INTERACTIVE TERM SUPERVISED TEXT DOCUMENT CLUSTERING

by

Seyednaser Nourashrafeddin

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
October 2014

This thesis is dedicated to my parents, who have raised me to be the person I am today.

Contents

List of Tables

List of Figures

Abstract

Text document clustering has broad applications in practice. For instance, a conference chair should place accepted papers into meaningful sessions. Students writing a thesis, or professors writing a proposal or planning a reading course need to organize their reference papers. Organizing documents into folders on a personal computer, or grouping emails into multiple inboxes are other instances of document clustering. Unsupervised document clustering algorithms require no user effort, but the obtained partitionings may be far from what the user intended to generate. User-supervised clustering algorithms involve the user in the clustering process and let her decide on the number and topics of document clusters. Generating useful clusters with minimum user effort is the main challenge in this mode. To address this challenge, we propose a user-supervised clustering algorithm, designed in three stages. First, we design a novel unsupervised clustering algorithm that can be easily extended into a user-supervised algorithm, thanks to its double clustering approach. We evaluate its performance against state-of-the-art clustering algorithms in unsupervised mode. We also extend this algorithm into an ensemble algorithm to incorporate Wikipedia concepts in document representation. We demonstrate that the integration can improve the quality of document clusters even though representing documents by Wikipedia concepts solely, may result in inferior clusterings to bag of words representation. Second, we propose three user-supervised versions for our clusterer based on term supervision (in the form of term labeling), document supervision, and dual supervision. We then demonstrate that with a comparable amount of simulated user effort, our proposed term labeling is more effective than a baseline term selection method. Third, we propose a graphical interface to support our term-supervised clusterer in interaction with human users. We then conduct a user study to evaluate the interface and its underlying clusterer. Analyzing the participants' opinions and comments reveals the usefulness of the proposed term-supervised clustering algorithm.

Chapter 1

Introduction

1.1 Motivation

Clustering is an unsupervised machine learning method which provides insight into a collection by generating groups of objects with similar characteristics. It has many applications in practice. For instance, clustering in microarray analysis helps researchers to extract correlated genes under certain conditions or disease [108]. Image retrieval process is boosted after clustering similar images in databases [64]. Moving-object data is clustered so as to find similar trajectories or sub-trajectories in satellites and tracking facilities [73]. It is widely used in analyzing marketing data in order to group consumers, target markets, or shopping items [1, 81]. It is also used in analyzing climatology data and weather forecasting [47].

Clustering is an important step preceding browsing in a text collection. The problem of clustering is widely studied in the data mining literature and numerous algorithms have been proposed [2]. Grouping similar documents brings forward precious information about the text topics in many applications. For instance, the output of search engines is clustered in order to help users in query refinement and knowledge extraction [38, 92]. Or, it is used in grouping similar sentences in mining customer opinions [40].

Traditionally, an unsupervised text clustering algorithm generates the same partitioning of a collection for every user. No user effort is required in this case and the user has no way to interact with the clustering algorithm if the partitionings obtained automatically are far from what she intended to generate.

Unsupervised clustering approaches thus cannot be used when the user likes to cluster documents according to her point of view [4, 54]. A user may even like to generate different partitionings of the same collection at different times. On the other hand, there is no ground truth for many collections in practice and it is the user who evaluates the quality of clusterings. These issues require the involvement of the users

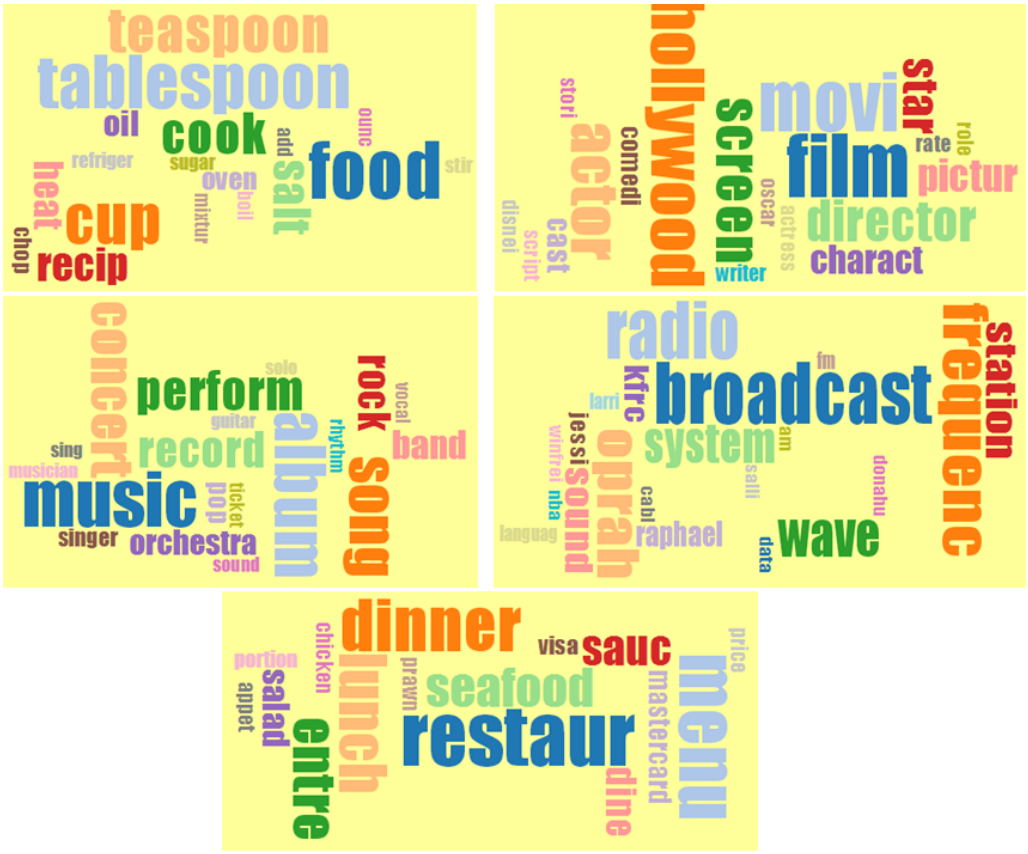

Figure 1.1: A partitioning of a document collection with five clusters

in the clustering process.

The following example will show why it is better to involve the users in the clustering process. Three different partitionings of a document collection are shown in Fig. 1.1 to Fig. 1.3. Based on the ground truth, there are five classes (topics) in this collection including: cook, restaurant, music, movie, and radio. A clustering algorithm generates the partitioning shown in Fig. 1.1 for user A. User A is satisfied with the output. However, user B likes to have three clusters like what are shown in Fig. 1.2; a cluster of food as a merge of the clusters cook and restaurant, a cluster of music-radio and the cluster of movie. There is also a user C who likes to have only two clusters as shown in Fig. 1.3; a cluster of food including the clusters cook and restaurant, and a cluster of entertainment including the clusters music, movie, radio. This scenario may also happen for the same user in different times. Hence, a user needs to specify the number and topics of document clusters in this example.

