Textmining and Organization in Large Corpus

Wei Ning

Kongens Lyngby 2005

Abstract

Nowadays a common size of document corpus might have more than 5000 documents. It is almost impossible for a reader to read thought all documents within the corpus and find out relative information in a couple of minutes. In this master thesis project we propose text clustering as a potential solution to organizing large document corpus.

As a sub-field of data mining, text mining is to discover useful information from written resources. Text clustering is one of topics in text mining, which is to find out the groups information from the text documents and cluster these documents into the most relevant groups automatically. Representing document corpus as a term-document matrix is the prevalent preprocessing in text clustering. If each unique term is taken as a dimension, a common size of corpus may contain more than ten-thousands of unique term, which results in extremely high dimensionality. Finding good dimensionality deduction algorithms and suitable clustering methods are the main concerns of this thesis project.

We mainly compare two dimensionality deduction methods: Singular Vector Decomposition (SVD) and Random Projection (RP), and three selected clustering algorithms: K-means, Non-negative Matrix Factorization (NMF) and Frequent Itemset. These selected methods and algorithms are compared based on their performance and time consumption.

This thesis project shows K-means and Frequent Itemset can be applied in large corpus. NMF might need more research on speeding up its convergence speed.

Preface

This thesis is submitted to fulfill the requirements of the Master of Science in Computer System Engineering. The project was done by Wei Ning during the period July 2005 to December 2005 at the department of Informatics and Mathematical Modelling (IMM), Technical University of Denmark (DTU). The work was supervised by Professor Jan Larsen.

Acknowledgments

I would like to thank my supervisor Jan Larsen for his assistance throughout the thesis. He assisted me on all the stages of my thesis and always gave me good idea and helpful instruction. Meanwhile, I would like to thank Professor Lars Kai Hansen for his giving me ideas on clustering algorithms. And thank Ph.D. students Rasmus Elsborg Madsen, Anders Meng, Morten Mrup, Ling Feng, Finn arup Nielsen for their help during my research.

Special thanks to my parents and my friends for their support.

Lyngby, December 2005

Wei Ning

Contents

Chapter 1

Introduction

This thesis represents the written part of the Master's Thesis project: Textming and Organization in Large Corpus. This project concerns the problems of organizing large document corpus and outline the feasibility to apply text clustering in large document corpus.

The major contributions of this project are:

- the feasibility of applying Random Projection on dimensionality deduction of text data
- the feasibility of applying Non-negative Matrix Factorization as text clustering techniques on large document collection
- the feasibility of applying Frequent Item Set as text clustering techniques on large document corpus

1.1 Problem Definition

Large document corpus may afford a lot of useful information to people. But it is also a challenge to find out the useful information from huge number of documents. Especially with the explode of knowledge around the cyber-world, corporates and organizations demand efficient and effective ways to organize the large document corpus and make later navigating and browsing become more easy, friendly and efficient.

An obvious characteristic of large document corpus is the huge volumes of documents. It is almost impossible for a man to read through all the documents and find out the relative for a specific topic. How to organize large document corpus is the problem we concern.

1.2 Project Scope

This master thesis project is to evaluate the feasibility to apply text clustering algorithms in organizing large document corpus.

The aim of this project was initiated to investigate the possibility of clustering huge numbers of documents into groups. Hereby we mainly concern the most challenging issue of large document corpus: huge number of documents. And we compare and find feasible clustering algorithms which may achieve good performance with highly efficiency when dealing with thousands of documents.

1.3 Large Document Corpus Characteristics

Before we start discussing text clustering, we should define the characteristics of large document corpus. Large document corpus may have the following characteristics [4]:

- huge number of documents: as we consider of applying text mining on large corpus, we expect a corpus has at least 5000 documents.

- high dimensionality: if a unique term is considered as one dimension, 5000 documents may contain more than 100000 unique terms, therefore the term-document matrix for this documents corpus will be a 100000×5000 matrix. The high dimensionality matrix bring memory and time complexity challenges to later clustering algorithms.

- ambiguity: synonyms and words with multiple meaning are very common in documents. Documents content can be very close though words they use are quite different because those words are synonyms. Meanwhile even some documents use the same words but their content are quite different. Therefore determining document similarity only based on words they use is not sufficient.

- noisy data: spelling mistakes, abbreviations and acronyms are unavoidable in large document collections. All these kinds of issue may introduce noise into the corpus.

Considering the characteristics of the large document corpus, this thesis project is to compare dimensionality deduction methods and clustering algorithms that might deal with the high dimensionality issue. And we summarize the feasible approaches that can be applied to large document corpus with efficiency and effectiveness.

1.4 Report Structure

In Chapter 2: **Background** we give an introduction to the background of text mining and organization in large data corpus. First we introduce the basic concept of text mining and document clustering. And then we outline the background knowledge of text clustering including the document representation model, taxonomy of clustering methods and clustering processes. In the end we give a short description about the current prevalent text clustering methods.

In Chapter 3 **Preprocessing** we outline the preprocessing steps applied on documents before they are clustered.

Removing stop-words and stemming are two very important preprocessing steps. Basically stop-words are not useful for further clustering algorithm. By removing stop-words unnecessary features are removed and noise is deduced. Stem-

ming serves for the similar purpose meanwhile it might help the clustering algorithms discover the similarity between documents.

Term weighting is another important preprocessing step. In a feature-document matrix each unique term is a feature. Important meaningful terms might be more valuable than less meaningful terms. Thereby it is necessary to have a term weighting method to give different term with different weighting.

In Chapter 4 **Feature Extraction and Matrix Dimensionality Reduction** We discuss two methods to extract the key features from the corpus and reduce the dimensionality: Singular Vector Decomposition and Random Projection.

Singular Vector Decomposition (SVD) is a very popular feature extraction approach with strict mathematical computations. And it has been proved that it is able to extract the latent features and reduce the dimensionality.

Random Project is another approach to reduce the dimensionality. Compared with SVD, random project is computationally simple but its result might be unstable because of it projects the original data matrix into a low-dimensionality subspace by random.

In Chapter 5 **Clustering Algorithm**: We give an introduction to three selected clustering algorithms, including their principle, algorithm details, complexity, advantages and weakness. These three clustering algorithms are K-Means, Non-negative Matrix Factorization and Frequent Itemset.

K-Means is considered as one of the standard clustering algorithms because of its effectiveness and simpleness. Each sample is considered as a data point in the feature space. K-means algorithm groups the data points based on the distance between each other. Though straightforward and powerful K-means does not organize the cluster hierarchy.

Non-negative Matrix Factorization (NMF) is a matrix factorization algorithm to find out the positive factorization of a given positive matrix. Like SVD, NMF might be able to discover the hidden patterns from the feature matrix. NMF might surpass SVD in that the clusters are determined from the factorization result of NMF.

Frequent Itemset mining is an approach to find frequent itemsets in a transactional database. It helps to find out the relevant product items in a supermarket from the supermarket transactions. When applied on text cluster, each document is considered as a transaction and each term is taken as an item. Frequent itemset is a promising clustering technique for text clustering.

In Chapter 6 **Validation and Evaluation**: we introduce one ways to calculate the relative merits of clustering structures and three measures, Entropy, F-Measure and Overall Similarity, to evaluate the performance of the chosen clustering algorithms.

Entropy and F-Measure are both external measures which require the document label information as input to evaluate the performance of the clustering algorithms. Overall similarity is an internal measure and it can be used to compare the cluster cohesiveness in the absence of document label information.

In Chapter 7 **Experiment Data and Result** we introduce the document corpus on which we apply the selected cluster algorithms. There are two documents corpus used in this thesis project: 20 Newsgroup and Reuters. We assess the performance and the time consumption of the key feature extraction approaches and the clustering algorithms.

In Chapter 8 **Summary and Future Work** we summarize our findings and contributions in thesis project. Meanwhile we outline the directions for further research.

Chapter 2

Text Clustering Background

Clustering and classifying are important ways for people to get to know new things and organize the existing knowledge. Grouping the similar objects together can make them easy to locate and filter. With the help of computers, the grouping process could be done automatically with highly efficiency and effect. In this chapter we introduce to the overview of text mining and text clustering.

2.1 Text Mining, Clustering and Classification

"Text Mining is the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources. " [2] Text mining techniques are the fundamental and enabling tools for efficient organization, navigation, retrieval and summarization of large document corpus [18]. With more and more text information are spreading around on Internet, text mining is increasing in importance. Text clustering and text classification are two fundamental tasks in text mining.

Text Clustering is to find out the groups information from the text documents and cluster these documents into the most relevant groups. Text cluster-

ing groups the document in an unsupervised way and there is not label or class information. Clustering methods have to discover the connections between the document and then based on these connections the documents are clustered. Given a huge volumes of documents, a good document clustering method may organize those huge numbers of documents into meaningful groups, which enable further browsing and navigation of this corpus be much easier [18]. A basic idea of text clustering is to find out which documents have many words in common and place these documents with the most words in common into same group.

Text Classification is to organize documents into predefined categories with meaningful labels. As text classification needs the information about those predefined categories, it is applied in a supervised way.

In this thesis project we suppose there is no category information before hand. So we concentrate on text clustering approaches.

2.2 Document Representation Model

Clustering is to discover the similarity between samples based on their characteristics. Before clustered, the samples have be represented with characteristics in some measurements which can be understood by computers. And then the most relevant characteristics are picked out and the similarities are calculated based on these characteristics.

2.2.1 Document Sample

Sample Type In general one sample could be described by features. According to the properties of the features, there are different ways to process different features[42]:

- Numeric measurement: uses some real numbers to present the values of the features.
- Ordinal measurement: uses some discrete values to present the values of the features. These discrete values are not real values but present

the order. One example could be that the level of the grade includes: Excellent, Good, Normal, Pass and Fail.

- Naming measurement: uses some discrete values, which does not present real values nor the order. For example the gender of human being includes male and female.

The Definition of Document Sample The most suitable measurement for document is the numeric measurement. A very straightforward way is that each document may be represented by a term vector in which each unique term is an independent feature. Each element in this vector stands for the occurrence of the corresponding term.

2.2.2 Boolean Model

Long before the Internet, information retrieval has already existed. And the boolean retrieval model is the most simple of these retrieval methods and relies on the use of Boolean operators [3]. Within the Boolean Model, a document is represented as a set of boolean values each of which repents whether a specific term presents in the document: normally a 1 means present and a 0 means not present.

With the Boolean Model, the criteria for searching a document is specified by a query within which the terms are linked with AND, OR and NOT these kinds of boolean operators. Because of the nature of boolean algorithms, the Boolean Model has the following advantages:

- The searching is fast and the implementation is simple and straightforward.
- Boolean Algebra can be applied to the Boolean Model.

Boolean model has its inherent weakness. One major issue is that an incorrect query may make the relevant documents non relevant [3]. Because a boolean value is used to identified a term present in a document or not, this results in that whether a document is relevant to the query is a binary decision. A wrong word in the query could rank a relevant document non relevant [3]. Meanwhile, because boolean model can only tell if a term presents in the documents or

not, it does not include further information about how importance of presenting words in the document. Therefore partial matching and ranking become very difficult.

2.2.3 Vector Space Model

In vector space model a document is represented by a vector which contains the term weighting information for the document. And then a collection of documents can be represented by a vector space [40].

In the simplest form each document can be represented by a term frequency (TF) vector:

$$d_{tf} = (tf_1, tf_2, tf_3, ..., tf_n)$$

where D_{tf} is the document vector and tf_i is the frequency of the ith term in this document.

Then a document collection can be represented by a TF matrix:

$$D = (d_1^T, d_2^T, d_3^T, ..., d_m^T)$$

where the ij entry indicates the frequency of the ith term in the j document and each d_j denotes a document vector.

Besides considering the term frequency within one document, it is useful to take into account the term frequency within the whole document corpus. In one document, a frequent presenting term probably is not as important as another less frequent one because this term is too frequent within the whole document corpus, which makes it contribute less to differentiate the document from the rest. So in the vector space model we consider the term weighting instead of only the term frequency.

There are two main steps to construct the vector space model [3]:

1. Document Indexing: not all the words within the document describe the content, and some very common words like *a, an, the* should be removed;

several words describe the similar content but with different formats, and this similar content bearing words should be formed into the same spell. Those two processes are called stop-word removal and stemming.

2. Term Weighting: based on the term statistics within the document and the whole document collection, a term weight is given to a weighting which denotes how important the term contributes to differentiate the document from others.

Compared with boolean model, vector space model does provide more information about the documents. But as in [5] Garcia outlined that there are following limitations with vector space model:

- calculation intensive: from the computational standpoint it is very slow, requiring a lot of processing time.
- not optimal for update: each time a new term is into the term space, all vectors have to been recalculated again. Thereby algorithms based vector space model might have the issue of online update.
- high dimensionality: vector space model takes each unique term as one dimension. This leads to extremely high dimensionality and complex calculations.

Meanwhile vector space model is a "bag of words" method which dose take the words sequence into account though words sequence is very important in represent the content of a document. It might lose some important connections between documents.

To overcome the weakness of vector space, the potential solutions could be to

- use some keyword-sets to represent the document: this can lead to much lower dimensionality.
- use n-grams: an n-gram is an n-word sub-phrase often occurring in natural language [21]. N-grams to some extent are words sequence and could provide more meaning information about documents than single words.

2.3 Similarity Measurement

In essence, document clustering is to group documents based on how relative they are. To cluster documents correctly, it is very important to measure how much a document is relative to another. And the extent of relativeness should be some real numbers and can be compared.

There are two prevalent ways to measure how one document is close to another [42]:

- Similarity: similarity measures to how much extent a document is similar to another.
- Distance: distance measures how much a document is away from another one. If a document is more close to anther than any others, it is considered that these two documents are similar to each other than any others.

Similarity Measurement The most common way to compute the similarity between documents is the cosine measure. If considering x and y are two document vectors, the cosine measure is defined as [13]:

$$cosine(x, y) = \frac{(x \bullet y)}{\|x\| \, \|y\|} \tag{2.1}$$

where $\bullet$ indicates the vector dot product and $\|x\|$ is the length of vector x.

Distance Measurement In [42] distance measurement is defined as:

Given S is a collection of samples, if measurement $D : S \times S \rightarrow R$ is a distance measurement if it satisfies the following requirements:

1. $D(X, Y) \geq 0$
2. $D(X, X) = 0$
3. $D(X, Y) = D(Y, X)$
4. $D(X, Y) + D(Y, Z) \geq D(X, Z)$

Minkowski distance is a widely used distance measurement clustering analysis [42]:

$$D(x,y) = (\sum_i |x_i - y_i|^q)^{\frac{1}{q}}$$

where x and y represent two document vectors and x_i and y_i are the corresponding elements in them.

When $q = 1$, it is absolute value distance.
when $q = 2$, it is Euclid distance.
when $q \to \infty$, it is Chebyshev distance.

Euclidean distance is widely used in clustering algorithm because of its simpleness and straightforwardness. It has an advantage that the distance is preserved when conducted orthogonal transformation [42], which might be useful when applying transformation on matrix.

Cluster Centroid Given a set S of the documents and their corresponding vector representations:

$$S = (d_1^T, d_2^T, d_3^T, ..., d_m^T)$$

the centroid vector c is defined as

$$c = \frac{\sum_{d=1}^{m} d}{|S|}$$

which is obtained by averaging the weights of all terms in documents of S.

2.4 Document Clustering Categories

Based on the their characteristics text clustering can be classified into different categories.

The most common classifications are hierarchical clustering and flat clustering. Depending on when to perform clustering or how to update the result when new documents are inserted there are online clustering and offline clustering. And according to if overlap is allowed or not there are soft clustering and hard clustering. Based on the features that are used, clustering algorithms can be grouped to document-based clustering and keywords-based clustering.

2.4.1 Hierarchical and Flat Clustering

Hierarchical and flat clustering methods are two major categories of clustering algorithms.

Just like departments in a company may be organized in a hierarchical style or a flat one, clusters of a document corpus may be organized in a hierarchical tree structure or in a pretty flat style.

2.4.1.1 Hierarchical Clustering

Hierarchical clustering techniques produce a nested sequence of partitions, with a single, all inclusive cluster at the top and singleton clusters of individual points at the bottom [13]. The hierarchical clustering result can be viewed as an up-side-down tree: the root of the tree is the highest level of clusters, the leaves of the tree are the lowest level clusters which are the individual documents, and the branches of the tree are the intermediate level in the clustering result. Seeing from different level might get different overview of clusters. For instance in figure 2.1:

- If seeing from level value 1, all documents are clustered into only one group;
- If from level value 0.8, documents are clustered into two group $G1$ and $G2$. Where $G1$ includes documents $D1$, $D2$, $D3$ and $D4$; $G2$ includes $D5$ and $D6$.
- If from level value 0.6, $G1$ can be divided into two sub-clusters $G11$ and $G12$. And then documents are clustered into three groups $G11$, $G12$ and $G2$ which respectively contain $D1$ and $D2$, $D3$ and $D4$, $D5$ and $D6$.
- When from a lower level (value 0.4), each document denotes one cluster.

