings of the 7th annual ACM international workshop on Web information and data management*, ACM, 2005, 10-16.

[47] Meng, L., Huang, R., and Gu, J. "A review of semantic similarity measures in wordnet." *International Journal of Hybrid Information Technology*, **6(1)**, 2013, 1-12.

[48] Wu, Z. and Palmer, M. “Verb semantics and lexical selection”, *Proceedings of 32nd annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 1994.

[49] Leacock, C., and Chodorow, M. "Combining local context and WordNet similarity for word sense identification." *WordNet: An electronic lexical database*, **49(2),** 1998, 265-283.

[50] Resnik, P. “Using information content to evaluate semantic similarity in a taxonomy”, Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995.

[51] Lin, D. "An information-theoretic definition of similarity." *ICM*, **98,** 1998.

[52] Jiang, J.J., and Conrath, D.W. “Semantic similarity based on corpus statistics and lexical taxonomy”, *Proceedings of 10th International Conference on Research in Computational Linguistics, ROCLING’97*, 1997.

[53] Tversky, A. “Features of Similarity”, *Psycological Review*, **84(4)**, 1977, 327-352.

[54] Zhou, Z., Wang, Y., and Gu, J. "New model of semantic similarity measuring in wordnet." *Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on*, **1**, IEEE, 2008.

[55] Perkins, J. *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing Ltd, 2014.

[56] Wang, T., and Hirst, G. "Applying a Naive Bayes Similarity Measure to Word Sense Disambiguation." *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, June*, 2014.

[57] Berger, A., and Lafferty, J. "Information retrieval as statistical translation." *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 1999, 222-229.

[58] Klimt, B., and Yang, Y. "Introducing the Enron Corpus." *CEAS,* 2004.

[59] U C Berkeley Enron Email Analysis. http://bailando.sims.berkeley.edu/enron_email.html

[60] 20 newsgroups. http://qwone.com/~jason/20Newsgroups/

[61] Hotho, A., Staab, S., and Stumme, G. "Ontologies improve text document clustering." *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, IEEE, 2003.

[62] Fodeh, S., Punch, B., and Tan, P. N. "On ontology-driven document clustering using core semantic features." *Knowledge and information systems*, **28(2)**, 2011, 395-421.