A possible solution to this problem is that a clustering algorithm is run with different

Figure 1.2: A partitioning of the same collection with three clusters

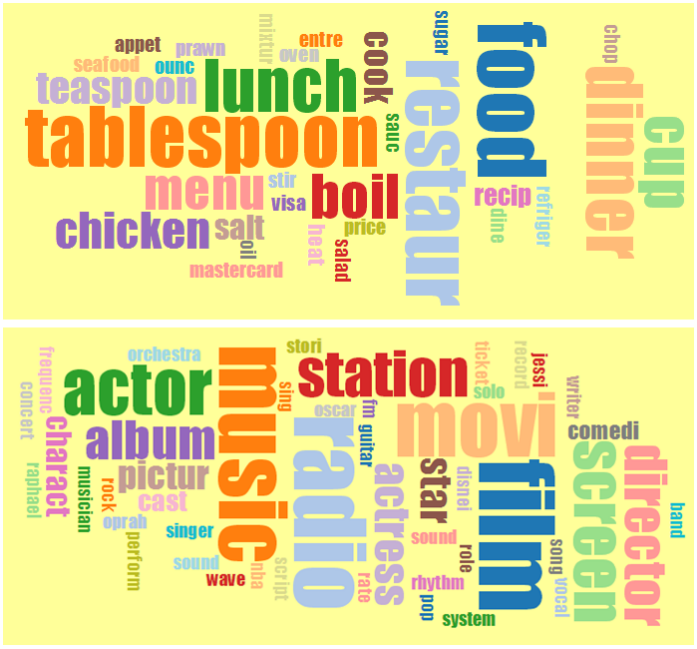

Figure 1.3: A partitioning of the same collection with two clusters

numbers of clusters. The number of clusters is usually a user-defined parameter in text clustering. The user is asked to specify the number of clusters a priori and the clustering algorithm generates the same number of clusters. Although the number of clusters is what the user wants to have, but she has no control over the topics of clusters and there is no guarantee that the algorithm generates the same clusters as she intends to generate. For instance, k-means always generates three clusters of cook, music, and movie if the user asks for three clusters in above-mentioned example. However, she might like to generate three clusters of movie, music, and radio or three clusters of cook, restaurant, and movie-music-radio.

The user should thus be able to specify not only the number of clusters but also the topics of clusters. To address this problem, user-supervised clustering algorithms have been proposed [54, 90]. These algorithms involve the user in the clustering process in an interactive mode so as to generate her desired partitionings. A user-desired partitioning has two characteristics:

- User-desired number of clusters: the user interactively tries different numbers of clusters to determine her desired number.
- User-desired topics of clusters: the user decides about the topics of the document clusters based on information provided about the clusters obtained interactively.

1.2 User Supervision

Document and term supervision are two kinds of interaction used in text clustering. Labeling documents, or specifying *"must-link"* or *"cannot-link"* pairwise constraints among documents has been proposed in the traditional semi-supervised algorithms as document supervision [9, 10, 15, 25, 30, 65, 106, 110] .

Traditional semi-supervised algorithms usually involve users' preferences as prior knowledge in the clustering process. After a supervision phase, user's knowledge is incorporated in the clustering process to generate clusters matching her preferences. This approach of collecting user preferences raises the following questions:

1. What is the best way to select a set of documents in order to elicit users' preferences? Is random selection a reliable approach for this purpose?

2. Most existing semi-supervised algorithms simulate user's supervision as a noise-free and consistent input [9, 10, 15, 25, 30, 65, 106, 110]. However, the users may make mistakes in supervising documents with similar topics. The accuracy of the input knowledge is dependent on the users' expertness and also on the selection method used to collect documents for supervision. How much do noisy user inputs affect the quality of clusterings?

3. Term labeling is used in the classification problem to incorporate user expectations [36, 95]. The user is asked to label important terms for classes. It has been demonstrated that term labeling is effective in document classification. Is term labeling a good approach in text document clustering?

The other way of involving users in text clustering is term supervision. Compared to document supervision, fewer studies have been done on term supervision. Term supervision (in the form of term selection) is used in [53, 54] to actively form a feature set for document clustering. The user is asked to specify discriminative terms during a phase of document supervision. A feature space is then formed including the terms specified as discriminative. The discriminative terms may also have higher weights in the feature space.

The term labeling approach proposed in this thesis lets the user not only specify discriminative terms but also organize them into preferred groups. Each preferred group of discriminative terms acts then as a cluster of keyterms specifying a topic in the document collection. Based on the topic keyterms, terms that exist in the collection are clustered. Our algorithm then extracts seed documents of each term cluster. The final document clusters will be generated based on the distance of documents to the seed documents.

1.3 Our Goals

Our hypothesis in this thesis is that term labeling is an effective way to involve users' knowledge in text clustering. We believe that term labeling is more effective than term selection and document labeling in user-supervised clustering. With a comparable amount of human effort, the user can provide more details about clusters by term labeling than specifying discriminative terms (term selection) or labeling

training documents (document labeling). Term labeling helps the underlying clustering algorithm to generate better clusters matching the user's preferences.

To evaluate our hypotheses, we first propose a novel unsupervised text clustering algorithm that can be easily adapted for interactive use. We compare the algorithm to state-of-the-art clustering algorithms in order to demonstrate its performance. We then propose three user-supervised versions for this algorithm based on term labeling, document labeling, and dual labeling.

We compare our term labeling and document labeling algorithms to determine whether term labeling is more effective than document labeling. To determine whether term labeling is better than term selection, we compare our term labeling approach to a baseline term selection approach.

In the baseline term selection approach, the user is asked to specify discriminative terms. A list of potential keyterms, extracted from the current document clusters, is presented to the user. The user then supervises the list by selecting discriminative terms. A feature set is then formed from the selected terms and document clustering is performed in the corresponding feature space in the next iteration. The user's feedback is only based on the presented single keyterms, and no information about the relevance of keyterms to the document clusters would be provided. She interacts for a few iterations until she chooses to terminate. She has the freedom to select different discriminative terms in each iteration.

In our proposed term labeling approach, the user is asked to perform term labeling. The top keyterms of each document cluster are presented to the user in the form of a term cloud. The user then modifies term clouds by relocating terms among them. The supervised term clouds are subsequently used to re-cluster documents. The advantage of this approach is that rather than presenting single terms to the user, it provides a potential association of terms with document clusters in form of term clouds. The term clouds help the user to identify an irrelevant keyterm and remove it from a cloud, or relocate it to a different cloud, which has relevant keyterms. She has the freedom to generate different term clouds in each iteration.

We compare the term selection approach to the term labeling approach with different parameter settings. Our experiments determine whether term labeling is more effective than term selection in terms of improving quality of clusters compared

to unsupervised mode. The amount of human effort is almost the same in both methods since the same number of terms are presented to the users in each experiment. The only difference is that the user should specify the label of terms in our approach, which we believe it is not very time consuming if related terms are presented in form of term clouds.

We anticipate that our interactive text clustering algorithm helps the user to generate clusters according to her point of view. Not only does presenting keyterms in the form of term clouds provide invaluable insights into the topics of clusters, it also enables users to decide on their desired number of document clusters through merging, splitting, or removing term clouds. Finding a desired number of document clusters is a key feature of the graphical interface implemented to support our proposed term labeling clustering algorithm. We also provide a document view panel in the interface. The panel helps the users in exploring the documents related to a focused term cloud in order to provide a deeper view about the topics.