Basically there are two approaches to generate such a hierarchical clustering [13]:

- Agglomerative: Start from and leaves, and consider each document as an individual cluster at the beginning. Merge a pair of most similar clusters until only one single cluster is left.

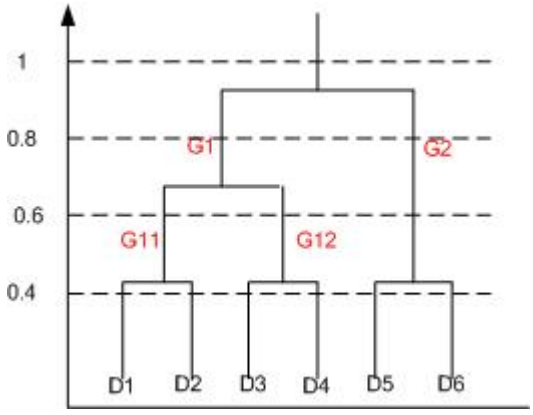

Figure 2.1: Hierarchical Clustering

- Divisive: Start from the root, and consider the whole document set as a single cluster. At each step divide a cluster into two (or several) subclusters until each cluster contains exactly one document or until the required number of clusters is archived.

Agglomerative techniques are relatively more common: it is quite straightforward and most common distance calculation and similarity measurement techniques can be applied. Traditional agglomerative hierarchical clustering steps can be summarized as the following [13]:

Given a collection of documents $D = d_1, d_2, ..., d_n$

1. Consider each document as an individual cluster. Compute the distance between all pairs of clusters and construct the $n \times n$ distance matrix D in which d_{ij} denotes the distance between cluster i and cluster j.
2. Merge the closest two clusters into a new cluster
3. Update the distance matrix: calculate the distance between the new generated cluster and the rest clusters.
4. Repeat step 2 and 3 until only one single cluster remains, which is the root cluster of the hierarchy

To generate a flat partition of clusters, a cut is made at the specific level of the hierarchical cluster tree, and on that level each branch represents a cluster and

all the leaves (documents) under the same branch belong to one cluster.

Agglomerative techniques need to consider which inter-cluster similarity measures to use [1]:

- single-link measure: join the two clusters containing the two closest documents.
- complete-link measure: join the two clusters with the minimum "most-distant" pair of documents.
- group average: join the two clusters with the minimum average document distance.

2.4.1.2 Flat Clustering

Different from hierarchical clustering, flat clustering creates a one-level (unnested) partitions [13] of documents instead of generating a well organized hierarchical cluster tree. Normally flat clustering techniques demand the expected number of clusters K as an input parameter, start with a random partitioning and then keep refining until algorithms converge. The convergence state is the final state that all clusters are stable and no more documents are switched between clusters.

Similarly flat clustering techniques may also create hierarchical cluster tree. By repeating the flat clustering techniques from the top level (root) of the tree to the lowest level (the leaves), a hierarchical cluster tree can be generated.

Hierarchical and flat clustering have their own advantages and weaknesses: Hierarchical clustering provides more detail about the whole document corpus, in which clusters are well organized in a tree structure. The price is the relatively higher complexity. On the contrary flat clustering techniques are normally simple and easy to implement. They could be applied with more efficiency when comparing with hierarchical clustering techniques.

When dealing with large document corpus, efficiency is the major issue we concern. In this thesis project we mainly consider flat clustering techniques. We

[1] adapted from [21]

also evaluate a hierarchical clustering algorithm in this thesis project because hierarchical clustering might give more help in knowing the structure and relation in a large document corpus than flat clustering.

2.4.2 Online and Offline Clustering

According to when clustering is performed, clustering algorithms can be divided into online clustering algorithms and offline clustering algorithms [21].

Online clustering algorithms perform document clustering when receiving the request and return the request within a limited period. It is obvious that online clustering demands very fast operations (low complexity) and make the clustering result up-to-date. Normally online clustering algorithms are applied on small or medium corpus.

Offline clustering, on the contrary, processes the documents and groups them into relevant clusters before receiving the request. When a request is received, offline clustering algorithms perform a few simple operations and then represent the clustering result. Compared with online clustering, offline clustering performs most of the operations before receiving the requests, it is relatively complex (high complexity) and can be applied on large document corpus. The major disadvantage of offline clustering is that the clustering result is not up-to-date. Sometimes it cannot reflect the fact that if a single document or a few documents are added into the corpus before most operations are applied in a long period of time.

Online clustering and offline clustering have their different applications: the former is normally applied to group the search results and the latter is to organize the document corpus.

A clustering algorithm is also classified as online clustering if it only updates the necessary documents in the corpus instead of re-clustering all documents when new documents are added into the document corpus. Given an existing document corpus and the clustering result, when new documents are added into the document collection, online clustering algorithms only apply clustering calculation on the new inserted documents and a small part of the original document collection. This relatively less calculation complexity results in fast clustering

speed when new documents are inserted into the document corpus occasionally and makes possible that the cluster result is up-to-date .

As we are considering dealing with large document corpus instead of clustering search results, we mainly concern offline clustering in this thesis project.

2.4.3 Hard and Soft Clustering

Depending on whether overlapping is allowed in the clustering result, clustering methods may generate hard clustering results or soft ones.

It is very common for one document has multiple topics, it might be tagged with multiple labels and be grouped into more than one clusters. In this scenario overlapping is allowed. For instance, for a document which describes how scientists discovered the way bats use to "hear" flies and catch them, how this biological technique was applied to create modern radar technique and how the radar benefited to martial engineering, it is quite reasonable to say that this document can be classified into "biology", "radar", "martial engineering" and some other relevant classes if there are any others. So, soft clustering includes this kind of clustering algorithms which may cluster documents into different clusters and each document may belong to several clusters and keep the boundaries of the clusters "soft". In summary with soft clustering each document is probabilistically assigned to clusters [6], just as shown in Figure2.2.

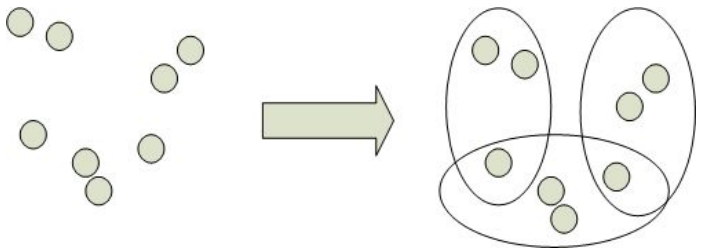

Figure 2.2: Soft Clustering

However there are some situations that demand one document should only be organized to the most relevant category. This kind of clustering is called hard

clustering because each document belongs to exactly one cluster. It is very important for the hard clustering algorithms to decide which cluster is the most matched one. Given the document above, a very reasonable way is to group it into the "radar" because it is mainly about the invention and the applications of radar. The idea of hard clustering can be illustrated in Figure2.3.

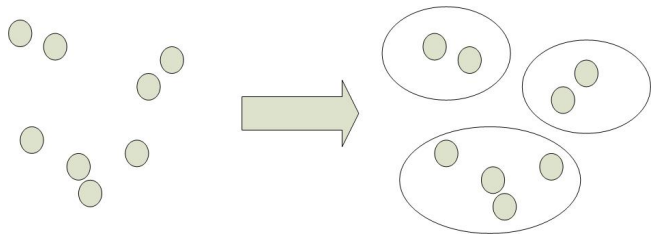

Figure 2.3: Hard Clustering

In this thesis project we would like to find a way to organize the document like a library in which one document can belong to one single category. So we mainly concern hard clustering.

2.4.4 Documents-based and Keyword-based Clustering

Keyword-based and document-based clustering are different in the features base on which the documents are grouped.

Document-based clustering algorithms are mainly applied on document vector space model in which every entry presents the term-weighting of term in the corresponding document. Thereby a document is mapped as a data point within a extremely high-dimensional space where each term is a axis. In this space the distance between points can be calculated and compared. Close data points can be merged and clustered into the same group; distant points are isolated into different groups. Thereby the corresponding documents are grouped or separated. As document-based clustering is based on the "document distance", it is every important to map the documents into the right space and apply appropriate distance calculation methods.

Keyword-based clustering algorithms only choose specific document features and based on these relatively limit number of features the clusters are gener-

ated. Those specific features are chosen because they are considered as the core features between the documents and they are shared by the similar documents and are sparse in unlike documents. Thereby how to pick up the most core feature is a very important step in keyword-based clustering.

2.5 Text Clustering Process

A common text clustering includes the steps in figure 2.4:

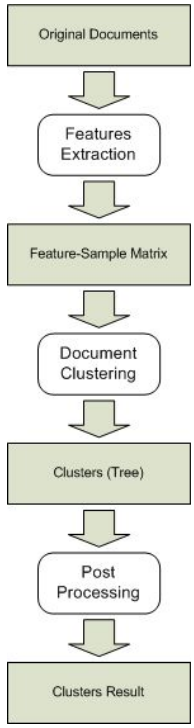

Figure 2.4: Text Clustering Steps

Feature Extraction The first step is feature extraction. It takes the original documents as input, processes these raw documents, analyzes them and picks out the relevant characteristics that might describe these documents. The output of feature extraction normally is a matrix in which each column stands for

a document and each row denotes a characteristic.

The quality of feature extraction has a great effect to further clustering algorithms. The feature extraction method should make similar documents are close to each other in the feature space and dissimilar documents are far away from each other. If useful features are drop out and irrelevant features are included, the distance between documents are messed up. No matter how good the clustering algorithms are, they cannot group the documents on the distorted distance.

Feature Extraction includes the following sub-steps:

- Stop-words removing
- Stemming
- Term Weighing
- Key Feature Extraction and Matrix Deduction

Document Clustering The second step is to apply clustering algorithms and get a clusters map. This clusters map not only can indicate which cluster a document belongs to but also may outline the relation between clusters. The input of this step is the feature-document matrix. With that matrix, a document is mapped as a data point in a high dimensionality space.

Normally clustering step is based on statistical or mathematical calculations without professional knowledge. Each feature loses its special meaning and it only represents a dimension in the space.

Post Processing For different applications there are different ways to do post processing. One common post processing is to select a suitable threshold to generate the final cluster result. After document clustering we get a basic cluster map in which the clusters are organized like a tree or in a flat way. Thereby some post processing algorithms may be applied to find out the correct clusters relation.

2.6 Clustering Algorithm Introduction

Agglomerative hierarchical clustering is one of the most straightforward clustering algorithm. It merges the most similar pair of clusters at each steps until only one cluster left. Divisive hierarchical clustering algorithms process in a contrary way: starting from one cluster including all documents and split a cluster into two at each step. Principal Direction Divisive Partitioning (PDDP) algorithm is a recently proposed technique [15] which is a non-iterative techniques based on the Singular Vector Decomposition (SVD) to split the clusters.

K-Means is one of most celebrated and widely used clustering algorithms [15] [13]. It is the best representative of flat clustering algorithms. Although K-Means is often consider not as "good" as agglomerative [14], Michael Steinbach, George Karypis and Vipin Kumar [13] illustrated that "a simple and efficient variant of K-Means, bisecting K-Means can produce document clusters that are better than those produced by regular K-Means and as good as those produced by agglomerative hierarchical clustering techniques" [13].

In [18] We Xu, Xin Liu and Yihong Gong proposed a novel document partitioning method based on Non-negative Matrix Factorization (NMF) of the feature-document matrix. In theory NMF can derive the latent semantic space from the original data matrix. Thereby the cluster membership of each document can be determined from this latent semantic space.

Frequent Itemset is an association rule mining issue in data mining. Many algorithms have been developed for this issue, and one of popular algorithms is Apriori [25]. In [26] Ling Zhuang and Honghua Da introduced a new approach to apply Frequent Itemset method to find out the optimal initial centroid of the document collection for K-Means. In some document corpus this novel approach resulted a better performance than regular K-Means. Benjamin C.M. Fung, Ke Wang and Martin Ester [24] used frequent itemsets to construct cluster and to organize these clusters into a topic hierarchy.

A suffix tree for a string S of n characters is a Patricia trie containing all n suffixes of S [44]. Suffix tree is widely applied on data compression, computational biology and string searching and so on. Oren Zamir and Oren Etzioni [45] presented a feasible solution to apply suffix tree into document clustering. Compared with clustering algorithms based vector space model, suffix tree treat a document as a string instead of a set of words [45]. It makes use of the words

sequences when clustering documents.

Chapter 3

Preprocessing

In this chapter, we introduce the techniques which are applied to documents before they are clustered.

3.1 Stop-words Removing

Stop-words are words that from non-linguistic view do not carry information [1]. Stop-words removing is to remove this non-information bearing words from the documents and reduce noise.

One major property of stop-words is that they are extremely common words. The explanation of the sentences still held after these stop-words are removed. Most existent search engines do not record stop-words in order to reduce the space and speed up the searches. To organize large corpus, removing the stop-words affords the similar advantages. Firstly it could save huge amount of space. Secondly it helps to deduce the noises and keep the core words, and it will make later processing more effective and efficient.

Stop-words are dependent on natural language. Different languages have their

own stop-words list. For example:

- English: a, an, the, this, that, I, you, she, he, again, almost, before, after
- Danish: en, et, det, jeg, du, hun, han, igen, senere

Basically there are three types of stops words: generic stop-words, mis-spelling stop-words and domain stop-words. Generic stop-words can be picked up when reading the documents; the latter two have to wait till all documents in the corpus have been read through and statistical calculations have been applied.

Generic Stop-words Generic stop-words are in general non-information-bearing words within a specific language and they can be removed without considering any domain knowledge. For English, the extremely common words could be "a", "an" and "the" and so on.

One way to pick up and remove these generic stop-words is to create a stop-list which lists all generic stop-words within a specific language. When reading through a document, once a term found in the document presents in the stop-list, then it can be removed instead of putting into further processing.

Mis-spelling Stop-words Mis-spelling stop-words are not real words but mis-spelling words. Inevitably people may by mistake input some words that are not in the dictionaries, like spelling "world" as "wrold". Of course within a context a human being may find out this is a spell error and still be able to get the correct meaning from it. But it would be difficult for a computer to ensure the correct spell, although some search engines may find out the mis-spelling words and give a list of possible corrections. One way to deal with these spelling errors is to take them as stop-words and remove them from the further processing.

A good criterion to identify a mis-spelling stop-word is the term frequencies within the whole document collection. In a large documents corpus with more than 10000 documents, those terms only occurring once or twice are quite possible mis-spelling stop-words.

Some terms occurring once or twice within the whole document set are not mis-spelling stop-words. Still these terms can be removed because these too infrequent words give little help to later clustering.

In this thesis project we take terms occurring less than three times as mis-spelling stop-words. Thereby we remove all these infrequent words.

Domain Stop-words Domain stop-words in general are not extremely common words but they turn into stop-words only under specific domain knowledge or contents. For example, in a document corpus containing documents from categories animal, automobile, geography, economy, politics and computer, the word "computer" is not a stop-word because it is not common in all other categories and it helps to differentiate the computer-relative documents from other documents such as animal-relative or geography-relative ones. But when considering a corpus within which all documents are discussing about different aspects of computers such as software, hardware and computer applications, words "computer" will be too common to be included in the latter processing.

In a large document corpus where all the words in the stop-list and all infrequent words are removed, words occurring in more than 80% of the documents are very possible domain stop-words and they should be removed. These highly frequent words are removed because they are too common in the corpus, including them will not provide help to distinguish individual docuemnt.

3.2 Stemming

Stemming is the process to transform a word into its stem.

In most languages there exist different syntactic forms [21] of a word describe the same concept. In English, nouns have singular and plural forms; verbs have present, past and past participle tenses. These different forms of the same word could be a problem for text data analysis [1] because they have different spellings but share the similar meaning. To English, The different spelling for verb *learn* could be *learns* (present tense), *learning* (presenting tense) and *learned* (past tense). And the singular and plural forms of noun *dog* are *dog* and *dogs*. Taking these different forms of a same word into account may cost too much space and

lose the connection between these words. And as a result it introduces noise and make the later processing more difficult. Stemming is necessary before the documents are clustered.

Though stemming process may be helpful to the clustering algorithms, it may also negative affect them if *over-stemming* occurs. Over-stemming means that words are unsuccessfully stemmed together because they are sufficient different in meaning and they should not be grouped together [33]. Over-stemming introduces noise into the processing and results in poor clustering performance.

A good stemmer should be able to convert different syntactic forms of a word into its normalized form, reduce the number of index terms, save memory and storage and may increase the performance of clustering algorithms to some extent; meanwhile it should try to avoid over-stemming.

Conflation approach is to match different variants of the same words. Currently available conflation methods can be classified as [31]:

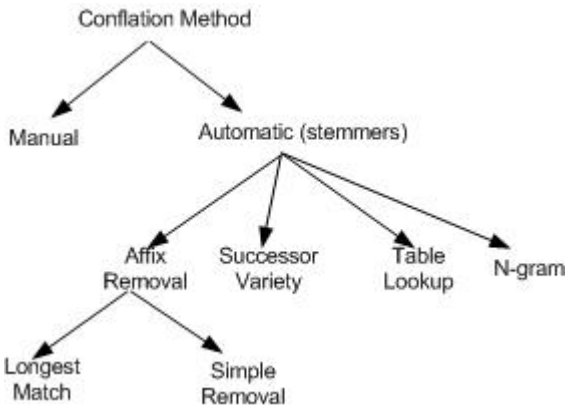

Figure 3.1: Classification of Conflation Methods

Table Lookup is a common way to stem the terms. Given a table containing all the stems and their possible syntactic varieties, it could be implemented efficiently to stem the words in documents. And the problem of table look up approach is that it is difficult to maintain such a table or a number of tables which contain all different varieties for the stems. Thereby table lookup method is impractical when applied on large document corpus.

Successor Variety is based on structural linguistic studies of words and morpheme boundaries [31]. It is applied to large collection of documents and get the statistical information of the successor varieties of prefixes. And then by applying the segmentation method to determine the stem, all the successor varieties of the stem are discovered and removed. The risk of successor variety method is that improper segmentation method is inclinable to over-stem the documents.

Affix Removal is to remove the suffix and prefix of words and keep the stems. It is based on linguistic concepts namely roots (stems) and affixes (suffixes and prefixes in English)[31]. Within affix removal there are mainly two types of methods [31]:

- **Porter Algorithm:** relies on complex rules such as condition/action rules, stem condition, suffix pattern and replacement to remove the affix.
- **Lovins Method**: uses a longer suffix list and apply *longest match* approach, iterative longest match, partial match and recording to keep the stem.

Porter Stemmer is a widely applied method to stem documents. It is compact, simple and relatively accurate. It does not require to create a suffix list before applyed. In this thesis project we apply Porter Stemmer in our preprocessing. Please refer to [32] for the detail algorithm descriptions.

3.3 Term Weighting

Term weighting is to formalize the following statements [7]:

- content-enduring words that occur several times in a document are probably more meaningful (and more important) than those occur just once.
- in a document collection infrequently used words are likely to be more interesting than common and frequent-appearing words

- each document should be treated as fair as possible.

there are three term factors corresponding to these three statements:

- Local term factor,
- Global term factor,
- Normalization factor

And then the term weighting schema may be written as:

$$TW = L(tf) * G(df) * N(tl)$$

where tf denotes the unique term frequency within a document; df is the number of the documents in which the term occurs in the whole document corpus and tl is the number of the unique terms within the document. Function L, G and N separately take care of local term factor, global term factor and normalization. Here we use $*$ to connect these factors together instead of simply multiplying them.

3.3.0.1 Local Term Factor

Local term factor is to weight a term based on its frequency within the document

A Document is represented by terms which present with orders. So terms within the document contains the basic descriptive information about the document [3], which is important in the later clustering processing. Basically within a document, the more frequent a content-enduring word occurs, the more important is it in that document.

Moreover, to increase or decrease the effect of the term frequency to the term weighting, we could apply mathematic algorithms on term frequency. The common used local term factor functions could be [1]:

$$L(tf) = \begin{cases} tf \\ log(1+tf) \\ 1 - \frac{1}{1+tf} \end{cases}$$

where tf is the number of a term occurring in a document.

Although term order is very important in representing documents, in this thesis project we do not take it into account. But in further researches order should be taken into account.

3.3.0.2 Global Term Factor

Global term factor is to evaluate the term/document frequency within the whole document corpus. A term may not only be weighted within a document but also in the document corpus. Under some conditions although a term occurs quite frequent in a document it might be not weight as much as another one which is less frequent.

One of the common global term factors is Inverse Document Frequency (IDF).

$$IDF = log(N/n_i)$$

where N is the total number of document in the corpus and n_i denotes the number of the documents in which the specific term occurs.

The basic assuming behind IDF is that the importance of a term is proportional with the number of document the term appears in [3]. The more documents the term appears in, the less important it is to differentiate the documents.

Another global term factor is Relevance Frequency. Though it is not considered in this thesis project it deserves further research.

3.3.0.3 Normalization Factor

Normalization Factor is to take care of the effect of document length.

Since long documents contain more words than short documents, they normally have a much larger term set than short ones. The large term set could make

a long document seem more relevant to a specific topic than the short one although the fact is opposite. So we assume that a term that occurs the same number of times in a long document and in a short one should be more valuable in the latter[3]. Normalization is to implement that assumption and make each document have the same significance.

In this thesis project within the normalization step we normalize the document vector to unit Euclidean length. Given a document vector $X = [x_1, x_2, ...x_n]^T$, each element is normalized as:

$$x_i \leftarrow \frac{xi}{\sqrt{\sum x_i^2}}$$

When new documents are added into the corpus, even if there is no new unique term inserted, the global term factor is affected, so is the normalization factor. Thereby if taking global term factor into account term weighting is not practical for online update.

In this thesis project we use normalized $TF \times IDF$ as the term weighting method.

Chapter 4

Key Feature Extraction and Matrix Dimensionality Deduction

Dimensionality reduction has been always one of the hot topics in text mining and information retrieval researches in that it helps to reduce the dimensionality and keeps or discovers core features.

When conducting text clustering, the high dimensionality of the feature-document matrix will lead to burdensome computation and large amount of memory. Besides introducing irrelevant and noisy features which may mislead the clustering algorithms, high dimensionality data may be too sparse for the clustering algorithms to find out useful structure in the data [36].

Different methods for reducing the dimensionality of the feature matrix thereby reducing the noise and the complexity of the document data and speedup further clustering algorithms have been investigated. The idea behind these methods is to extract the key/hidden features from the original feature-document matrix and to apply clustering algorithms only on these key features. One method is to keep only N most important terms from each document (as judged by the

chosen term weighting scheme),where N is much smaller than document size [39]. Another feasible way of dimensionality reduction is to project the data onto a lower-dimensional orthogonal subspace that capture as much of variation of data as possible [36].

We will focus on two popular contemporary dimension reduction techniques: Single Value D (SVD) and Random Projection (PR).

4.1 Singular Value Decomposition (SVD)

4.1.1 Description

SVD is more like a key feature extraction approach than a matrix dimensionality deduction method. It is a mathematical technique to create a new abstract vector space that is the best representation of the original document collection in the least-squares sense [39]. Comparing wit simple continuity frequencies or co-occurrence counts, SVD depends on a powerful mathematical analysis which may correctly infer much deeper relations between features and documents [41].

SVD can be used to decompose a rectangular matrix as a production of three other matrices - a matrix of left singular vectors, a diagonal matrix of singular values, and a matrix of right singular vectors. Given a $M \times N$ matrix A, applying SVD on it to make

$$A = USV^T \tag{4.1}$$

The left singular matrix U describes the original row entities as vectors of derived orthogonal factor values, the right one V describes the original column entities in the same way, and the diagonal matrix S contains the singular values[41] appearing in order of decreasing magnitude. One of the most important theorems of SVD states that a matrix formed from the first k singular triplets of the SVD (left vector, singular value, right vector combination) is the best approximation to the original matrix that uses k degrees of freedom. In another words, the leading singular triplets capture the strongest, most meaningful, regularities in data [41].

If the matrix A is a term-document data matrix in which each row stands for a unique term and each column stands for a document, it is quite obvious that the left matrix U provides the mapping from the original term space to a newly

generated space, and the left matrix V provides the mapping from the original document space to new space. To reduce the dimensionality of the original data matrix, we could delete the smallest singular values in S until only the k leading singular values left where k is the specific number of dimensionality we would like to reduce to:

$$A \approx A' = US'V^T \tag{4.2}$$

where S' only keeps the leading k singular values.

4.1.2 Complexity

Given a term-document matrix $X_{N\times M}$ where N is the number of terms and M is the number of documents, the computation complexity of SVD is $O(N \times M \times min(M, N))$ [38]. If to keep the first k leading singular triplets of S, the complexity can be $O(N \times M \times k)$.

4.1.3 Advantage

SVD is considered as a key feature extraction method instead of a simple dimensionality deduction. Depending on a powerful mathematical analysis, SVD is capable of discovering deep relations ("Latent Semantic" or "Latent Pattern") between words and documents [41]. These deep relations allow later clustering algorithms to closely approximate human judgments of meaning between words and documents and improve the performance of clustering algorithms. Meanwhile, SVD is independent from any linguistic information. The processing of SVD approach can be done automatically without input from language experts [3].

We can use SVD to extract the major associative patterns from the document space and ignore the small patterns [3]. As the consequence the dimensionality is highly deduced and clustering algorithms might need fewer computations.

4.1.4 Disadvantage and Improvement

When applied in text clustering, SVD has its limitations.

Firstly SVD is based on Vector Space Model which makes no use of words order. Although it may extract some correct reflection of word meanings without the aid of words order [41], SVD still may result incompleteness or error on some occasions.

Secondly in SVD, the combination of a document may include some negative entries which do not make sense in text domain [18]. In the SVD space the k leading singular vectors describe a k-dimensional subspace of the abstract LSA vector space. The right singular matrix project documents onto the k-dimensional subspace and can have positive and negative entries. These entries stand for the latent components. For a document the positive components tend to appear and negative components tend not to appear. Though it makes sense from a mathematical point of view but not in text domain.

Thirdly, the computation of applying SVD on large document corpus is extremely complex because SVD decomposes the original matrix into two orthogonal matrices and one singular value matrix [36] and orthogonal operation is time-expensive. One way to speedup the calculation is to only apply SVD on a sub set of documents to get the subspace and then project all documents onto the subspace.

Because $A = USV^T \Rightarrow U^T A = U^T USV^T = SV^T$

Given a data matrix A and A_s is a random subset of A.

$$A_s = U_s S_s V_s^T$$

Then we can get the new generated sub matrix

$$D \approx U_s^T A = SV^T$$

And then clustering algorithms can be applied on D which is a latent feature-document matrix.

If matrix A is not square, for instance the row number is much more than the column number, another way to speed up SVD is:

Because:

$$AA^T = SV^T VS^T U^T = USS^T U^T$$

applying SVD on AA^T we get:

$$SVD(AA^T) = Q\Lambda Q^T$$

There by:

$$U = Q$$

$$S = \Lambda^{\frac{1}{2}}$$

From $X^T = VS^T U^T$ we get:

$$V \approx X^T US^{-1}$$

where V is the latent feature-document matrix.

4.2 Random Projection

Random Projection (PR) is recently considered as a powerful and efficient method for dimensionality reduction. Basically as specified in [36], the key idea of RP arises from the Johnson-Lindenstrauss lemma [34]: if data points in a vector space are projected onto a randomly selected subspace of suitable high dimension, then the distance between data points are approximately preserved.

In random project, the original n-dimensional data is projected to a k-dimensional ($k << n$) subspace through the origin using a random $k \times n$ matrix R in which columns have unit lengths.

Given the origin data matrix as X$n \times m$, a random $k \times n$ matrix R, then

$$X_{k\times m}{}^{RP} = R_{k\times n} X_{n\times m}$$

is the projection of the origin data into the lower k-dimensional subspace.

To choose the appropriate random matrix R is the key point. Although it is suggested [36] that the elements r_{ij} of R are often Gaussian distributed, in [37] Achlioptas showed that the Gaussian distribution could be replaced by a distribution such as:

$$r_{ij} = \sqrt{3} \times \begin{cases} +1 & with\ probability \frac{1}{6} \\ 0 & with\ probability \frac{2}{3} \\ -1 & with\ probability \frac{1}{6} \end{cases} \tag{4.3}$$

Ella Bingham and Heikki Mannila in their paper [36] have shown that random matrix suggested by Achlioptas has practical significance. It preserves the distance between data points quite nicely and saves computations. The recommended value of k is 600 [36].

4.2.1 Complexity

Comparing with SVD, random projection is computationally very simple: as specified in [36] the complexity of generating the random matrix R and projecting the $n \times m$ data matrix X into k dimensions is $O(k \times n \times m)$. Considering that the text data matrix is very sparse with about c nonzero entries per column, the complexity is $O(c \times k \times m)$.

In our experiment random projection is much faster than SVD.

4.2.2 Advantages

The main advantages of applying random projection in dimensionality deduction processing is its computation is relatively simple but still preserves the distance between documents [36]. Thereby applying random project in text clustering might largely reduce the dimensionality and is still expecteds to get a good clustering performance.

4.2.3 Disadvantages

Although random projection might perform quite well to reduce the dimension, it is highly unstable [36]. As the high dimensions data is randomly projected to

a lower-dimension space, different random projects may project original data to different lower-dimension spaces. And these spaces differentiate from each other in the capability to preserve the distances, which is the key important factor in clustering algorithms. So different random projects may lead to different clustering results, and sometime the differences between results could be very dramatic.

4.3 SVD or RP?

Suggested in [36], the criteria of comparing different dimensionality deduction methods are the amount of distortion caused by the method and its computational complexity. In this thesis project we shall apply both SVD and Random Project in our dimensionality deduction process. In the experiment part we shall compare the performance of K-means on the data matrices generated by SVD and Random Project and their respective time consumptions.

4.4 Other Dimensionality Deduction Approaches

Frequent Itemset is a method for discovering interesting relationships in large databases. It might be to applied on document corpus to find out the frequent terms between documents. And using those frequent terms instead of the whole term space will greatly deduce the dimensionality.

Though in this thesis project we do not make research on the feasibility of applying frequent itemset to deduce the dimensionality of term-document matrix, we consider it a potentially promising solution.

Chapter 5

Clustering Algorithm

In this chapter we give detail descriptions to three selected clustering algorithms on their theory principle, time complexity, advantages and weaknesses.

5.1 K-Means

K-Means is probably the most celebrated and widely used clustering technique [12]. And it is the classical of the iterative centroid-based divisive clustering algorithm. It is different from hierarchical clustering in that it requires the number of clusters, K, be determined beforehand.

5.1.1 Algorithm Description

K-Means is an algorithm for partition (or cluster) N data points into K disjoint subsets S_j containing N_j data points so as to minimize the sum-of-squares criterion:

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |X_n - \mu_j|^2$$

where X_n is a vector representing the nth data point and μ_j is the geometric centroid of the data points in S_j [9].

The procedure of K-Means is:

1. Randomly make any partition and clustering the data points into K clusters.
2. Compute the centroid of each cluster based on all the data points within that cluster.
3. If a data point is not in the cluster with the closest centroid, switch that data point to that cluster.
4. Repeat step 2 and 3 until convergence is achieved. By then each cluster is stable and no switch of data point arises.

It can be imagined that if the number of clusters is bigger than the number of data points, each data point will be a centroid. If the number of the cluster is less than the number of data points, for each data point, the distances to all centroids will be computed and compared and the data point will be assigned to the cluster that has the minimum distance. Because the locations of the centroids are unsure, they have to be computed and adjusted based on the last update data. And then all the data are assigned to the new centroids. This algorithm repeats until no data point is switching to another cluster. The convergence will occur if [1]:

- Each switch in step 3 will decrease the sum distance J.
- There are only finitely numbers of data points into K clusters.

Mathematically these two conditions are satisfied in K-Means. So with K-Means algorithm the data will always converge.

[1] adapted from [10]

5.1.2 Time and Memory Complexity

The complexity to compute the distance between K centroids and N data points is $O(N \times K)$. This computation will repeat I iterations. In sum, to cluster N data point into K clusters with I iterations, the time complexity is $O(N \times K \times I)$

The memory requirement of K-Means is linear. To run the K-Means algorithm, the system only needs to store the data points with their features, the centroids of K clusters and the membership of data points to the clusters.

5.1.3 Disadvantage and Improvement

K-Means algorithm has its inherent weaknesses.

5.1.3.1 Initial K wanted

The first and probably "serious" weakness is that the number of clusters K must be determined beforehand. K-Means by itself cannot figure out the optimal number of clusters in the corpus. This would result in that the number of clusters is a pending issue. If the number of clusters is much fewer than the optimal one, irrelevant data points are clustered into the same cluster and this might confused the user. If the number of clusters is much more than the optimal one, really-related data point might be separated into different clusters.

To find the optimal K value, a validation algorithm is required to determin if the generated result: the clusters and their relevant documents, is the optimal one.

5.1.3.2 Flat Clustering Result

K-Means algorithm generates a flat cluster structure which cannot provide any hierarchy information. All clusters are at the same level. So K-Means share the same disadvantages that non-hierarchical algorithms may have when comparing

with hierarchical algorithm.

Although K-Means itself cannot generate hierarchical clusters, it could be done by making K-Mean generate two clusters every time, which in another words is to make K-Means bisecting. Bisecting K-Means separates documents into two clusters every time instead of K clusters. The basic steps to run Bisecting K-Means are [2]:

1. In the initialization, select a data point as the centroid the left part called c_L; compute the centroid of the whole data set i_M, and then compute the right centroid as $i_R = c_M - (i_L - c_M)$;
2. Separate the rest data point into two clusters. If a data point is more close to the left centroid point i_L, it is clustered to the left cluster. Otherwise it is put to the right one.
3. Compute the centroid of the left and right clusters, c_L and c_R
4. If $i_L = c_L$ and $i_R = c_R$, then stop. Otherwise let $i_L = c_L$ and $i_R = c_R$. Go to step 2.