Our graphical interface is based on term cloud representation. By selecting a document cluster, its corresponding term cloud is shown to the user. The term cloud includes the top keyterms of the document cluster. The user relocates terms among term clouds to specify the topics of document clusters. She also decides on the number of term clouds. This number specifies the number of document clusters.

1.4 Our Contributions

The main contributions of this thesis are summarized in this section. The contributions include the following:

1. We proposed an evolutionary algorithm to cluster text document collections [88, 89]. The main novelty of the algorithm is a multi-objective genetic algorithm which distills term clusters in order to remove non-discriminative terms. We also proposed a heuristic approach to find seed documents from the distilled term clusters. We conducted several experiments and the results demonstrated that this algorithm can outperform state-of-the-art double and co-clustering algorithms. The heuristic approach to find seed documents and the idea of distilling term clusters are used in the other clustering algorithms proposed in this thesis.

2. We proposed lexical double clustering algorithm (*LDC*) [90]. The novelty of the algorithm is in a greedy method which removes non-discriminative terms of term clusters by utilizing a feature selection method. The main advantage of this algorithm is that we can incorporate user interaction in the clustering process easily later. We have conducted several experiments to compare our clusterer, in unsupervised mode, with state-of-the-art text clustering algorithms like the *LDA* model [17]. Our experiments show that the proposed clusterer can generate comparable results to the *LDA* model.

3. We extended *LDC* into a new ensemble clustering algorithm in order to incorporate Wikipedia concepts in the document representation [91]. The novelty of the algorithm is in the consensus method. Two clusterings of a collection are first generated based on term-document and concept-document representations. The consensus method then combines the clusterings to generate an aggregated clustering. The documents with the same clusters in both clusterings are extracted for this purpose. These documents are then used to train a classifier. After the training, the classifier classifies the remaining documents. The experimental results show that this ensemble algorithm can successfully integrate Wikipedia concepts and significant improvements have been obtained even though the concepts alone resulted in poor clusterings.

4. We proposed three user-supervised versions of *LDC* based on term supervision (in the form of term labeling), document supervision, and dual supervision. We have conducted several experiments using simulated users to evaluate the user-supervised algorithms. The experimental results show that the quality of clusters is improved significantly compared to the unsupervised mode. We also compared our term labeling to a baseline term selection method. The experiments show that our term labeling is more effective than the term selection method with a comparable amount of simulated user effort.

5. We finally proposed a graphical interface to support the term-supervised *LDC* algorithm in interaction with human users. Our interface lets the user generate her preferred document clusters in terms of the number and topics of document clusters. Document clusters are represented as term clouds in the interface.

Creating a new term cloud, removing the existing ones, splitting two clouds or merging them are the available options in the interface. We also conducted a user study using a group of 30 participants. Analysis of the participants' opinions and comments provides an evaluation of the proposed interface and its underlying term-supervised clusterer.

1.5 Outline of the Thesis

The remainder of this thesis is organized as follows. We review related work in the area of text clustering in Chapter 2. A few feature selection methods proposed for text data are first introduced. We then review some important works performed in the field of double clustering, co-clustering and topic modeling. Enhancing text document clustering by using external resources like Wikipedia and WordNet is also covered in this chapter. We finally review some related interactive text clustering algorithms. We explain advantages and disadvantages of their graphical interfaces.

We explain the setup of our empirical experiments in Chapter 3. First, we review the text datasets used in our experiments and their characteristics. We explain how these datasets are pre-processed using Natural Language Processing methods. We then describe the evaluation measures used in this thesis.

Our evolutionary text clustering algorithm is described in Chapter 4. We first give a motivation about the double clustering and the contributions of our work in this area. We then describe our evolutionary algorithm in detail. Experiments performed in order to compare our algorithm to the well-known double clusterers and co-clusterers are reported. Finally, there are conclusions about the proposed evolutionary algorithm and future work.

Chapter 5 presents *LDC* in which the evolutionary module of the algorithm of Chapter 4 is replaced by a greedy feature selection method. Several experimental results are reported in this chapter. The quality of clusters obtained by the evolutionary algorithm and *LDC* is compared first. A comparison between *LDC* and the *LDA* model is then reported. We finally report a comparison to see whether the Google distance [61] is better than Cosine similarity in term clustering.

An ensemble approach for text document clustering based on Wikipedia concepts is proposed in Chapter 6. We propose a new consensus method to combine the

clusterings generated by using Wikipedia concepts and document terms. Experimental results reveal that we cannot ignore document terms from the clustering process.

We propose three user-supervised versions of *LDC* in Chapter 7. Experiments are conducted to evaluate the performance of the user-supervised algorithms. A comparison between our proposed term labeling and a baseline term selection algorithm is also reported in this chapter. Experimental results reveal that the term labeling is more effective than the term selection in our experiments.

We describe our graphical interface and the user study in Chapter 8. The detail of our user study and the obtained results are reported in this chapter. The analysis of the user study confirms our claims about the proposed term labeling approach.

A summary of the thesis with the final conclusions are presented in Chapter 9. We review the workflow of this thesis from proposing an evolutionary clusterer to the interactive term-supervised clustering interface. Proposed future work in the area of text clustering concludes this thesis.

Chapter 2

Related Work

Feature selection methods and clustering algorithms proposed for text data are reviewed in this chapter. Four unsupervised and two supervised feature selection methods are reviewed first. We review distance metrics used in text clustering algorithms. We then review some clustering methods well-known in the area of text clustering. The methods include a partitional clustering algorithm, some double and co-clustering algorithms, and two probabilistic topic modeling approaches. We then explain how users can interact with the document clustering process and related research is reviewed. Finally, we review some graphical user interface implemented for interactive text document clustering.

2.1 Feature Selection

In text clustering, a document collection is usually represented as a document-term matrix in the bag of words (BOW) model [2]. The rows of the matrix correspond to documents and the columns correspond to terms. Each entry of the matrix is a feature value, usually term frequency–inverse document frequency[1] ($TFIDF$), which indicates the importance of a term in the respective document. The dimensionality of this representation is very large and the underlying matrix is typically sparse. This is mainly because each document has a small fraction of terms that exist in a corpus. The respective document vector has thus numerous zero values corresponding to the terms that exist in the corpus but not in the document. It has been shown that the performance of text clustering declines as the dimensionality of this representation increases [105]. This is mainly because some terms are too general to discriminate topics. These non-discriminative terms act as noisy attributes in document clustering. Therefore, it is desirable to reduce the dimensionality of the BOW document representation by keeping only discriminative terms.

[1] `http://en.wikipedia.org/wiki/Tf-idf`

There are two common solutions for this problem: feature transformation and feature selection. Feature transformation techniques project the data objects from the original high dimensional space into a lower dimensional space. In this space, each feature is a linear or non-linear combination of the original features. Commonly used feature transformation techniques for text data are Latent Semantic Indexing [32], Independent Component Analysis [60], and Random Projection [16]. Feature transformation has the drawback that the newly generated features are not real words and it is not easy to interpret the obtained clusterings. Moreover, we cannot use the new features in our interactive text clustering since they are not real words. Hence, we have used only feature selection techniques in this thesis.