At the beginning the whole documents into two clusters by the above steps. And then these steps are applied to the largest cluster, which is separated into two clusters as well. And then again the next largest one is selected and Bisecting K-Means is applied. This process repeated again and again until the specified numbers of clusters are found (if the number of clusters is given) or every cluster only contains one document.

5.1.3.3 Not Unique and Not Optimal

From different initial partitions K-means algorithm may result in different clustering results. And even when K-Means algorithm converges, it cannot promise to find the optima and the result does not has the minimum distortion. If the randomly the initial centroid points are the blue points as shown in figure 5.1, the clustering result is very possible not to be optimal.

To avoid the issue that convergence occurs but the distortion is not minimized,

[2] adapted from [12]

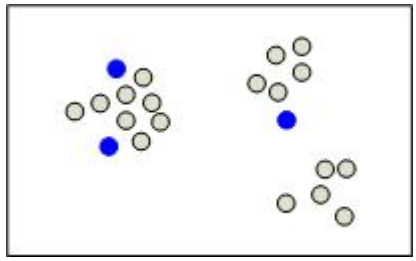

Figure 5.1: Non optimal initial centroids

one solution is to be very careful to start the initial partition [10].

The first step in K-Means is to make random partitions, which is to compute the initial centroids and then start the iterations. If from the beginning the centroids are put as far as possible from each other [10], K-Means may converge better and faster. For Instance, at first randomly pick up a data point as the first centroid. And then select another one from the rest data points as the next centroid which is as far as possible from the first centroid. Select the the third one as the third centroid which is the furthest from the first two. This selection continues until the Kth centroid is selected from the rest of data points, which is the furthest one from all the first $K-1$ centroids. With these selected K centroids, the common K-Means iteration starts. This careful selected initial partition may have a better chance than the common K-Means to reach the optimal convergence. And it could make the algorithm converge with more efficiency.

Another solution could be applying K-Means clustering several times. Every time the initial partitions are selected randomly. At last all these results are compared and computed. A summary solution is calculated out which has more chances to find the optima than a single run K-Means.

5.1.3.4 Not work well with non-global clusters

As mentinoed in [10], based on the distance measurement, the result of K-Means is circular cluster shape. When dealing with non-global clusters especially chain-like clusters 5.2, K-Means does not work well.

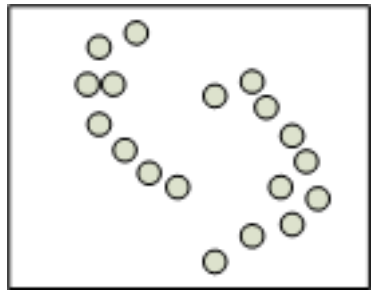

Figure 5.2: Chain-like clusters

One solution to this issue is K-Medoids algorithm. K-Medoids differentiates from K-Means in that: in K-medoids, each cluster is represented by one the data point in the cluster which is closest to the center of the cluster; while in K-Means each cluster is represented by the center of the cluster. Still K-Medoids is not good at solving the chain-like clusters.

Though K-Medoids has a theoretical better performance than K-Means, It is not recommended to be applied on large document corpus. In K-Medoids the complexity of each iteration is $O(k \times (n-k)^2)$ [11] where n is the number of documents and k is the number of clusters. Taking the iterations into account the total complexity of K-Medoids is $O(i \times k \times (n-k)^2)$. Thereby K-Medoids is not considered in this thesis project.

5.2 Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF) is a matrix factorization algorithm to find out the positive factorization of a given positive matrix [18]. It is a useful decomposition for multivariate data [16].

5.2.1 Description

NMF is designed to solve the following problems:
Given a non-negative matrix V, find out two non-negative matrix factors W and H such that [16]:

$$V \approx WH$$

Considering a set of multivariate data matrix $V_{n\times m}$ in which n is the dimension of one data sample and m is the number of samples. With NMF, this matrix is factorized into a $n \times k$ matrix W and a $r \times m$ matrix H and WH is very approximate to the original matrix V. If the value of k is chosen to be much smaller than n and m, W and H will be much smaller than V, which results in a compressed version of original data matrix [16].

$V \approx WH$ can be rewritten column by column and then it turns into $v \approx Wh$ where v and h are the corresponding columns of V and H. We can see that every data vector v is approximated by a linear combination of the columns of W, weighted by the components of h [16]. In another words, every sample is represented approximately by a block matrix W and the weight vector v. W provides the necessary data to reconstruct the v and h tells how to use the data in W.

Working in a similar way as SVD, NMF could find out the hidden pattern behind the term-document matrix V. Matrix W indicates how much the terms independently contribute to the clusters. Every element w_{ij} tells to which extent the term i belongs to cluster j. The larger the w_{ij} is, the more the ith term contributes to the $j$cluster. Similarly matrix H shows the relation between clusters (the rows) and documents (the columns). If h_{ji} is the maximum value in the ith columns, it indicates that the ith document is very likely to belong to the jth cluster.

When applying NMF to text clustering, the original matrix V is defined as:

$$V = [V_1, V_2, ...V_m]$$

$$V_i = [v_{i1}, v_{i2}, ...v_{im}]^T$$

where V_i denotes a document sample, and v_{ij} represents the term weighting of words t_j in the document V_i.

Our goal is to factorize non-negative term-document matrix V into non-negative $n \times k$ matrix W and $k \times m$ matrix H and minimize the following objective function:

$$E(W, H) = \|V - WH\|^2$$

where $\|*\|$ denotes the Euclidean distance between two matrix.

Daniel D. Lee and H.Sebastian Seung [16] prove that $\|V - WH\|^2$ is non-

increasing under the update rules:

$$H_{ir} \leftarrow H_{ir} \frac{(W^T V)_{ir}}{(W^T W H)_{ir}}$$

$$W_{rj} \leftarrow W_{rj} \frac{(V H^T)_{rj}}{(W H H^T)_{rj}}$$

And the objective function is invariant under these if and only if W and H are at a stationary point of the distance.

The steps to apply NMF in clustering are [18]:

1. Given a large document corpus, construct the term-document $n \times m$ matrix V in which n is the number of terms and m is the number of documents. v_{ij} denotes the ith weighted term-frequency in the jth document. The weighted term-frequency is based on TFIDF and is normalized on column.
2. Perform the NMF algorithm on matrix V and get the two matrices W and H
3. Based on the matrix H we could determine the clusters and the corresponding documents. For instance, the document j are assigned to the cluster c if h_{cj} is the maximum value for the document vector j.

5.2.2 Non-Negative Sparse Coding

Partik O. Hoyer [20] propose to combine sparse coding and NMF into *non-negative sparse coding* (NNSC). Similarly NNSC is to find out the hidden components. Moreover NNSC is trying to find a decomposition in which the hidden components are sparse. This makes each input document can be well represented by only a few significant non-zero hidden patterns[20].

NNSC is to factorize non-negative term-document matrix V into non-negative $n \times k$ matrix W and $k \times m$ matrix H and minimize the following objective function:

$$E(W, H) = \frac{1}{2} \|V - WH\|^2 + \lambda \sum_{ij} f(H_{ij})$$

The algorithm for NNSC is [3]:

[3] adapted from [20]

1. Initialize W^0 and H^0 to random positive matrices with appropriate dimensions.
2. set $W' = W - \mu(WH - V)H^T$
3. set any negative values in W' to zero
4. normalize W', here we get the new $W = W'$
5. set $W_{ij}^{new} = W_{ij} \times \frac{(W^T X)_{ij}}{(W^T WHT + \lambda)_{ij}}$
6. repeat step 2 to 5 until convergence.

where μ is the step size and λ is the parameter to control the tradeoff between sparseness and accurate reconstruction.

In our test we found out sometimes the algorithm might step so far that it increase the value of objective function instead of minimize it. Thereby we have to insert a validation step before step 6 to make sure the new W and H decrease the value of objective function. Otherwise rolling back arises.

5.2.3 Complexity

The complexity of NMF in updating and normalizing the W and H is $O(K \times M)$ where K is the number of clusters and M is the number of documents. Depending on how many iterations the factorization performs, the total complexity of NMF algorithm is $O(I \times K \times M)$ in which I is the number of iterations. So dose NNSC.

Compared with NMF, NNSC makes two result matrices sparse. Theoretically calculation on sparse matrix is simpler than on common matrix. Thereby NNSC might have the complexity advantage than NMF when applied in large document corups.

5.2.4 Advantage

Like many other factorization algorithms, NMF is try to find numerical linear representation of original matrix V. Non-negative.

In [18], Wei Xu, Xin Liu, Yihong Gong have illustrated that NMF surpasses SVD in the following:

- With NMF, the input document matrix is non-negative and the two output matrix are non-negative as well. This makes that each document is an additive combination of latent semantics. In SVD the matrix may contain negative entries, which should not exist in text domain. So NMF makes more sense than SVD when applying in text clustering.
- In the NMF space similar documents are spread along to the same axis, which represents a cluster. From the result of NMF we can determine the clusters and their relative documents by grouping the document to the axis (cluster) in which it has the largest project value. But with SVD, the document-cluster information could not be figured out directly from the result of SVD. This requires further clustering algorithms such as K-Means to apply on the SVD space to find out the final clustering result.

5.2.5 Disadvantage and Improvement

5.2.5.1 Initial K wanted

NMF by itself cannot determine the number of clusters. It requires the number of clusters k as one of input parameters. Neither can it organize the clusters in a hierarchical tree unless it is applied in a similar way as Bisecting K-Means algorithm.

5.2.5.2 Result Not Unique

NMF is try to find the most approximate factorization for the original matrix V so that:

$$V \approx WH$$

For a given data matrix NMF might find out different combinations of W and H which satisfy the objective function. For instance, give a data like:

NMF might find two different solutions like in 5.4.

When does NMF give unique solution? David Donoho and Victoria Stodden [17] illustrated that situations where the data does not obey strict positivity are

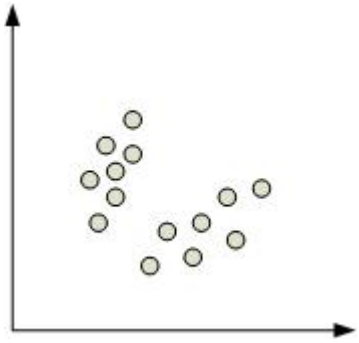

Figure 5.3: Data Space

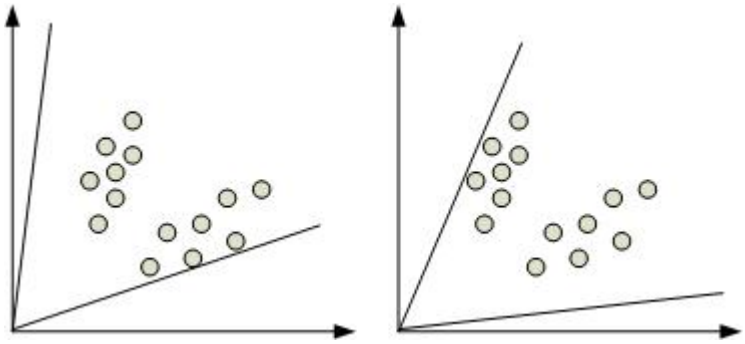

Figure 5.4: Solutions found by NMF

the boundaries of uniqueness. Therefore algorithms must look for the situations where the data does not obey strict positivity to make the solution unique:

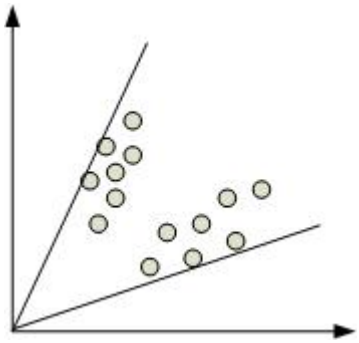

Figure 5.5: Solution 1 found by NNSC

Suggested by PatriK O. Hoyer [20], NNSC makes the factorization result sparse and hit the boundaries of uniqueness. In his experiment [20] NNSC surpassed NMF in finding out the hidden component (axis). In our pre-experiments we found out, NNSC is more time-consuming than NMF. Thereby in this thesis project we only apply NMF. But we still consider NNSC is an important variant of NMF and should be made on further research.

5.2.5.3 Low Converge Speed

Comparing with SVD, NMF does not require further clustering algorithm to group the document, but comparing with other clustering algorithms like K-means its convergence speed is relatively low, especially when dealing with large numbers of documents.

As discussed above, the complexity of NMF is $O(I \times K \times M)$ in which I is the number of interations, K is the number of clusters and M is the number of documents. With fixed K and M, the efficiency of NMF depends on how fast it could make the factorization converge. The less iteration to reach convergence, the more effective NMF is. One possible solution to speed up the convergence is to make each iteration step further. Thus Ruslan Salakhutdinov and Sam Roweis [19] put forward the Adaptive Overrelaxed NMF(ANMF) algorithm to improve the convergence speed. And they have specified that ANMF only cost one-fourth of iterations in NMF and got the similar clustering result. Further ANMF algorithm details may be found in[19].

5.3 Frequent Itemset

Frequent Itemset clustering algorithm is a keyword-based algorithm which picks up the core words with specific criteria and groups the documents based on these keywords.

5.3.1 Background

Frequent Itemset problem is a popular and well researched issue. The purpose is to discover interesting relationship in large databases. A well-known application

is applying frequent itemset method to analyze and find out useful patterns of customers' purchase behaviors from customers' buying transactions.

In [27] Frequent Itemset mining problem is defined as:

"A transactional database consists of sequence of transaction: $T = \langle t_1, \ldots, t_n \rangle$. A transaction is a set of items ($t_i \in I$). Transactions are often called baskets, referring to the primary application domain (i.e. market-basket analysis). A set of items is often called itemset by the data mining community. The (absolute) support or the occurrence of X (denoted by $supp(X)$) is the number of transactions that are supersets of X (i.e. that contain X). The realtive support is the absolute support divided by the number of transactions (i.e. n). An itemset is frequent if its support is greater or equal than a threshold value."

There are several ways to apply frequent time set on text clustering. In [24] a very interesting approach called *Frequent Itemset-based Hierarchical Clustering (FIHC)* is introduced and it is declared of efficiency and effectiveness. Thereby we apply FIHC in this thesis project to see how good it is when applying on large document corpus. In this section we give a introduction to the algorithm. Please refer to [24] in which Benjamin C.M. Fung, Ke Wang, Martin Ester gave detail description about applying FIHC on document clustering [4].

5.3.2 Algorithm Description

Applying FIHC to cluster documents and generate hierarchy tree includes three steps: [24]:

- Picking out all frequent itemsets that might satisfy the frequent itemset criteria.
- Constructing Clusters: this step is to generate the initial clusters and make them disjoint.
- Building Cluster Tree: this step constructs the hierarchical cluster tree and prunes it.

We define the following definitions:

[4] The following sub-sections are adopted from [24]

- A *global frequent itemset* is a set of items (terms) that appear together in more than a minimum fraction of the whole document collection
- *minimum global support* is the minimum required-percent of all documents for a itemset to be a global frequent itemset.
- A *global frequent item* refers to a term that belongs to some global frequent itemset.
- A *global frequent k-itemset* is a global frequent itemset containing k items.
- A global frequent item is *cluster frequent* in a cluster if the item is contained in some minimum fraction of documents in that cluster.
- *Cluster support* of an item in a cluster is the percentage of the documents in that cluster that contain the item.
- *Minimum cluster support* is the minimum required percentage of the documents in a clusters for a global frequent item to be cluster frequent in that cluster.

5.3.2.1 Picking out all frequent itemsets

Frequent Itemset problem has been widely investigated and many solutions have been put forward. Apriori algorithm is a widely used algorithm find out frequent itemsets. The idea behind Apriori is that all subsets of a frequent itemset are also frequent [21]. Thereby frequent k-itemsets can generated from frequent $k-1$-itemsets. Defining L_k is the set of frequent k-itemsets and C_k is the set of candidate k-itemsets, the basic Apriori algorithm can be described as [5]:

1. $k = 1$
2. find frequent itemset, L_k from C_k
3. form C_{k+1} from L_k
4. $k = k + 1$;
5. repeat 2-4 until C_k is empty

Algorithm detail can be found from [30].

[5] adapted from [28]

5.3.2.2 Constructing Clusters

Constructing clusters includes two sub steps: constructing initial clusters and disjointing clusters.

Constructing Initial Clusters For each global frequent itemset an initial cluster is constructed to include all the documents that contain this itemset. Thereby the global frequent itemset is the *cluster label* for the corresponding cluster and it identifies that cluster. Some documents may belong to multiple initial clusters because they may include more than one global frequent itemset. And some documents are unclustered documents for that they do not contain any global frequent itemset.

Disjointing Clusters For each multiple-cluster document, operations are applied to identify the best initial cluster for a document and keep it only in the best initial cluster. The idea to identify the best initial cluster for a document is that:
"A cluster is "good" for a document if there many global frequent items in the document that appear in "many" documents in that cluster"[24]

Starting from this idea, a score measurement is used to identify the best initial cluster for multiple-cluster documents:

$$Score(C_i \leftarrow d_j) = (\sum_{t} w(t) \times cluster-support(t)) - (\sum_{t'} w(t') \times global-support(t')) \tag{5.1}$$

where C_i is the ith cluster; d_j is the jth document; t denotes a global frequent item in d_j and also is cluster frequent within cluster C_i; t' represents a global frequent item in d_j but is not cluster frequent in cluster C_i; $w(t)$ and $w(t')$ are the global frequent item term weighting within document d_j. Here term weighting TF×IDF is applied.

By using the score measurement above, a multiple-cluster document is only kept in the cluster which gets the maximum score for that document. If there more than one clusters maximize the score measurement for a document, the docu-

ment is kept in the cluster that has the most number of items as its cluster label.

5.3.2.3 Building Cluster Tree

After the last step we get a set of disjoint clusters. In this step we use these disjoint clusters to build a cluster tree and the relationships between parents and children are created based on their similarity. After a cluster tree is constructed, we might merge the similar children under the same parents and merge the similar parent and children.

Constructing Tree The cluster tree is constructed based on three common rules:

- Each cluster has exactly one parent
- The topic of a parent cluster is more general than the topics of children
- The topic of a parent cluster and the topics of its children clusters are similar to some extent.

The cluster tree is created from the top (the root) to the bottom (the leaves). The root cluster is on level 0. All unclustered documents are put under the root at level 1 and put into one cluster. Depending on the number of items in their cluster labels, clusters are at the corresponding levels: all the clusters that use global frequent 1-itemset as cluster labels are put to level 1 under the root cluster; all the clusters using global frequent k-itemset as cluster labels belong to level k under the $k-1$ clusters.

When a k-itemset cluster C_j is put into the tree, it must be linked to a parent cluster at $k-1$ level. For a cluster, potential parents are those (k-1)-itemset clusters that their cluster labels are the subsets of the k-itemset cluster label. We can see there are at most k potential parents for a k-itemset cluster and at least one. Among all potential parents for one cluster, the "best" parents is chosen based on the score measurement. In 5.1 the d_j represents a conceptual document that is created by merging all documents in C_j. The cluster score is calculated between this conceptual document d_j and all its potential parent clusters. And the one who gets the highest score is chosen as the parent of C_j

Pruning Tree Pruning tree is to merge the very similar clusters and then to make the cluster hierarchy more meaningful.

In [24] an cluster similarity measurement is defined as:

$$Sim(C_i \leftarrow C_j) = \frac{Score(C_i \leftarrow d(C_j))}{\sum_t w(t) + \sum_{t'} w(t'))} + 1 \tag{5.2}$$

where C_i and C_j are two clusters; $d(C_j)$ means to combine all the document in C_j into one conceptual document; t is a global frequent item in $d(C_j)$ and is cluster frequent in C_i; t' is a global frequent item in $d(C_i)$ but is not cluster frequent in C_i; $w(t)$ is the term weighting of the global frequent item t in $d(C_j)$. This cluster measurement is to measure the similarity from C_j to C_i. To measure the inter-cluster similarity between C_j and C_i, [24] suggested to use:

$$Inter - Sim(C_i \leftrightarrow C_j) = (Sim(C_i \leftarrow C_j) \times Sim(C_i \leftarrow C_j))^2 \tag{5.3}$$

C_j and C_i represent two clusters including their descendant clusters.

As the global support and cluster support are always between 0 and 1. Thereby the maximum value of $Score(C_i \leftarrow d(C_j))$ is +1 and the minimum value is -1. After divided by the normalization factor $\sum_t w(t) + \sum_{t'} w(t'))$ and added with term +1, the value of [**?**] is within [0,2]. And then the range of $Inter - Sim(C_i \leftrightarrow C_j)$ is [0,2] as well. From the equations 5.2 and 5.3 that an $Inter - Sim$ value over 1 implies that the weight of similarity exceeds the weight of dissimilarity, and vice versa if the value below 1. So the similarity between two clusters is high if their $Inter - Sim$ value is over 1. And we should merge these very similar clusters to prune the tree.

Pruning Child is to shorten a tree by merging the child clusters and their parents. The pruning is applied from bottom to top and only to level 2 and below. For each cluster and it children the $Inter - Sim$ is calculated. The child cluster is pruned if it is similar to its parents cluster. If a cluster is pruned, all its child clusters turn into the children of their grand-parent.

Pruning Sibling is applied on level 1 of the tree. The $Inter - Sim$ for each pair is calculated. And most similar clusters (with the highest $Inter - Sim$ value) are merge into one and their children turn into the children of the merged cluster. The pruning continues until:

- the number of clusters on level 1 equals to the expected number of clusters, or

- all the very similar clusters are merged. There is no a pair of clusters with the $Inter - Sim$ value over 1.

By then a cluster tree is generated with the user-specific number of clusters.

5.3.3 Complexity

FIHC algorithm mainly includes three steps:

- finding out all global frequent itemsets

- initiating and pruning clusters

- building and pruning tree

The first step of finding out all global frequent itemsets is a widely studied mining issue. Rakesh Agrawal, Ramakrishnan SrikantHere [30] have shown a fast algorithm of mining all these frequent itemsets and got good performance when applying on large database.

Initiating clusters is relatively simple. Pruning clusters need to score relevant clusters for each document and assign it to the most relevant clusters. It costs more time than initiating clusters. In [24] discussed that time consumption of pruning clusters is no more than finding global frequent itemsets. After pruning Building and pruning tree conduct similar score and assignment work, as the number clusters are highly less than the number of documents, the time consumption is usually much less than finding global frequent itemsets.

In our experiment, we found out that when clustering Reuters corpus, the time consumption of initiating and pruning clusters can be more than picking out all frequent itemsets.

5.3.4 Advantage

5.3.4.1 Highly Reduced Dimensionality

Frequent Item Set approach differentiates from K-Means and Non-Negative Matrix factorization in that the latter two are documents-based algorithms but the former is a keyword-based clustering algorithm.

As mentioned in [25] the idea of Frequent Item Set approach is to cluster on the low-dimensional keyword sets instead of on the high-dimensional vector space. This dramatically reduce the large dimensionality of the document vector space. Although FIHC algorithm does not cluster on vector space, we believe some other keyword-based algorithm may benefit from this low dimensionality feature space.

5.3.4.2 Meaningful Cluster Label

Basically in Frequent Item Set documents are grouped together for they share the same keywords between each other. So each cluster may be tagged with the frequent term set as the labels. These labels can give the browser meaningful indication that what documents within the cluster are about.

5.3.4.3 Hierarchical tree structure without number of clusters as the input parameter

Unlike K-Means, NMF or many other clustering algorithms, in FIHC, the desired number of the clusters is not required as an input parameters. By applying FIHC the clusters are hierarchized and documents are organized in a well-organized way and user could have more details information about the whole structure of the corpus.

5.3.5 Disadvantage

The running time of FIHC is dramatically affected by the minimum global support. One of important step of applying FIHC is picking out all global frequent k-itemset that might satisfy the minimum global support. It is obvious that a low minimum global support may result in large number of global frequent itemsets and as the consequence the complexity increases dramatically.

The performance of clustering result is dramatically affected by the minimum global support and minimum cluster support. Some useful information is left out if these information is share only within a few documents and the terms shared by them do not satisfy the minimum global support.

Chapter 6

Validation and Evaluation

Basically there are two measures to evaluate how good a clustering algorithm is. One is Precision rate and the other is Recall rate.

Precision is to measure how much percent of the returned documents satisfy the query. Recall rate is defined to measure how much percent of relative documents is returned by the query.

For given D_l number of documents belong to class l, D_i of them are grouped correctly into cluster c which has D_c number of documents:

$$Recall = \frac{D_i}{D_l}$$

$$Precision = \frac{D_i}{D_c}$$

Recall and precision are two fundamental measures in text clustering. Most other evaluation measures are constructed based on them.

6.1 Validation Measure

Validation measure is to calculate the relative merits of clustering structures in a quantitive manner [23]. The merit here is referred to that if the documents are groups in the optimal number of clusters. Here we use this validation measure to find out the optimal number of clusters.

6.1.1 Index I Validation Measure

In [22] Ujjwal Maulik and Sanghamitra Bandyopadhyay put forward a simple but efficient way to validate the clustering result. Sameh A.Salem and Asoke K.Nandi applied this Index evaluation measure on their experiment [**?**]index2). Here we give an introduction on how to apply Index validation measure. For further details please refer to [22]. Index I is defined as:

$$I(K) = (\frac{1}{K} \times \frac{E_1}{E_K \times D_K})^P$$

in which

$$E_K = \sum_{k=1}^{K} \sum_{j=1}^{m} u_{kj} \left\| x_j - ck \right\|$$

$$E_1 = E_{k=1}$$

$$D_K = max_{i,j=1}^{K} \left\| c_i - cj \right\|$$

where K is the number of cluster, c_k is the center of the kth cluster and x_j is the feature vector of the jth document. And we use the power P to control the contrast between the different clustering configuration. We can see the when these three factor $\frac{1}{K}$, $\frac{E_1}{E_K}$ and D_K put the following constrains on I:

- $\frac{1}{K}$ is very straightforward to decrease I when K is increased.
- $\frac{E_1}{E_K}$ is to increase I when K is increased. For a given term-document matrix, E_1 is a constant. E_K will decrease when K is increased. Especially when K equals to the number of documents, E_K will be 0.
- D_K is to increase I when K is increased. But the maximum value of D_K is the distance between the two pair of furthest documents.

Ujjwal Maulik and Sanghamitra Bandyopadhyay [22] proved that when K is the optimal number of clusters, I will get the maximum values. Thereby we use this I index measure to validate that documents are grouped into the optimal number of clusters.

6.2 Evaluation Measures

To compare the "goodness" between different clustering techniques, we need measures to evaluate the clustering performance. One very obvious solution is to compare the output the automatic clustering results and the artificial labeled classes. When there is no artificial classes knowledge, there is another solution which is to compare the clustering results between different clustering algorithms. So basically there are two types of measures to compare different sets of clusters [13]:

1. **Internal Quality Measure**: to compare different clusters without reference to external knowledge. one important external knowledge is class labels which provide the information about the classes information about documents. Without the external knowledge, the internal quality measure normally measures the similarity between clusters. **Overall Similarity** is one of the internal quality measures.

2. **External Quality Measure**: to compare cluster results by comparing the groups produced by the clustering techniques to known classes [13]. Entropy and F-Measure are two major external quality measures.

Considering F-measure is a widely measure applied to evaluate the performance of clustering algorithms, we use F-measure as the major measure in this thesis project.

6.2.1 Entropy

Entropy is an external measure which provides the measure of "goodness" for un-nested clusters or for the clusters at one level of a hierarchical clustering [1].

[1] this section adapted from [13]

Given a cluster, its entropy can be calculated out by:

$$E_j = -\sum_i p_{ij} log(p_{ij})$$

where p_{ij} represent that the possibility of a document in the jth cluster belongs to the ith class.

And the total entropy for a clustering result is:

$$E = \sum_{j=1}^{m} \frac{n_j \times E_j}{n}$$

where m is the number of the clusters; n_j is the number of documents in the j cluster; n is the total number of documents.

6.2.2 F-Measure

F-Measure combines the precision and recall ideas from information retrieval[2]. According to Jason D. M. Rennie [47], F-Measure was first introduced by C. J. van Rijsbergen [46].

Given a matrix $C_{n\times m}$ in which n is the clusters and m is the class. Entry c_{ij} represents the number of class j in cluster i. c_i is the number of documents in cluster i and c_j is the number of documents in class j. We could get the recall and precision value as:

$$Precision(i,j) = \frac{c_{ij}}{c_i}$$

$$Recall(i,j) = \frac{c_{ij}}{c_j}$$

And then the F-measure value of cluster i and class j can be calculated by

$$F(i,j) = \frac{2 \times Precision(i,j) \times Recall(i,j)}{Precision(i,j) + Recall(i,j)}$$

By doing the above calculation we could get a F-Measure matrix $F_{n\times m}$ where n is the number of clusters and m is the number of labels. Every column in this

[2]this section adapted from [13]

matrix corresponds to a class. The maximum value in a column indicates the best mapped cluster for this class. So overall F-Measure value for matrix $C_{n\times m}$ is:

$$F = \frac{\sum_j (c_j \times F_{max}(i,j))}{N}$$

where max is taken overall clusters at all level, and N is the number of documents in the corpus.

We can see that F-Measure value will be always no larger than 1. If and only if all the same-class documents are grouped into the same cluster, the F-Measure value reaches its maximum value 1.

6.2.3 Overall Similarity

When there is no any external information available, we could use the overall similarity as a measure of the clustering performance.

In the feature space, documents are grouped based on the distance between them. Thereby a good clustering algorithm should group documents into clusters that are distant from each other as possible as they could. Overall similarity is to compute the distance of clusters and then to measure the performance of clustering algorithms.

Michael Steinbach, George Karypis, Vipin Kumar [13] suggest a method of computing the cluster cohesiveness to weight the similarity of the internal cluster similarity. In our implementation we use overall Euclid distance between cluster centroids as the overall similarity measure. It can be expected that the more a overall similarity is, the better is the clustering performance.

Chapter 7

Experiment Data and Result

In this chapter we describe the document corpus used for the performance evaluations, and compare the document clustering results on these document corpuses.

7.1 Experiment Data

In this thesis project we conduct experiment on Reuters and 20 Newsgroups corpus. These two document corpus have been among the ideal test document collections because these documents within them have been manually clustered based on their topics or their originations.

The 20 newsgroups corpus contains 19997 documents taken from the Usenet newsgroups collection. Each article is grouped into one or more semantic groups and the number of groups is 20. Most of these documents belong to only one group and about 4.5 percent of documents have multiple labels. All documents are partitioned evenly to 20 different topics. Some topics are very closely related to another while some are highly different from others:

Table 7.1: 20 newsgroup document topic

Document Topic	Related document number
comp.graphics	973
comp.os.ms-windows.misc	985
comp.sys.ibm.pc.hardware	982
comp.sys.mac.hardware	961
comp.windows.x	980
rec.autos	990
rec.motorcycles	994
rec.sport.baseball	994
rec.sport.hockey	999
sci.crypt	991
sci.electronics	981
sci.med	990
sci.space	987
misc.forsale	972
talk.politics.misc	775
talk.politics.guns	910
talk.politics.mideast	940
talk.religion.misc	628
alt.atheism	799
soc.religion.christian	997

The 20 newsgroups dataset can be downloaded from [43]. In our experiment we discard documents having multiple labels. This leads to a document corpus that contains 18828 documents. After removal of stops words and stemming, the number of unique terms is 42678.

The documents in Reuters are assigned one or more labels indicating which topics (classes) they respectively belong to. Compared with 20 newsgroups, the documents in Reuters are even more difficult to cluster because most documents are multi-topic relative and documents in each classes have a quite broad variety of content. The size of classes is dramatically different from one class to another class. Small size of classes may contain less than 5 documents, meanwhile some large size of classes contain even more than 1000 documents. In this thesis project experiment, we pick out the one-topic relative documents and exclude the classes which contain less than 3 documents.

7.2 Experiment Result

In this section we compare three different clustering algorithms based on two document collections by their performance and time consumptions. Meanwhile we compare two dimensionality deduction approaches and evaluate how much they affect (positive or negative) clustering results. Moreover we try to evaluate the effect of steps in preprocessing on clustering performance.

7.2.1 K-means, NMF Vs. FIHC

This experiment is compare the performances and time complexities of K-means, NMF and Frequent Itemset (FIHC). We apply these clustering algorithms on a Reuters corpus (9919 documents with unique terms) to generate results with different numbers of clusters (from 2 to 50) and the 20newsgroup corpus (18828 documents with 42678 unique terms) to generate 7 and 20 clusters.

The F-measure results of applying K-means, NMF and FIHC on Reuters documents are shown in figure 7.1. Generally K-means stands the best, although Frequent Itemset (FIHC) is often not much worse. NMF shows a good performance when the number of clusters is less than 10, but turns into worse than K-means and Frequent Itemset when the number of clusters is more than 20. When the numbers of clusters increases, the performances of these three clustering algorithms decrease to some extent. But they have a relatively high F-measure value when the number of clusters is between 2 and 10.

Time consumption of K-means is linear to the number of clusters. So is the time consumption of NMF, which increases even with a fast speed than K-means when the number of clusters increases. The time complexity of FIHC is almost the same and the change of clusters number does not really affect it.

In our experiments, the time complexities of K-means and FIHC are almost linear to the number of documents. Meanwhile the change of global minimum support for FIHC has a great effect on the running-time. NMF has a relative slow convergence speed; we could not succeed in applying it in a corpus which has more than 30000 documents in 12 hours.