Feature selection techniques select a subset of features, which seem more important than the others according to some criteria. Since no projection (mapping) is performed on the features, it is easier to interpret the clusterings obtained by these techniques. In this section, we review four unsupervised and two supervised feature selection methods that are widely used for text data. Each method assigns a score to each term (feature). The terms with higher scores can be selected to form a new lower dimensional feature space.

2.1.1 Supervised Feature Selection

Supervised feature selection methods are often used for classification problems where the class labels of instances are known. However, they can be used in clustering as well. Treating the current document clusters as classes, these methods can be used to find the most important terms of clusters. We use these methods in our interactive clustering so as to extract topic (cluster) keyterms. There are two well-known supervised feature selection methods for text data [75].

Information Gain (*IG*)

Information Gain measures the amount of information we get about the label of a document if a term appears or does not appear in it [116]. Given the classes of the

instances, $\{c_s\}_{s=1}^{k}$, the information gain of term t is computed as [116]:

$$\begin{aligned} IG(t) = &-\sum_{s=1}^{k} P(c_s)\log P(c_s) \\ &+P(t)\sum_{s=1}^{k} P(c_s|t)\log P(c_s|t) \\ &+P(\bar{t})\sum_{s=1}^{k} P(c_s|\bar{t})\log P(c_s|\bar{t}) \end{aligned} \tag{2.1}$$

$$\begin{aligned} P(c_s) &= \frac{|d_s|}{N} \\ P(t) &= \frac{|d_t|}{N} \\ P(\bar{t}) &= 1 - P(t) \\ P(c_s|t) &= \frac{|d_{ts}|}{|d_s|} \\ P(c_s|\bar{t}) &= 1 - P(c_s|t) \end{aligned} \tag{2.2}$$

where $|d_s|$ is the number of documents in c_s, $|d_t|$ is the number of documents the term t appeared in, $|d_{ts}|$ is the number of documents in c_s where the term t appeared in, $\bar{t}$ means any term except t, and N is the number of documents in the collection. The goodness of t is directly proportional to the value of IG.

χ^2 Statistic

The χ^2 Statistic measures the dependency between a term t_i and a class c_s using the following formula [39]:

$$\chi^2(t_i, c_s) = P(t_i, c_s)P(\overline{t_i}, \overline{c_s}) - P(t_i, \overline{c_s})P(\overline{t_i}, c_s) \tag{2.3}$$

where $P(t_i, c_s)$ is the probability that the term t_i appears in a random document d that belongs to c_s. The probability values are estimated by counting. The goodness of t_i in a collection can be computed as the maximum of $\chi^2(t_i, c_s)$ values over all classes.

2.1.2 Unsupervised Feature Selection

Unsupervised feature selection methods are widely used for text clustering since no information about class labels is available a priori. These methods can be directly

applied on the document-term matrix of a text collection so as to remove non-discriminative terms and decrease the dimensionality of the presentation. Among all feature selection methods proposed in Machine Learning, the following four methods are widely used for text corpora [75].

Mean-TFIDF

Similar to the document vectors, each term is represented as a vector of documents in the *BOW* model. Each entry of this vector indicates the degree of importance of the term in the respective document. For each term t_j, the mean value of its *TFIDF* over all documents is measured using the following equation [105]:

$$\text{Mean-TFIDF}(t_j) = \frac{1}{N}\sum_{i=1}^{N} TFIDF_{ij} \tag{2.4}$$

where $TFIDF_{ij}$ is the feature value of term t_j in document d_i and N is the number of documents in the corpus. The higher the mean-*TFIDF* (MT) score, the better suited the term to be included in the new feature space. The computational complexity of this technique is $O(N)$ for each term.

Var-TFIDF

Instead of using the mean value, the variance of *TFIDF* over all documents is computed in this method [67]. Using Eq. (2.4), the var-*TFIDF* (VT) score is computed using the following formula:

$$\text{Var-TFIDF}(t_j) = \frac{1}{N-1}\sum_{i=1}^{N} (TFIDF_{ij} - \text{Mean-TFIDF}(t_j))^2 \tag{2.5}$$

Discriminating terms have higher var-*TFIDF* scores. The computational complexity of this technique is $O(N)$ for each term.

Entropy Rank

In this method, the quality of each term is measured by the entropy reduction when the term is removed [29]. Terms are removed in turn and then the entropy is measured.

If removing a term results in the maximum entropy, the term is the most important one. The entropy of term t is measured using the following formula:

$$\text{Entropy}(t) = -\sum_{i=1}^{N}\sum_{j=1}^{N}[S_{ij}\cdot\log(S_{ij}) + (1-S_{ij})\cdot\log(1-S_{ij})] \tag{2.6}$$

where S_{ij} is the similarity between documents d_i and d_j when term t is removed and is computed by the following formula:

$$S_{ij} = e^{-\alpha\cdot dist_{ij}}, \alpha = -\frac{\ln(0.5)}{\overline{dist}} \tag{2.7}$$

where $dist_{ij}$ is the distance between documents d_i and d_j and $\overline{dist}$ is the average distance among all documents after term t is removed. The time complexity of *Entropy Rank* (ER) is $O(N^2M)$ for each term, where M is the number of terms. Using this score is impractical when there is a large number of documents in a corpus [75].

Term Contribution

In this method, the quality of each term is computed as its overall contribution to the document similarities [75]. The *Term Contribution* (TC) score of term t is defined as:

$$TC(t) = \sum_{i,j\cap i\neq j} f(t,d_i)\cdot f(t,d_j) \tag{2.8}$$

where $f(t,d_i)$ is the feature value of term t in document d_i, which is *TFIDF* in this work. The time complexity of this method is $O(N^2)$ for each term.

Once only the presence of terms in documents are important, we set $f(t,d) = 1$ if term t appeared in document d and zero otherwise. This results in a binary representation of a corpus. In this case, the value of TC can be rewritten as:

$$TC(t) = DF(t)(DF(t)-1) \tag{2.9}$$

where $DF(t)$ is the document frequency of the term t. Despite of its simplicity, $DF(t)$ is an effective feature selection in text categorization [116]. Equation (2.9) shows that $DF(t)$ is a special case of *Term Contribution*.

We use these unsupervised feature selection methods not only for reducing the dimensionality of our dataset document-term matrices, but also to distill our term

clusters. After term clusters are generated, we perform unsupervised feature selection on each term cluster to remove its non-discriminative terms. Feature selection is thus applied globally on the dataset matrices and locally on the term clusters in our work. We have also reported an experiment to compare the above-mentioned unsupervised feature selection techniques in Section 5.2.5.

2.2 Distance Metrics

In this section, we review some distance (similarity) measures which are widely used in partitional text document clustering algorithms. We assume that documents are represented as a vector of terms $\overrightarrow{d}$, where each entry of the vector specifies the degree of relevance of a term in the respective document.