The F-measure value for In 20 newsgroup corpus is shown in table 7.2. When

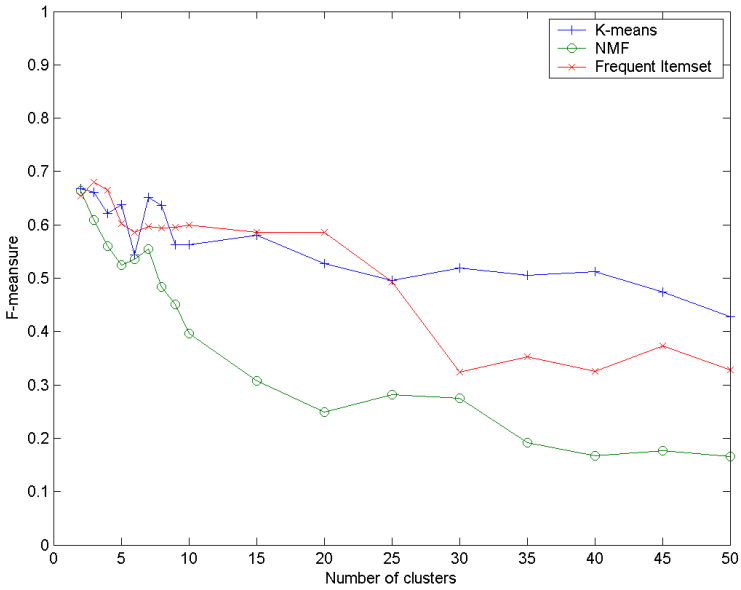

Figure 7.1: F Measure: K-means, NMF and Frequent Item Set(FIHC) in Reusters

applied on 20 newsgroups, NMF surpasses K-means and FIHC.

Table 7.2: F-measure: K-means, NMF and FIHC in 20 newsgroups

Number of clusters	K-means	NMF	FIHC
7	0.34018	**0.37096**	0.361634
20	0.44626	**0.51688**	0.364221

7.2.2 SVD Vs. RP

This experiment is to compare the performance and time complexity of SVD and Random Projection. We apply these two dimensionality deduction approaches on the data matrix of Reuters corpus; and then we apply K-means on two deduced matrices.

From figure 7.2 we can see that in Reuters corpus SVD achieves better performance than random project. It makes sense because SVD does stricter matrix projection than random project. From table 7.3 it is quite straightforward that in 20 newsgroups corpus the performance of K-means after SVD surpass the one after random project. Although the time consumption of SVD is much more than random project, when it comes to applying K-means part, the time consumption of K-means after SVD is much less than after random project.

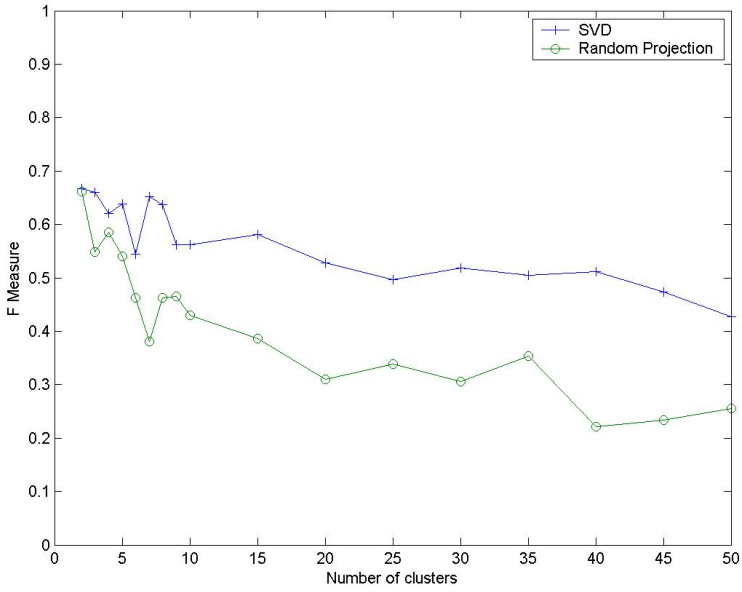

Figure 7.2: F-measure: K-means on SVD Vs. Random Projection in Reuters

Table 7.3: F-measure: K-means on SVD Vs. RP in 20 newsgroups

Number of clusters	SVD	RP
7	**0.34018**	0.3012
20	**0.44626**	0.39264

7.2.3 TFIDF Vs. normalize TFIDF

This experiment is to evaluate how much normalization affects the performance of K-means. We apply K-means on the Reuters corpus and the 20newsgroup

corpus.

The F-measure results of applying K-means on Reuters term-document matrix before and after normalized are shown in figure 7.3. The F-measure values on normalized matrix are more stable and better than those on not normalized one. When applied in 20 newsgroup corpus, applying K-means on normalized TFIDF matrix has a much better performance than applying K-means on not normalized TFIDF matrix 7.4. In these two experiments, the time consumption of both are almost the same.

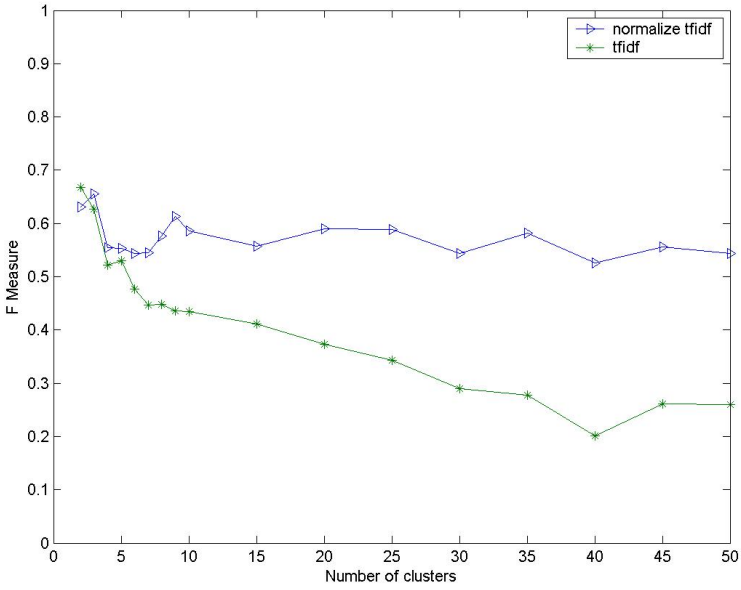

Figure 7.3: F-measure: TFIDF Vs. Normalized TFIDF in Reuters

Table 7.4: F-measure: K-means on normalized TFIDF Vs. TFIDF in 20 newsgroups

Number of clusters	Normalized TFIDF	TFIDF
7	**0.36614**	0.09615
20	**0.52181**	0.10075

7.2.4 Individual F-measure for each group in 20 newsgroups

We dig into the individual F-measure for clustering 20 newsgroups documents into 7 clusters (individual F-measure of NMF is shown in table 7.5, and the individual F-measure of K-means is shown in table 7.6. In these two tables the column stands for clusters and row denotes classes), and we find out:

- class 1, 8 and 12 are in the same cluster
- class 2, 13 and 16 are in the same cluster
- class 3 and 20 are in the same cluster
- class 5, 18 and 19 are in the same cluster
- class 6 and 7 are in the same cluster
- class 9, 10 and 11 are in the same cluster
- class 14, 15 and 17 in the same cluster
- class 4 is in one cluster by itself

NMF groups class 6, 7 and class 9, 10, 11 into the same cluster, but K-means puts class 6 and 7 in the same cluster with class 5, 18, and 19.

By comparing table 7.5 and table 7.6, we can find out the individual F-measure matrix for K-means is sparser than the one for NMF. This might be explained as K-means can group the documents from the same class into a few clusters and NMF spreads documents from the same class into all clusters. We can expect that NNSC(Non Negative Sparse Coding algorithm) might achieve the similar result as K-means.

From figure 7.4 we can see that the average of maximum individual F-measure for each class from NMF is more than the one from K-means. It can be explained that in this case NMF are better than K-means in discover the most relative clusters for documents from the same classes. Because the sizes of groups in 20 newsgroups are quite even, the higher average F-measure gives the high summary F-measure in the end. Thereby the summary F-measure of NMF is more than the one of K-means.

Table 7.5: Individual F-measure: NMF in 20 newsgroups to generate 7 clusters

a	1	2	3	4	5	6	7
1	0.0084	0.0675	0.0029	0.0048	**0.3907**	0.0364	0
2	0.0258	0.0208	**0.318**	0.0078	0.0036	0.0153	0.0203
3	0.0034	0.0031	0.002	**0.626**	0.0014	0.0016	0
4	**0.6863**	0.0315	0.0255	0.0006	0.0007	0.0089	0.0018
5	0.078	0.0633	0.101	0.0207	0.0044	**0.1598**	0.0793
6	0.006	**0.2618**	0.0192	0.0045	0.0109	0.042	0
7	0.0068	**0.2779**	0.0216	0.0071	0.0044	0.011	0.0026
8	0.0026	0.0264	0.0063	0.0039	**0.6012**	0.0283	0.0017
9	0.1706	**0.2017**	0.0028	0.0073	0.0112	0.0273	0
10	0.007	**0.2873**	0.0032	0.002	0.0089	0.0016	0
11	0.0414	**0.2231**	0.0041	0.0028	0.0118	0.0089	0.001
12	0.0374	0.0447	0.0051	0.0044	**0.2734**	0.0637	0.001
13	0.0077	0.0047	**0.3499**	0.0045	0.0015	0.0058	0.044
14	0.0146	0.0044	0.1532	0.0065	0.0022	0.0279	**0.4359**
15	0.0277	0.0139	0.133	0.0071	0.0015	0.0872	**0.3289**
16	0.0129	0.0079	**0.3601**	0.0026	0.0007	0.0063	0.0062
17	0.0198	0.0306	0.0827	0.0557	0.0029	0.1813	**0.184**
18	0.0068	0.0179	0.0098	0.0019	0.0007	**0.4668**	0.0053
19	0.006	0.1041	0.0071	0.0051	0.0013	**0.306**	0.0245
20	0.0034	0.0132	0.0086	**0.5865**	0.0007	0.0052	0.0026

Table 7.6: Individual F-measure: K-means in 20 newsgroup to generate 7 clusters

a	1	2	3	4	5	6	7
1	0	**0.2185**	0	0	0.0725	0.116	0.0015
2	0	0.0011	**0.1926**	0	0.1089	0	0
3	**0.7045**	0.001	0	0	0.0329	0	0
4	0	0	0.0151	0	0.066	0.0445	**0.6574**
5	0.0009	0	0.0514	0.0071	**0.1473**	0.0006	0.0054
6	0	0.0042	0.0018	0	**0.1623**	0.0042	0
7	0	0	0.0048	0	**0.1585**	0.0156	0
8	0	**0.5476**	0.0006	0	0.071	0.0276	0
9	0.0009	0	0	0	0.0439	**0.3977**	0.0014
10	0.0009	0.0054	0	0	0.0326	**0.4507**	0
11	0.001	0.0036	0	0	0.0518	**0.2974**	0.0031
12	0	**0.2401**	0.0007	0	0.0533	0.0883	0
13	0	0	**0.4056**	0.0256	0.0487	0	0.0027
14	0.0018	0	**0.2398**	0.3317	0.0579	0	0.0027
15	0	0	**0.2043**	0.1402	0.0877	0	0.0027
16	0	0	**0.2007**	0	0.1072	0	0.0054
17	0.0062	0.0011	0.0903	0.0975	**0.1242**	0	0.0013
18	0	0	0.0012	0.0014	**0.1611**	0.0108	0
19	0.0018	0.0031	0	0.0014	**0.1636**	0.0024	0
20	**0.4034**	0	0	0	0.0889	0.0012	0

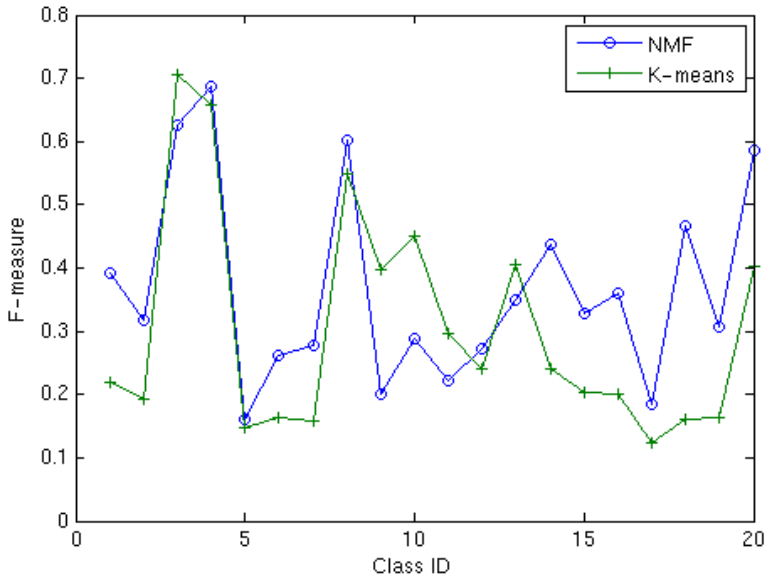

Figure 7.4: Individual F-measure: K-means Vs. NMF in 20 newsgroups

Chapter 8

Summary and Future Work

8.1 Main Findings

From our experiment, preprocessing does play an important role. In term weighting, normalization factor is to take care of the effect of document length and make each document have the same significance. In both Reuters and 20 newsgroups corps, there are improvement on the performance of clustering result if applying with normalization in the preprocessing. As the time complexity is rather simple, normalization is highly recommended in preprocessing.

Random projection is promising in that it may approximately preserve the original distance between data points (in our case data points are documents which are mapped to the high dimensionality matrix) with relatively simple calculation. But in our experiments the clustering performance and time consumption of applying random project to deduce the dimensionality is not satisfying. Firstly the clustering performance of random projection is worse than that of SVD. Secondly though applying random projection on dimensionality deduction is relative simple, applying K-Means on the result of random projection cost much more time than on the result of SVD. Although the summary time consumption of random project and K-Means is less than the summary time consumption of SVD and corresponding K-Means, we hesitate to suggest apply-

ing random projection to deduce the dimension in text clustering.

In [18] Non-negative Matrix Factorization was introduced as a very promising document clustering technique. In our experiment when the required number of clusters is relatively small, NMF has a good performance in the Reuters corpus and the time consumption is acceptable. When trying to clustering documents into a relatively more number of clusters (normally more than 10), the performance of NMF turns into relatively low. While in 20 newsgroups, NMF surpasses K-means and FIHC. Further research may be needed to investigate the reason behind it. Considering its relatively slow convergence speed, we hesitate to suggest applying original NMF as an effective text clustering algorithm in large corpus.

The Frequent Itemset-based Hierarchical Clustering (FIHC) suggested by Benjamin C.M. Fung, Ke Wangy and Martin Ester achieves good performance within acceptable time consumption in our thesis project experiments although the time consumption and performance of FIHC approach are affected by the global support value. In [24] Benjamin C.M. Fung, Ke Wangy and Martin Ester supposed that the complexity of assigning documents to clusters is not more than that of mining frequent itemsets. In our experiments, in 20 newsgroups document corpus their analysis holds but not in the Reuters corpus. Further research might be necessary to investigate what factor causes the difference. Meanwhile FIHC generates a hierarchical clustering result for the document corpus which offering more information about the relation between documents than flat clustering such as K-Means. Basically we consider FIHC a possible solution to cluster and organize large document corpus.

Despite its weaknesses, K-Mean is generally good within three selected clustering algorithms in our experiment. Even direct applying K-Means on normalized term-document matrix can achieve good result without the help of SVD, sometimes even better. Surprisingly in the 20 newsgroups document corpus applying K-Means on normalized term-document matrix get a better performance than on the latent feature-document matrix generated by SVD. We consider K-Means is an effective and efficient approach for document clustering. In some document corpus K-Means might even achieve a better performance with the help of SVD although including SVD into the clustering processing might result in more time consumption. Thereby we suggest K-Means a good candidate on text mining and organization of large document corpus.

8.2 Future Work

Suffix tree clustering algorithm may deserve further research. In [45] a novel text clustering algorithm is introduced, which uses suffix tree to connect documents by the words and do clustering on this tree. This suffix tree approach provides another ways to represent the document corpus beside Vector Space Model and it takes term order into account. Oren Zamir and Oren Etzioni [45] mentions the advantages of suffix tree approach includes fast clustering speed, overlapping allowing, taking term orders and browsable results.

Further research might be made on the feasibility of combining Frequent Itemset and K-Means. One disadvantages of the vector space model is the high dimensionality. A potential solution is to use some keywords to represent the documents instead of all terms within the documents corpus. Frequent Itemset is a ways to find out the frequent terms among documents, and these frequent terms might be a good candidate as keywords. Especially when documents are evenly spread into each class, using frequent terms to present documents might be promising.

Term order might be a possible factor to improve the clustering performance. In this thesis project N-gram is not included in vector space model and stemming. N-gram approach to some extent uses the term orders, which provides more content information than vector space model only on unique terms. Including n-gram into vector space model might help to get a better performance.

NMF still might be promising and it has been successfully applied in some fields such as sound mining and image mining. Further research should be made on how to speed up NMF for instance considering the help of ANMF [19] or taking sparseness into account.

Appendix A

Glossary

Affix is a morpheme that is attached to a base morpheme such as a root or to a stem, to form a word. Affixes may be derivational, like English -ness and pre-, or inflectional, like English plural -s and past tense -ed [wikipeida].

Agglomerative clustering starts from and leaves, and considers each document as an individual cluster at the beginning. It merges a pair of most similar clusters until only one single cluster is left.

Apriori is a wildly studied and used algorithm to find association rules/hyperedges in large database.

Bisecting K-means is the divisive version of K-means. Bisecting K-Means separates documents into two clusters every time instead of into K clusters.

Boolean model is the most simple of these retrieval methods and relies on the use of Boolean operators. Within the Boolean Model, a document is represented as a set of boolean values each of which repents whether a specific term presents in the document: normally a 1 means present and a 0 means not present.

Characteristic is a property of an object. It might help to distinguish the object from others.

Cluster centroid is the centroid of all documents within the cluster. Normally it is obtained by averaging the weights of all terms in documents within the cluster.

Cluster frequent A global frequent item is cluster frequent in a cluster if the item is contained in some minimum fraction of documents in that cluster.

Conflation is the process to combine syntactic variations of words. It is a approach to stem words.

Divisive clustering starts from the root, and consider the whole document set as a single cluster. At each step divide a cluster into two (or several) subclusters until each cluster contains exactly one document or until the required number of clusters is archived.

Entropy in information engineering refers to how much information is carried by signal. It is to describe how much randomness in a signal or random event. [21]

Euclidean distance in mathematics is the "ordinary" distance between the two points that one would measure with a ruler. [Wikipedia]

Feature is a term or other information which contributes to constructing the content of a document.

Flat clustering is a clustering process to group all document into clusters in the same level without any hierarchy.

Frequent itemset is a set of items which occur in a adequate number of transactions in a large database.

Global frequent item refers to a term that belongs to some global frequent itemset.

Global frequent itemset is a set of items (terms) that appear together in more than a minimum fraction of the whole document collection.

Minimum cluster support is the minimum required percentage of the documents in a clusters for a global frequent item to be cluster frequent in that cluster.

Minimum global support is the minimum required-percent of all documents for a itemset to be a global frequent itemset.

Hard clustering is a process results in that each document in the corpus is put into exactly one clusters. There is no overlapping between clusters.

Hierarchical clustering is a process to a nested sequence of partitions, with a single, all inclusive cluster at the top and singleton clusters of individual points at the bottom.

Information retrieval is an subfield of information science concerning representation, storage, access and retrieval of information. Specifically it is the art and science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand alone databases or hypertext networked databases such as the Internet or intranets, for text, sound, images or data [Wikipedia].

Keywords refers to a core term within a document. Keywords capture the main content/topic/theme of the document.

K-means is an algorithm to cluster objects based on attributes into k partitions.

Noise in this thesis project refers to irrelevant terms or information which might compromise the discovery of relations between documents.

Offline clustering generates clusters in database beforehand and only perform simple operations to display the cluster results when a query is received. When new documents are added, offline clustering might need large amount of running time otherwise the clustering result might be not up-to-date.

Online clustering applies clustering algorithm on-the-fly when a query is received. Normally its clustering result is up-to-date.

Overstemming means words are unsuccessfully stemmed together because they are sufficient different in meaning and they should not be grouped together.

PDDP stands for Principal Direction Division Partitioning, is a hierarchical divisive clustering algorithm.

Polysemy is a word or phrase with multiple, related meanings [Wikipedia].

Porter stemming is a widely used algorithm to implement stemming.

Precision in information retrieval refers to the number of relevant documents in the result compared to the total number of returned documents [21].

Prefix is a type of affix that precedes the morphemes to which it can attach. Prefixes are bound morphemes (they cannot occur as independent words) [Wikipedia].

Recall in information retrieval refers to the number of relevant documents in the return result compared to the total number of relevant documents in the corpus.

Soft clustering is the process to generate clusters between each overlapping is allowed. Within soft clustering documents might be put to more than one relative clusters.

Stemming is to determine a stem form of a given inflected words.

Stop-list is a list of stop-words.

Stop-words are words that from non-linguistic view do not carry information and thus are irrelevant for information retrieval.

Suffix in linguistics is an affix that follows the morphemes to which it can attach [Wikipedia].

Suffix tree for a string S of n characters is a Patricia trie containing all n suffixes of S. Hence it is a substring index.

SVD stands for Singular Value Decompostion, is a mathematical matrix decomposition techniques.

Synonymy refers to the existence of more than one name for one taxon [Wikipedia].

Term is a word unit or string of characters in a document.

Term indexing is to build indexes of term-document relations.

Term weighting is to determine the weights for terms within documents.

Text classification is to assign documents into predefined groups. It is a supervised text mining approach.

Text clustering is an automatic discovery of document clusters in a document corpus and group documents into relative clusters. It is an unsupervised text mining apporach.

Trie is a tree structure that stores strings in such a way that there is no node for every common prefix [21].

Vector space model use numeric term-document weights as weights modelling the relation between terms and documents [21].

Wikipedia is a multilingual Web-based free-content encyclopedia.