Euclidean Distance

Given two documents $\overrightarrow{d_a}$ and $\overrightarrow{d_b}$, the Euclidean distance[2] is measured as:

$$D_E(\overrightarrow{d_a}, \overrightarrow{d_b}) = (\sum_{t=1}^{M} |F_{t,a} - F_{t,b}|^2)^{0.5} \tag{2.10}$$

where $F_{t,a}$ is the feature value of term t in document $\overrightarrow{d_a}$. The minimum value of this measure is zero.

Cosine Similarity

The cosine of the angle between two document vectors $\overrightarrow{d_a}$ and $\overrightarrow{d_b}$ is used in this measure. Cosine similarity[3] is measured using the following formula:

$$\text{SIM}_C(\overrightarrow{d_a}, \overrightarrow{d_b}) = \frac{\overrightarrow{d_a} \cdot \overrightarrow{d_b}}{|\overrightarrow{d_a}||\overrightarrow{d_b}|} \tag{2.11}$$

This measure is independent of document length and its value is bounded in [0 1]. When the angle between two documents is zero, they have the maximum cosine similarity.

[2] http://en.wikipedia.org/wiki/Euclidean_distance
[3] http://en.wikipedia.org/wiki/Cosine_similarity

Jaccard Coefficient

This similarity metric is defined as the sum weight of common terms to the sum weight of terms that appear in either of documents but not in the common terms. Jaccard Coefficient is defined as [56]:

$$\text{SIM}_J(\overrightarrow{d_a}, \overrightarrow{d_b}) = \frac{\overrightarrow{d_a} \cdot \overrightarrow{d_b}}{|\overrightarrow{d_a}|^2 + |\overrightarrow{d_b}|^2 - \overrightarrow{d_a} \cdot \overrightarrow{d_b}} \tag{2.12}$$

This measure has the minimum value of zero and the maximum value of 1.

Pearson Correlation Coefficient

The similarity between two documents $\overrightarrow{d_a}$ and $\overrightarrow{d_b}$ is measured by the following formula in this metric [56]:

$$\text{SIM}_J(\overrightarrow{d_a}, \overrightarrow{d_b}) = \frac{M \sum_{t=1}^{M} F_{t,a} \times F_{t,b} - TF_a \times TF_b}{[M \sum_{t=1}^{M} F_{t,a}^2 - TF_a^2]^{0.5} \times [M \sum_{t=1}^{M} F_{t,b}^2 - TF_b^2]^{0.5}} \tag{2.13}$$

where $TF_a = \sum_{t=1}^{M} F_{t,a}$ and $TF_b = \sum_{t=1}^{M} F_{t,b}$. The range of this measure is from +1 to -1.

Kullback-Leibler Divergence

Treating each document as a probability distribution of terms, the Kullback-Leibler Divergence, or the relative entropy, is defined as a distance metric as [56]:

$$D_{KL}(\overrightarrow{d_a} || \overrightarrow{d_b}) = \sum_{t=1}^{M} F_{t,a} \times \log(\frac{F_{t,a}}{F_{t,b}}) \tag{2.14}$$

The minimum value of this measure is zero.

A comprehensive evaluation based on the above-mentioned similarity (distance) measures is performed in [56]. The conclusion of this evaluation is that the performance of Cosine similarity, Jaccard Coefficient, and Pearson Correlation Coefficient is very close and is significantly better than Euclidean distance. Clusterings obtained based on Kullback-Leibler Divergence have similar quality compared to the results of other measures.

The underlying clustering algorithm in this evaluation is k-means. The quality of clusterings is measured using entropy. The entropy of cluster C_i with size n_i is defined

as [56]:

$$E(C_i) = -\frac{1}{\log K} \sum_{h=1}^{K} \frac{n_i^h}{n_i} \log \frac{n_i^h}{n_i} \tag{2.15}$$

where n_i^h is the number of documents from the hth class in cluster C_i.

We also perform a similar evaluation based on the clustering algorithm proposed in this thesis in Section 5.2.6. Our evaluation shows that Euclidean distance can generate comparable results to Cosine similarity metric. This is mainly because Euclidean distance is related to Cosine similarity if the length of documents is normalized by $L2$ norm:

$$\begin{aligned}
D_E(\overrightarrow{d_a}, \overrightarrow{d_b}) &= (\sum_{t=1}^{M} |F_{t,a} - F_{t,b}|^2)^{0.5} \\
{D_E}^2(\overrightarrow{d_a}, \overrightarrow{d_b}) &= (\sum_{t=1}^{M} |F_{t,a} - F_{t,b}|^2) \\
{D_E}^2(\overrightarrow{d_a}, \overrightarrow{d_b}) &= (\sum_{t=1}^{M} |F_{t,a}|^2) + (\sum_{t=1}^{M} |F_{t,b}|^2) \\
&\quad -2 \times (\sum_{t=1}^{M} F_{t,a} \times F_{t,b}) \\
{D_E}^2(\overrightarrow{d_a}, \overrightarrow{d_b}) &= 1 + 1 - 2 \times (\sum_{t=1}^{M} F_{t,a} \times F_{t,b}) \\
{D_E}^2(\overrightarrow{d_a}, \overrightarrow{d_b}) &= 2(1 - \mathrm{SIM}_C(\overrightarrow{d_a}, \overrightarrow{d_b}))
\end{aligned} \tag{2.16}$$

2.3 Unsupervised Clustering

This section reviews double and co-clustering, topic modeling, and some greedy clustering algorithms that have been proposed for text clustering.

2.3.1 Double Clustering

Besides dimensionality reduction techniques, the other solution to address the problem of high dimensionality in text document representation is term clustering. Terms that exist in a corpus are clustered. Each document is then presented as a vector of term clusters instead of a vector of terms. Document clustering is thus performed in a lower

dimensional space. In general, there are two main approaches using term clusters in document clustering: double clustering and co-clustering.

Double clustering refers to algorithms that perform term clustering before document clustering [78, 102]. Given the term clusters, the document-term matrix can be represented in a more compact way based on the presence of the term clusters in the documents [102]. Besides, term clusters can be used in finding topic keyterms [89, 90]. In our proposed term labeling approach, these topic keyterms can help users to guide the clustering process toward their desired clusters. We propose our interactive term labeling based document clustering algorithm in Chapter 7.

Simultaneous clustering of terms and documents is referred to as *co-clustering* in the literature [7, 23, 34]. It may also be refereed to as bi-clustering [66, 100], two-mode clustering [82, 45], or subspace clustering [109]. A *co-clusterer* maps documents to document clusters and terms to term clusters simultaneously. The document-term matrix is viewed as an empirical joint distribution between two variables, the set of documents and the set of terms [102]. A co-cluster is then defined as a subset of documents associated with a subset of terms. The optimal solution is typically the one that has minimum loss of mutual information between documents and terms after co-clustering. We review top double clustering and co-clustering algorithms in the rest of this section.