Appendix B

Experiment Result

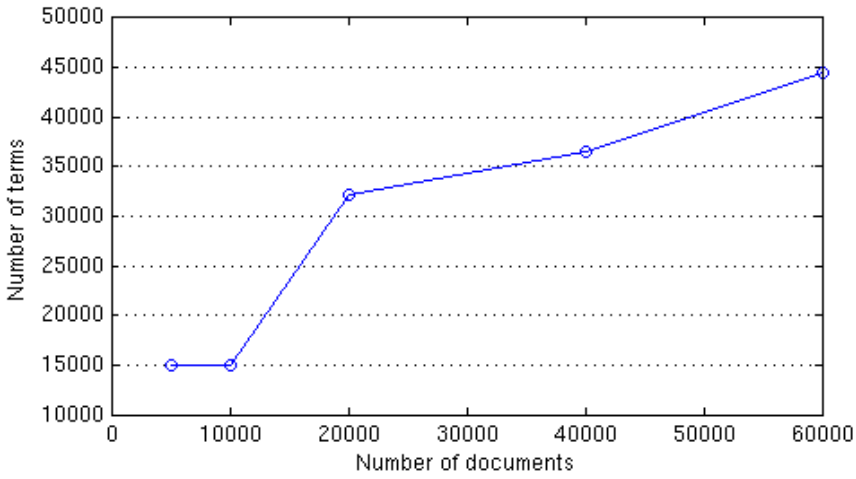

Figure B.1: Terms after stemming

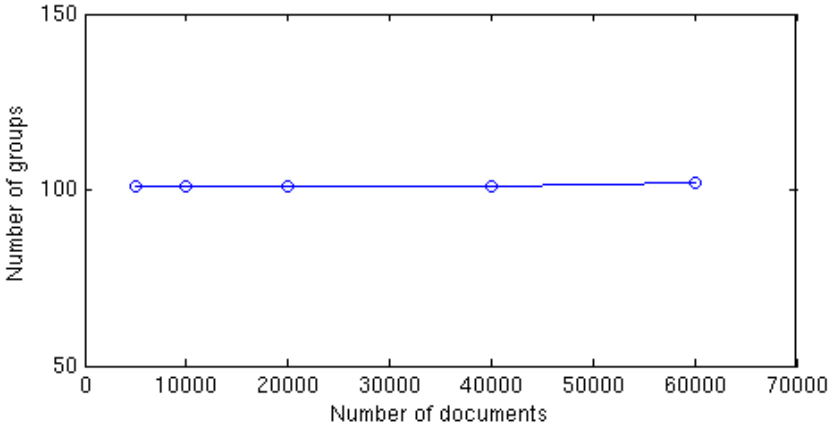

Figure B.2: Groups after stemming

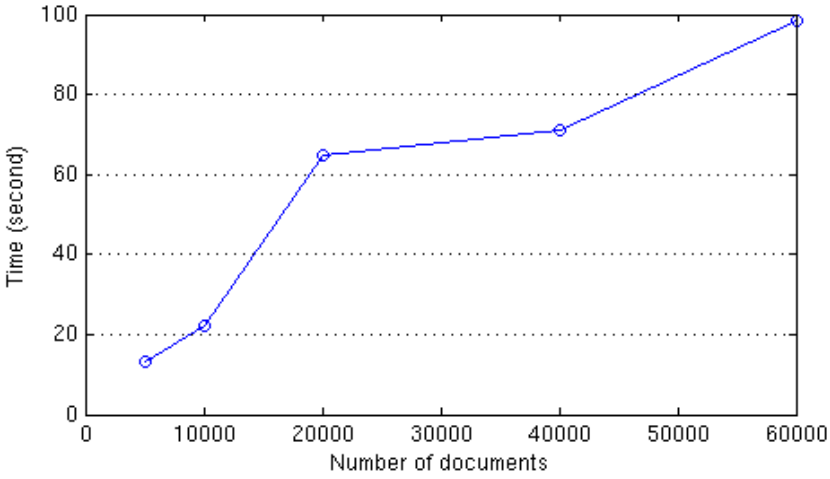

Figure B.3: Random Projection: Running time

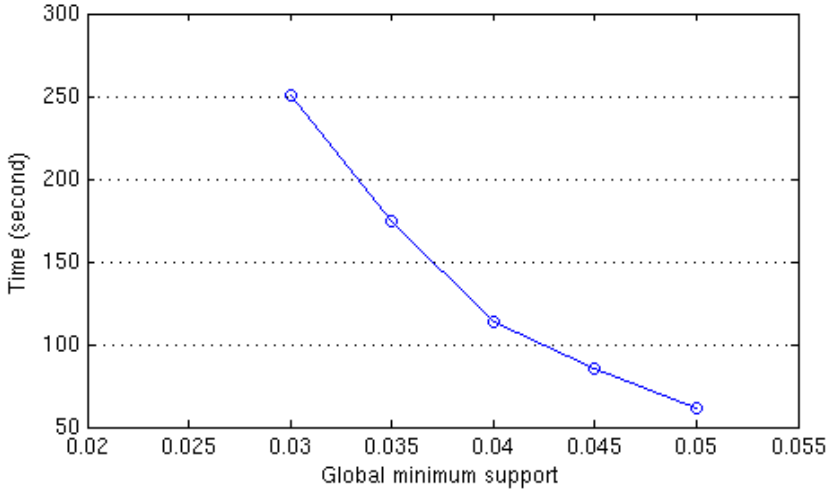

Figure B.4: FIHC global minimum support: Running time

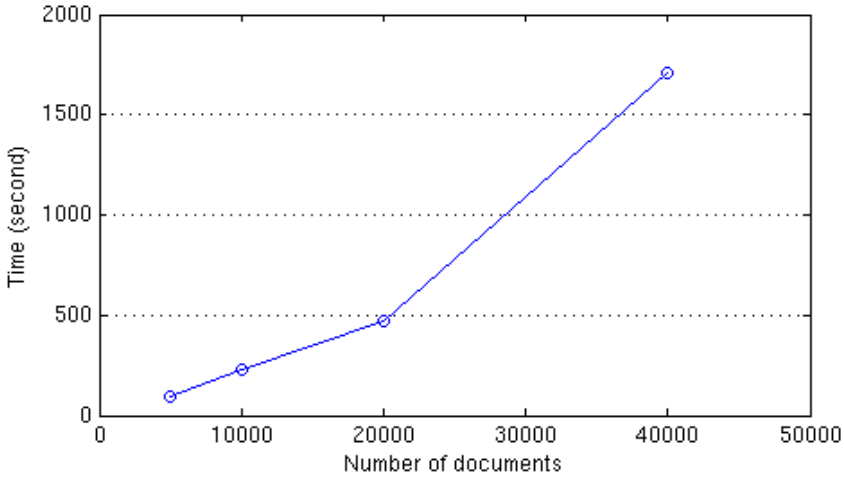

Figure B.5: K-means on TFIDF: Running time

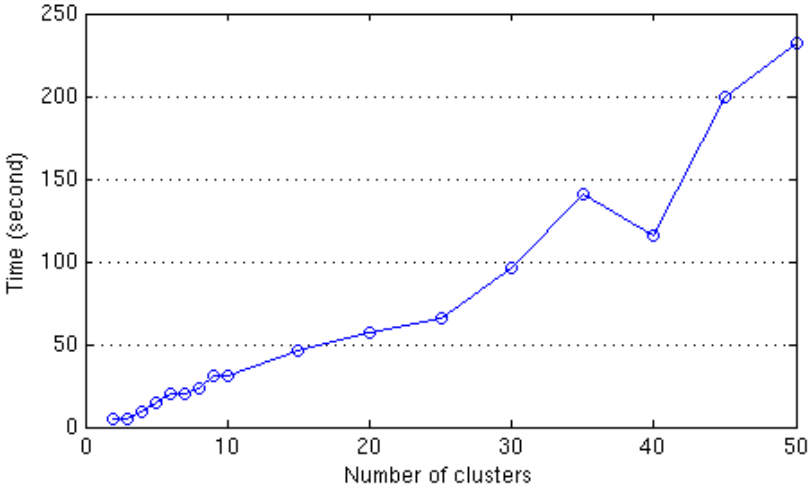

Figure B.6: K-means on normalized TFIDF: Running time

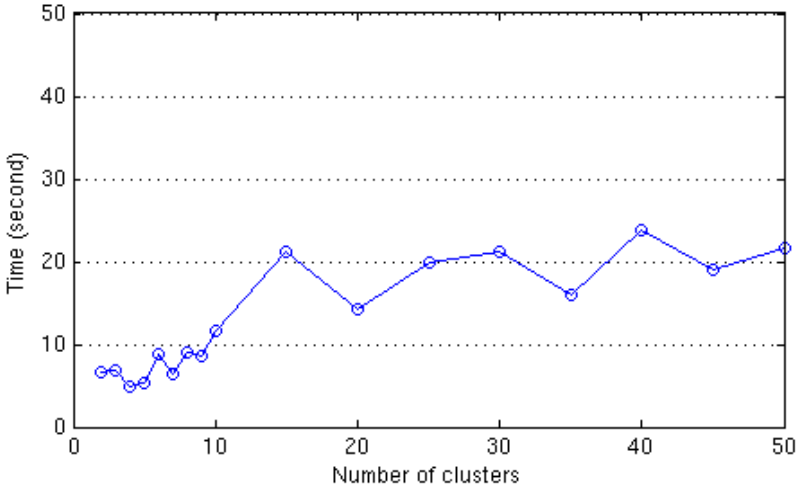

Figure B.7: K-means on SVD: Running time

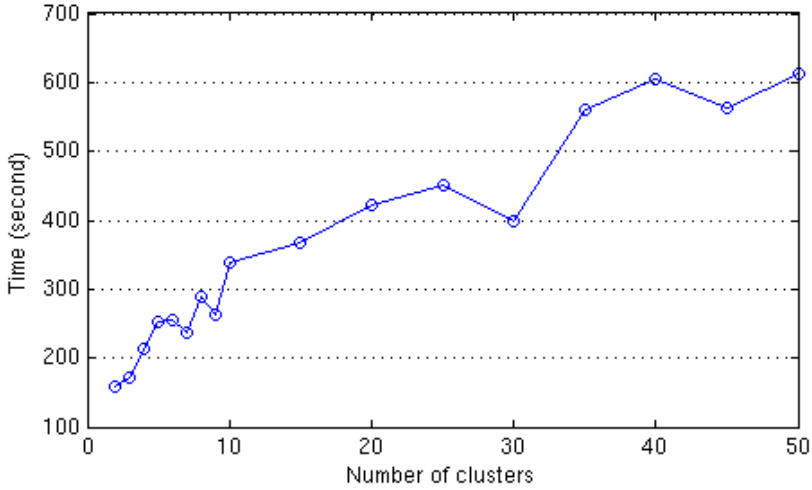

Figure B.8: K-means on random projection: Running time

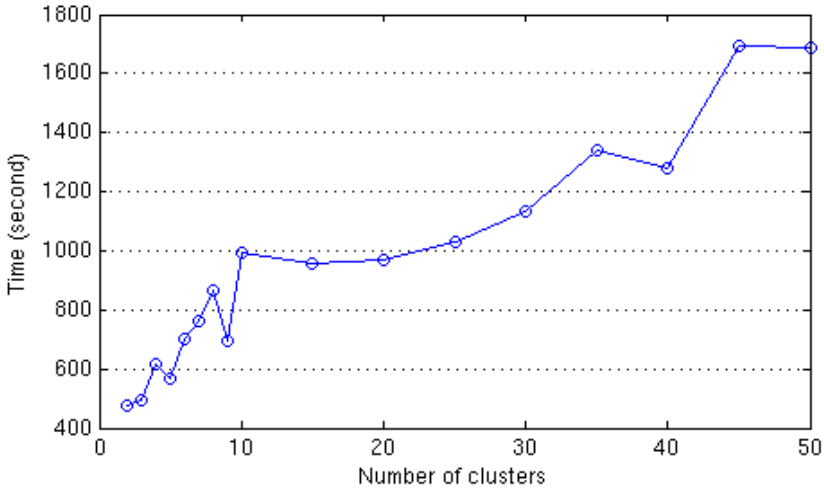

Figure B.9: NMF: Running time

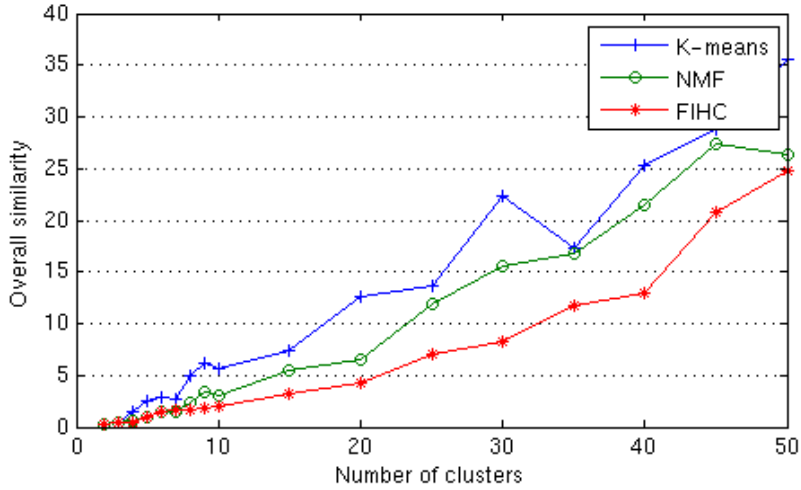

Figure B.10: K-means, NMF and FIHC: Overall similarity

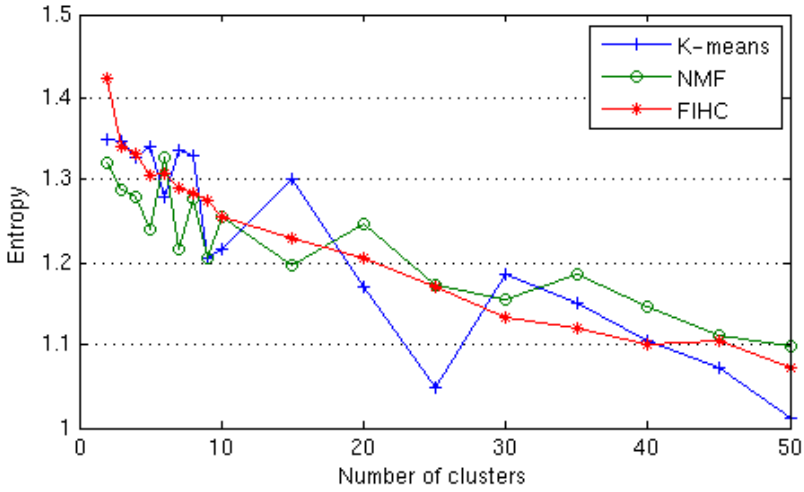

Figure B.11: K-means, NMF and FIHC: Entropy

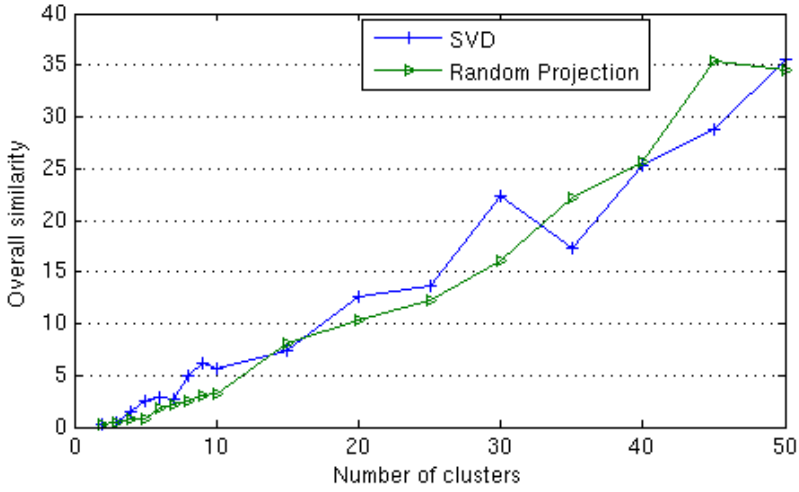

Figure B.12: Kmeans on SVD Vs. Random Projection: Overall similarity

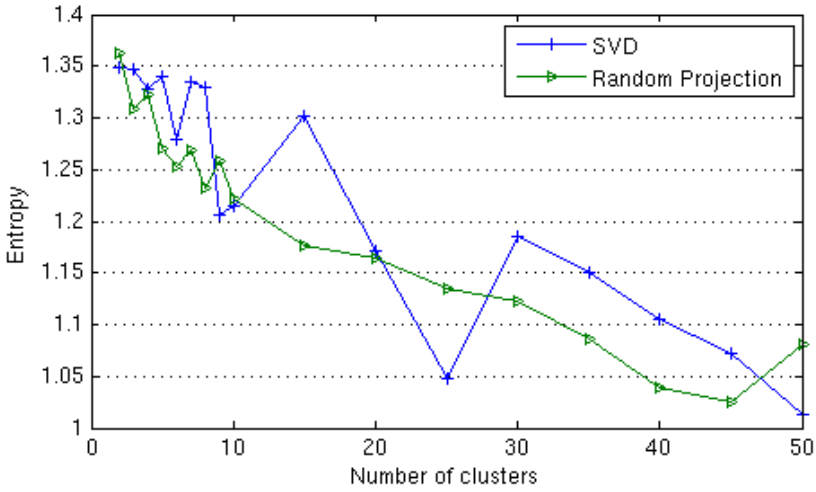

Figure B.13: Kmeans on SVD Vs. Random Projection: Entropy

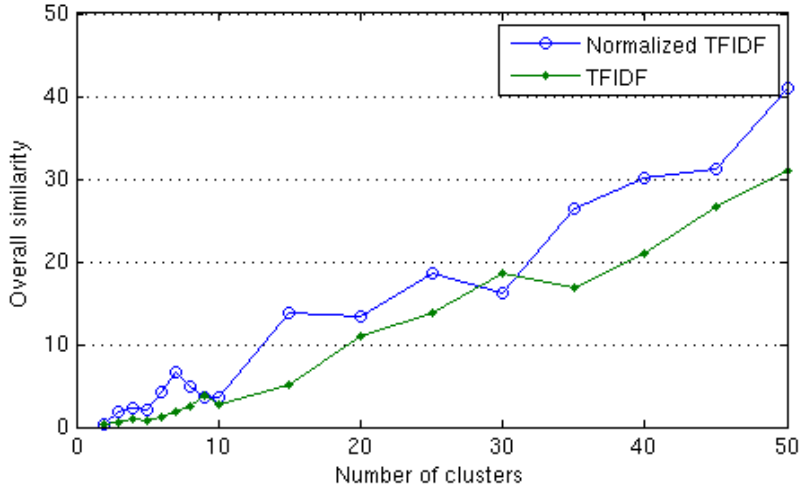

Figure B.14: K-means on Normalized TFIDF Vs. TFIDF: Overall similarity

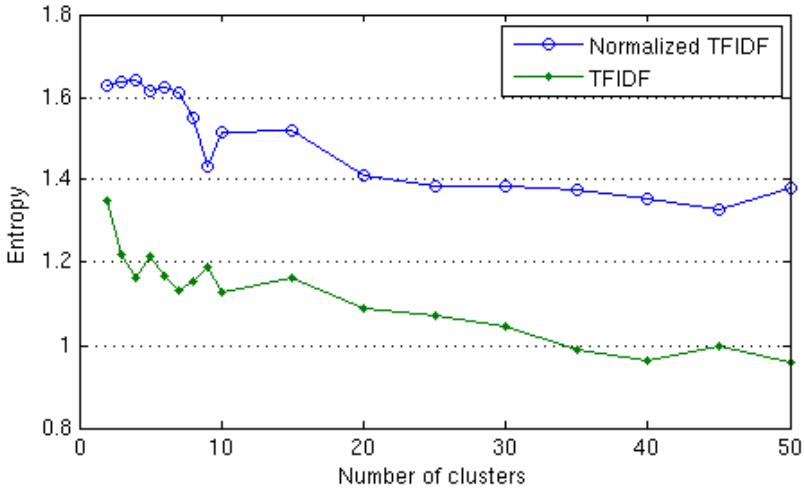

Figure B.15: K-means on Normalized TFIDF Vs. TFIDF: Entropy

Bibliography

[1] Marko Grobelnik, Dunja Mladenic and J. Stefan. Institute, Slovenia *Text-Mining Tutorial*

[2] Marti Hearst. *Untangling Text Data Mining.* ACL'99. 1999 http://www.sims.berkeley.edu/hearst/text-mining.html

[3] MOLE - Text Analysis Group. http://eivind.imm.dtu.dk/thor/projects/multimedia/textmining/

[4] Loretta Auvil, Duane Searsmith. *Using Text Mining for Spam Filtering.* Supercomputing. 2003.

[5] E. Garcia, *The Classic Vector Space Model.* Article 3 of Term Vector Theory and Keyword Weights. 2005. http://www.miislita.com/term-vector/term-vector-3.html

[6] Inderjit S. Dhillon, University of Texas, Austin *Information Theoretic Clustering, Co-clustering and Matrix Approximations.* MA Workshop on Data Analysis and Optimization. 2003.

[7] Clara Yu, John Cuadrado, Maciej Ceglowski, J. Scott Payne. *Patterns in Unstructured Data: Discovery, Aggregation, and Visualization*

[8] Andrew Moore, *K-means and Hierarchical Clustering - Tutorial Slides.* 2001. http://www-2.cs.cmu.edu/∼awm/tutorials/kmeans.html

[9] Eric W. Weisstein, *K-Means Clustering Algorithm.* From MathWorld. http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html

[10] Teknomo Kardi, *Weakness of K-Mean Clustering*. From K-Means Clustering Tutorials.
http://people.revoledu.com/kardi/tutorial/kMean/Weakness.htm

[11] Bing Liu. *Data Mining: Process and Techniques*. From CS583 - Data Mining and Text Mining
http://www.cs.uic.edu/∼liub/teach/cs583-spring-05/CS583-clustering.ppt

[12] Sergio M. Savaresi and Daniel L. Boley. *On the performance of bisecting K-means and PDDP*. In 1st SIAM Conference on DATA MINING, Chicago, IL, USA. paper n.5, pp.1-14. 2001.
http://www.siam.org/meetings/sdm01/pdf/sdm01_05.pdf

[13] Michael Steinbach, George Karypis, Vipin Kumar. *A Comparison of Document Clustering Techniques*. KDD. Workshop on Text Mining, 2000.
http://www-users.cs.umn.edu/∼karypis/publications/Papers/PDF/doccluster.pdf

[14] Richard C. Dubes and Anil K.Jain. *Algorithms for Clustering Data*, Prentice Hall, 1988.

[15] Sergio M. Savaresi, Daniel L. Boley, Sergio Bittanti and Giovanna Gazzaniga. *Cluster selection in divisive clustering algorithms*. H.3.3 . Information Search and Retrieval. 62-07 Data Analysis. 2002.

[16] Daniel D.Lee, H. Sebastian Seung. *Algorithm for Non-negative Matrix Factorization*. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, Advances in Neural Information Processing Systems 13, pp.556-562. MIT Press, 2001.

[17] David Donoho and Victoria Stodden. *When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts?* In Sebastian Thrun, Lawrence Saul, and Bernhard Scholkopf, editors, Advances in Neural Information Processing System 16. MIT Press, Cambridge, MA, 2004.

[18] Wei Xu, Xin Liu, Yihong Gong. *Document Clustering Based On Non-negative Matrix Factorization*. In ACM. SIGIR, Toronto, Canada, 2003.

[19] Ruslan Salakhutdinov, Sam Roweis. *Adaptive Overrelaxed Bound Optimization Methods*. In Proceedings of ICML, 2003

[20] Patrik O. Hoyer. *Non-Negative Sparse Coding*. In Neural Networks for Signal Processing XII, Martigny, Switzerland, 2002
http://www.cs.helsinki.fi/u/phoyer/papers/pdf/nnsc.pdf

[21] Kenneth Lolk Vester, Moses Claus Martiny. *Information retrieval In Document Spaces Using Clustering*. in Informatics and Mathematical Modelling, Technical University of Denmark, DTU. 2005

[22] Ujjwal Maulik, Sanghamitra Bandyopadhyay. *Performance Evaluation of Some Clustering Algorithms and Validity Indices*. In PAMI(24), No. 12, pp. 1650-1654. December 2002

[23] Sameh A.Salem, Asoke K.Nandi. *New Assessment Criteria for Clustering Algorithms*. in Statistical Classification 2. IEEE, pp 285-290. 2005.

[24] Benjamin C.M. Fung, Ke Wang, Martin Ester. *Hierarchical Document Clustering Using Frequent Itemsets*. In SIAM International Conference on Data Mining, 2003.
http://www.cs.sfu.ca/∼ester/papers/FWE03Camera.pdf

[25] Florian Beil, Martin Ester, Xiaowei Xu. *Frequent Term-Based Text Clustering*. In Proc. 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD)2002, Edmonton Alberta, Canada, pp.436-442. 2002.
http://ifsc.ualr.edu/xwxu/publications/KDD02.pdf

[26] Ling Zhuang, Honghua Dai. *A Maximal Frequent Itemset Approach For Web Document*. From Computer and Information Technology, 2004. CIT '04. The Fourth International Conference. pp.970-977. 2004.

[27] *An efficient implemenation of APRIORI algorithm*. in doxygen 1.3.9.1
http://www.cs.bme.hu/∼bodon/en/apriori/Documentation/html/

[28] Huan Liu, *Association Rules*. Chapter 6 in Introduction to Data Mining. 2003.
http://www.eas.asu.edu/∼mining03/chap5/chap5_list.html

[29] Michael Hahsler,Bettina Grun, Kurt Hornik. *arules - A Computational Environment For Mining Association Rules and Frequent Item Sets*. in Research Report Series / Department of Statistics and Mathematics, Nr. 15, April 2005.
http://www.jstatsoft.org/counter.php?id=140&url=v14/i15/v14i15.pdf&ct=1

[30] Rakesh Agrawal, Ramakrishnan Srikant. *Fast Algorithms for Mining Association Rules*. In Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94), pp. 487-499. San Francisco, CA: Morgan Kaufmann Publishers, 1994.

[31] Dr. Edward A. Fox, *CS5604 - Information Storage and Retrieval (F2001)*
http://ei.cs.vt.edu/∼cs5604/

[32] The Porter Stemming Algorithm. http://www.tartarus.org/∼martin/PorterStemmer/

[33] Chris O'Neill, Chris Paice. *The Lancaster Paice/Husk Stemming Algorithm*. 2001.
http://www.lancs.ac.uk/ug/oneillc1/stemmer/general/stemmingerrors.htm

[34] W.B. Johnson, J.Lindenstrauss. *Extensions of Lipshitz mapping into Hilbert space*. In Conference in modern analysis and probability, volume 26 of Contemporary Mathematics, pp. 189-206. 1982.

[35] P. Frankl, H.Maehara. *The Johnson-Lindenstrauss lemma and the sphericity of some graphs*. in J. of Combinatorial Theory B, 44. pp.355-362, 1988.

[36] Ella Bingham, Heikki Mannila. *Random projection in dimensionality reduction: Application to image and text data*. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001), San Francisco, CA, USA, pp. 245-250. August 26-29, 2001.

[37] Dimitris Acholioptas. *Database-friendly Random Projections*. In Proc. ACM Symp. on the Principles of Database. Systems, pp.274-281, 2000.

[38] Mehmet Koyuturk, Ananth Grama, and Naren Ramakrishnan. *Compression, Clustering and Pattern Discovery in Very High Dimensional Discrete-Attribute Datasets*. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 4, pp. 447-461, April, 2005.

[39] Kristina Lerman. *Document Clustering in Reduced Dimension Vector Space*.
http://www.isi.edu/∼lerman/papers/Lerman99.pdf

[40] Ian Soboroff. *IR Models: The Vector Space Model*. In Information Retrieval Lecture 7.
http://www.csee.umbc.edu/∼ian/irF02/lectures/07Models-VSM.pdf

[41] Thomas K Landauer, Peter W. Foltz, Darrell Laham. *An Introduction to Latent Semantic Analysis*. Discourse Processes , 25, pp. 259-284. 2002.
http://lsa.colorado.edu/papers/dp1.LSAintro.pdf

[42] Dongpo Bo. *Research of Clustering and its Applications in Text Mining*. From the Chinese Academy of Sciences. PhD thesis. 2000.

[43] 20 Newsgroups
http://people.csail.mit.edu/jrennie/20Newsgroups/

[44] Suffix Tree. From Wikipedia.
http://en.wikipedia.org/wiki/Suffix_tree

[45] Oren Zamir and Oren Etzioni, *Web Document Clustering: A Feasibility Demonstration*. In Proc. Acm Sigir'98. pp. 46–54. 1998.
http://www.cs.washington.edu/homes/etzioni/papers/sigir98.pdf

[46] C. J. van Rijsbergen. *Information Retireval*. Butterworths, London, 1979.

[47] Jason D. M. Rennie. *Derivation of the F-Measure*. 2004.
http://people.csail.mit.edu/jrennie/writing/fmeasure.pdf

Bibliography