A double clusterer for text data is proposed in [102]. The idea behind this algorithm is that instead of clustering documents by their terms, it is better to cluster them by using term clusters. The algorithm first clusters terms such that the term clusters preserve most of the mutual information on documents. The document-term matrix is then replaced by a document-term-cluster matrix. The algorithm subsequently clusters documents such that the document clusters preserve most of the information on term clusters. Suppose that D is the set of documents and T is the set of terms. For each document of the collection, one can compute term conditional distribution using the following equation:

$$p(t|d) = \frac{n(t|d)}{\sum_{t \in T} n(t|d)} \quad \text{and} \quad p(d) = \frac{1}{|D|} \tag{2.17}$$

where $n(t|d)$ is the occurrence frequency of term t in document d, and $|D|$ is the number of documents in the collection. The mutual information between D and T is

also defined as:

$$I(D;T) = \sum_{d\in D, t\in T} p(d)p(t|d)\log\frac{p(t|d)}{p(d)} \tag{2.18}$$

In the first stage, the algorithm starts from $|T|$ term clusters $\tilde{t}$. Each of these term clusters contains only one term. At each step, two term clusters are merged such that it causes the minimum loss of mutual information $I(D;\tilde{T})$. After generating term clusters, each document is represented by its term clusters conditional distribution:

$$p(\tilde{t}|d) = \frac{n(\tilde{t}|d)}{\sum_{\tilde{t}\in\tilde{D}} n(\tilde{t}|d)} = \frac{\sum_{t\in\tilde{t}} n(t|d)}{\sum_{\tilde{t}\in\tilde{Y}}\sum_{t\in\tilde{t}} n(t|d)} \tag{2.19}$$

Following the same approach, document clustering is performed such that at each step merging two document clusters result in minimum loss of mutual information $I(\tilde{D};\tilde{T})$. Experimental results show that double clustering reveals the structure of document clusters better than direct document clustering without any term clustering [102].

The main issue about the above clusterer is that it is not known when one should stop the hierarchical term clustering. No method is proposed to find the optimal number of term clusters. In the evaluation part of the paper, different numbers of term clusters are used to find the best results based on document class labels. The other issue is how to use the double clustering algorithm in partitional clustering approaches. It has been shown that the partitional clustering approach generates better text document clusters than the hierarchical clustering approach [52, 56, 119].

We will propose a partitional double clustering algorithm in Chapter 5. We remove general terms from term clusters using a feature selection method. We also make a simplifying assumption that the number of term clusters should be the same as the number of document clusters. Based on the experiments on some real text corpora, our algorithm outperforms the above clustering algorithm.

A double clusterer is proposed as a semi-supervised document clustering in [78]. The algorithm is based on the assumption that an expert provides a few labeled documents for each topic (cluster). Fuzzy c-means [13] is initialized by using the keyterms of the labeled documents so as to cluster terms. The documents are then clustered based on the contribution of their terms in the term clusters using the following formula:

$$d_i \in c_j \quad \text{if} \quad j = \text{argmax}_p \sum_{l=1}^{M} f(t_l, d_i)u_{pl} \tag{2.20}$$

where M is the number of terms in the dataset, u_{pl} is the membership value of term t_l in term cluster c_p, and $f(t_l, d_i)$ is the document-term matrix value of term t_l in document d_i. Compared to this algorithm, we cluster documents using the seed documents extracted from term clusters. We will show in Section 4.3 that our evolutionary clustering algorithm outperforms this algorithm on some real document collections.

A double clustering algorithm based on topic keyterm clusters is proposed in [20]. Each document is first pre-processed so as to identify meaningful phrases (multi-word keyterms). All the extracted phrases are then used to form a weighted undirected graph. Each node of this graph corresponds to a keyterm and the weight of edges are derived from keyterm correlations using the following equation:

$$correlation(t_i, t_j) = \frac{fr(t_i \cap t_j)}{\max(fr(t_i), fr(t_j))} \tag{2.21}$$

where $fr(t_i \cap t_j)$ is the number of documents in which keyterms t_i and t_j co-occur. The assumption of this study is that highly co-occurring terms can characterize topics.

The average edge weight of the graph is first computed and used as a threshold to remove all the edges with low weights. Central keyterms are then extracted from the graph. A central keyterm v_i has high composite weight which is calculated using the following formula:

$$\text{compositeWeight}(v_i) = \sum_{d_i} TFIDF_i + \frac{1}{m} \sum_{j=1}^{m} r_{ij} \tag{2.22}$$

where d_i is a document that keyterm v_i appears in, m is the degree of node v_i in the graph, and r_{ij} is the weight of edge $\langle v_i, v_j \rangle$. The k nearest neighbor algorithm is then applied on the central keyterms to create central keyterms clusters. The cosine similarity between keyterms of the clusters and document contents are finally used to cluster documents. This part of the algorithm is not explained clearly in the respective paper. It is also not mentioned how the value of k in k-nearest neighbor algorithm affects the quality of clusterings. Moreover, the algorithm is not compared with any double clustering algorithm and no implementation is available for comparison.

A co-clustering algorithm based on mutual information is proposed in [34]. Information-Theoretic Co-clustering views a data matrix as an empirical joint distribution of two random variables, the set of rows and the set of columns. The goal is to

preserve the mutual information between these two variables as much as possible due to clustering rows and columns. Given a co-clustering, one reorders the rows of the matrix in a way that all rows belonging to the first row-cluster come first, followed by those of the second cluster, and so on. The columns of the matrix are reordered in a similar way. This reordering divides the joint distribution into blocks or co-clusters. Hence, the joint distribution is represented as a joint distribution of the blocks. The loss in mutual information before co-clustering and after co-clustering is computed using the following formula:

$$I(D;T) - I(\hat{D};\hat{T}) \tag{2.23}$$

where $\hat{D}$ is a row partitioning and $\hat{T}$ is a column partitioning. Since $I(D;T)$ is fixed, a good co-clustering that maximize $I(\hat{D};\hat{T})$ is desired. Given the joint distribution $p(D,T)$, it has been proved that finding the best co-clustering is equivalent to finding the closest distribution q to p in *Kullback-Leibler* divergence [34]:

$$q(d,t) = p(\hat{d},\hat{t})p(d|\hat{d})p(t|\hat{t}) \tag{2.24}$$

$$p(\hat{d},\hat{t}) = \sum_{d\in\hat{d}}\sum_{t\in\hat{t}} p(d,t) \quad p(\hat{d}) = \sum_{d\in\hat{d}} p(d) \quad p(d|\hat{d}) = \frac{p(d)}{p(\hat{d})} \tag{2.25}$$

where $\hat{d}$ is a cluster in row partitioning $\hat{D}$. Given the above formulas, the row-cluster prototypes $q(T|\hat{d})$ and column-cluster prototypes $q(D|\hat{t})$ are formulated. Given the joint probability distribution $p(D,T)$, the number of row clusters k, and the number of column clusters l, the partitional algorithm generates co-clusters in the following way:

1. Create initial row and column partitionings randomly.
2. Compute row cluster prototypes $q(T|\hat{d})$ for $1 \le \hat{d} \le k$.
3. Re-cluster rows based on the $q(T|\hat{d})$.
4. Compute column cluster prototypes $q(D|\hat{t})$ for $1 \le \hat{t} \le l$.
5. Re-cluster columns based on the $q(D|\hat{t})$.

6. Stop if $D(p \parallel q^{(t)}) - D(p \parallel q^{(t-1)})$ is smaller than threshold σ, otherwise go to step 2.

The algorithm can be applied on the document-term matrices of text. The main issue about the algorithm is that there is no way to find the optimal number of column clusters l. In the evaluation part of the paper, the value of k is set as the true number of document classes and different values for l are considered to find the best results. We will show in Section 4.3 that our evolutionary algorithm outperforms this algorithm on some real text datasets.

An output of a microarray experiment can be represented as a gene-condition matrix. The rows of the matrix correspond to genes and the columns are the environmental conditions. Each entry of the matrix is a number, representing the activity of a gene under a certain condition. A co-cluster of this matrix reveals a pattern of a group of genes under certain conditions. Two partitional co-clustering algorithms for gene expression data are presented in [23]. The co-clustering problem is defined as a co-clustering of rows and columns of the data matrix. Given a co-clustering, two squared *residue* measures are defined to compute its homogeneity:

- The first measure is the sum of squared distances between each entry of a co-cluster and the mean of the co-cluster.
- The second measure is the sum of squared distances between each entry of a co-cluster and the corresponding row mean and column mean that entry belongs in.

In order to find the minimum values of these two measures, two co-clustering algorithms similar to k-means are proposed. At each iteration of the algorithms, columns are re-clustered into l column clusters. Hence, rows are re-clustered into k row clusters based on the column clusters just created. Like the algorithm mentioned above, no method to find the optimal number of term clusters is provided in this algorithm. Please refer to Section 4 of the paper for more details.

A general framework for matrix approximation is presented in [7]. The framework is a generalization of the algorithms proposed in [34, 23] The co-clustering problem is viewed as a special case of matrix approximation of an input data matrix where the quality of co-clustering is obtained by an approximation error. The best co-clustering

generates an approximation of the input matrix with minimum error. The error is measured using *Bregman* divergences like *I-Divergence, Squared Euclidean Distance, and Itakura-Saito Distance*.

Six different co-clustering bases are introduced to form an approximation matrix in this work. Each basis is like a constraint that preserves different summary statistics of the input data matrix. Given a co-clustering, the rows and columns of the input matrix are first reordered such that the rows of each row-cluster and the columns of each column-cluster sit next to each other. Each co-cluster looks like a block of the input matrix. An approximation matrix is then created based on the selected basis.

One way of approximation, for instance basis $C2$, is that each entry of the approximate matrix is the mean of all entries existing in the corresponding block. In this way the approximation preserves the co-cluster means. In basis $C5$, average row and column means are also preserved beside the co-cluster means. For more information about the basis and the approximation schemes, please refer to Section 4 of [7]. The dimensionality of the approximate matrix is k by l, where k is the number of row-clusters and l is the number of column-clusters. It has been mentioned in the paper that two bases $C2$ and $C5$ are appropriate for text clustering [7]. We have compared our evolutionary clustering algorithm with four algorithms of this framework in Section 4.3.

All the double clustering and co-clustering algorithms reviewed in this section are based on hard partitioning. Moreover, the numbers of row-clusters and column-clusters are assumed independent and user-defined. To address these issues, two clustering algorithms are proposed in [112] based on information bottleneck clusterings proposed in [34, 102]: Information Bottleneck Co-Clustering (*IBCC*) and agglomerative Information Bottleneck Co-Clustering (*aIBCC*).

Similar to the *ITCC* algorithm [34], *IBCC* is a partitional co-clustering algorithm. The main difference between these two algorithms is that, *ITCC* is based on hard partitioning but *IBCC* is based on soft partitioning. This benefit is achieved by defining a different objective function for *IBCC*. *IBCC* is also enhanced by a simulated annealing approach to bypass local minima.

Compared to the agglomerative double clustering algorithm proposed in [102], *aIBCC* is a co-clustering agglomerative algorithm. In each iteration of *aIBCC*, either

two document clusters or two term-clusters are merged based on a merge cost. It has been claimed in [112] that this mechanism captures the relation between the numbers of document clusters and term clusters.

2.3.2 Greedy Direct Clustering

A greedy unsupervised clustering algorithm similar to k-means is proposed in [118] for text documents. The algorithm starts with random selection of k seed documents. Each document is then assigned to the cluster with the most similar seed. After this initial clustering, an iterative refinement phase starts. In each iteration of this phase, documents are revisited randomly and each one is moved to a new cluster provided that the reassignment improves the value of the criterion function. The refinement stage stops if no improvement could be made in an iteration.

A bisecting version of the algorithm is also proposed. The bisecting clustering algorithm initially bisects the entire collection. One of the two clusters is then selected and bisected to generate three clusters. This process is repeated until a total number of k clusters are generated. We used four algorithms of this work, which are efficient in text clustering based on our experiments. They include two direct and two bisecting methods based on the following I_2 and H_2 criteria:

$$I_2\text{: maximize } \sum_{p=1}^{k} \sum_{d_i \in C_p} \cos(d_i, \mu_p) \tag{2.26}$$

$$E_1\text{: minimize } \sum_{p=1}^{k} n_p \cos(\mu, \mu_p) \tag{2.27}$$

$$H_2\text{: maximize } \frac{I_2}{E_1} \tag{2.28}$$

$$\mu = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{2.29}$$

where μ_p is the centroid of document cluster C_p, n_p is the number of documents in C_p, μ is the centroid of all documents, and N is the number of documents in a dataset. No term clustering is used in these direct clustering algorithms. We have compared our proposed clustering algorithms to these algorithms in order to see whether term clustering is always effective in text document clustering.

2.3.3 Topic Modeling

Probabilistic models are widely used in different areas of text mining such as document clustering [17, 97], document classification [17, 62, 104], summarization [3, 21], tag recommendation [94], information retrieval [68, 115], and dimensionality reduction [27].

A well-known probabilistic model for text document clustering is Topic Modeling. Topic Modeling creates a generative process in order to generate the terms in the documents of a collection. The main assumption behind any topic modeling is that a document belongs to multiple topics with different degrees of membership. Similarly, each term is related to multiple topics with different degrees. Topic Modeling finds the latent topics, which correspond to the document clusters in the underlying corpus.

The output of the model is in the form of probability values: the probability that a document belongs to a topic, and the probability that a term related to a topic. Based on the probability values, a soft or hard partitioning of the documents along with their keyterms can be created. Two well-known Topic Models are Probabilistic Latent Semantic Indexing (*PLSI*) [48] and Latent Dirichlet Allocation (*LDA*) [17].

Probabilistic Latent Semantic Indexing

PLSI is an extension of the feature transformation method proposed in [32] as Latent Semantic Indexing (*LSI*). As the *LSI* method projects documents and terms into a lower dimensional space, *PLSI* projects them into a latent topics (semantic) space. The generative process of *PLSI* for each word w in document d is based the following steps [48]:

1. Select a document based on the multinomial distribution $p(d)$
2. Select a topic $i \in 1, 2, .., k$ based on the topic distribution $p(z = i|d)$
3. Sample a word v based on $p(w = v|z = i)$

where k is number of clusters or topics specified by the user in advance, and the joint probability distribution $p(v, d)$ is expressed as:

$$p(v, d) = p(v|d)p(d) \text{ and } p(v|d) = \sum_{i=1}^{k} p(z = i|d)p(v|z = i) \tag{2.30}$$

The other way to express the joint distribution [27], which reveals the connection of *PLSI* to *LSI* is defined as:

$$p(v, d) = \sum_{i=1}^{k} p(z = i)p(d|z = i)p(v|z = i) \tag{2.31}$$

where $p(d|z = i)$ and $p(v|z = i)$ are the projections of documents and terms into the latent topics space.

Given a document-term matrix X with N rows (document vectors) and M columns (term vectors) as the observed data, a log-likelihood of the model can be expressed as [27]:

$$L = \sum_{d=1}^{N} \sum_{v=1}^{M} x_{dv} \log \sum_{i=1}^{k} p(w = v|z = i)p(z = i|d)p(d) \tag{2.32}$$

where x_{dv} is the feature value of the word v in the document d, and $p(d)$, $p(z|d)$, and $p(v|z)$ are the parameters to be estimated. The parameter estimation or learning of the model is then performed for instance by maximizing the log-likelihood function using the expectation-maximization approach [33].

Latent Dirichlet Allocation

The idea behind *LDA* is that a document can be viewed as a probability distribution over latent topics and each topic in turn is viewed as a probability distribution over terms. Given the document-term matrix, and parameters k and α, the output of the model are two matrices: a document-topic matrix with dimensionality N by k and a topic-term matrix with dimensionality k by M.

A graphical model of the *LDA* is shown in Fig. 2.1. Each random variable is represented by a circle. The only observed variable, which is term, is shaded. Each arrow in the model indicates that the value of the pointed variable depends on the value of the pointing variable. Each plate and the value inside it indicate the number of times the sampling should be performed. Based on the graphical model, the generative process of *LDA* is performed in the following way [17]:

- For each document, the probability distribution over topics is a Multinomial distribution drawn from a Dirichlet distribution with parameter α.
- For each topic, the probability distribution over terms is a Multinomial distribution drawn from a Dirichlet distribution with parameter β.

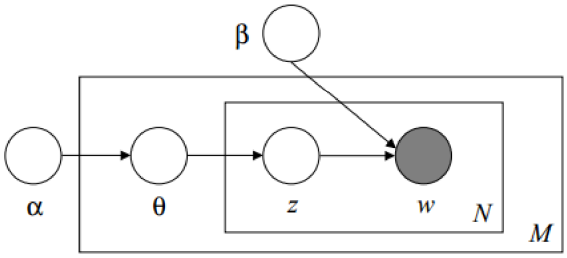

Figure 2.1: The *LDA* graphical Model [17]

- For each term in a document:

  1. A topic is chosen based on the document topic distribution.
  2. The term is chosen based on the topic term distribution.

The idea behind the generative process is based on the co-occurrence of terms in the documents [17]. It means that if the probability of term t_i for topic c_l is high and term t_j co-occurs with term t_i frequently, the term t_j is more likely related to the topic c_l. Using the co-occurrence of terms helps *LDA* to find the latent topics, but it has a disadvantage. *LDA* will favor generating a topic including all the sub-topics if the sub-topics share similar keyterms [17]. For example *LDA* will favor generating a cluster of *sport* instead of generating four clusters of *baseball*, *basketball*, *football*, and *volleyball* if they share common sport terms.

Another probabilistic generative model for text document clustering is proposed in [59]. The idea behind this model is that only some of the terms in a document are related to its cluster (topic keyterms), and the other terms follow a general term distribution shared among documents (general terms). Therefore, each document is modeled as a mixture of two multinomial distributions, one for its cluster and the other shared in the corpus. Moreover, a document is not modeled as a distribution over topics but it is associated with a single topic.

We have compared our partitional clustering algorithm to the *LDA* model proposed in [17]. The experimental results have been reported in Section 5.2.2.

Probabilistic models are also used for co-clustering [99, 111]. Almost all existing co-clustering algorithms like those reviewed in Section 2.3.1, are based on hard partitioning of rows and columns. Each row is assigned to one row-cluster and each

column is assigned to one column-cluster. However, probabilistic model based co-clustering approaches consider a probability distribution membership in a row- and column-cluster [99, 111]. Dirichlet distribution is used for prior distributions of row- and column-clusters.

Given an $N \times M$ document-term matrix, k_1 row clusters $\{z_1 = i, [i]_1^{k_1}\}$, and k_2 column clusters $\{z_2 = j, [j]_1^{k_2}\}$, the generative process of Bayesian co-clustering [99] is as follows:

1. For each row d, the probabilistic distribution is drawn from a Dirichlet distribution with parameter α_1, π_{1d}.

2. For each column t, the probabilistic distribution is drawn from a Dirichlet distribution with parameter α_2, π_{2t}.

3. The entry x_{dt} in row d and column t of the matrix is generated as:

   - choose z_1 from discrete distribution $Disc(\pi_{1d})$
   - choose z_2 from discrete distribution $Disc(\pi_{2t})$
   - choose x_{dt} from $p(x|\theta_{z_1 z_2})$

where $\theta_{z_1 z_2}$ is the parameter of the model for co-cluster(z_1, z_2). A variational EM-based algorithm is proposed for inference and parameter estimation in this model.

Latent Dirichlet Bayesian co-clustering [111] smooths the Bayesian co-clustering model [99] by introducing a prior for $\theta_{z_1 z_2}$. The prior is drawn from a Dirichlet distribution with parameter β. Given an $N \times M$ document-term matrix, k_1 row clusters $\{z_1 = i, [i]_1^{k_1}\}$, and k_2 column clusters $\{z_2 = j, [j]_1^{k_2}\}$, the generative process of Latent Dirichlet Bayesian co-clustering is as follows[111]:

1. For each row d, the probabilistic distribution is drawn from a Dirichlet distribution with parameter α_1, π_{1d}.

2. For each column t, the probabilistic distribution is drawn from a Dirichlet distribution with parameter α_2, π_{2t}.

3. The entry x_{dt} in row d and column t of the matrix is generated as:

   - choose z_1 from discrete distribution $Disc(\pi_{1d})$

- choose z_2 from discrete distribution $Disc(\pi_{2t})$
- choose $\theta_{z_1 z_2}$ from a Dirichlet distribution with parameter β
- choose x_{dt} from $p(x|z_1, z_2, \theta_{z_1 z_2})$

where $\theta_{z_1 z_2}$ is the parameter of the model for co-cluster(z_1, z_2). A collapse Gibbs sampling and a collapse variational inference are proposed for parameter estimation in this model.

Similar to these two model based co-clusterers, our double clustering algorithm (LDC) which is proposed in Chapter 5, is based on soft partitioning of terms and documents. A soft partitioning of terms is first generated by fuzzy c-means and a defuzzification method. Seed documents of each term cluster are then extracted and used to generate document centroids. Each document is finally assigned to each document cluster with a membership value. The membership values are computed based on the similarities of documents to the document centroids.

Different from these model based co-clusterers, LDC is based on double clustering approach and no probabilistic model is used to generate term or document clusterings. LDC is a partitional double algorithm proposed for text document clustering.

2.4 Enhancing Text Clustering using Wikipedia

Several research works have been proposed to integrate Wikipedia in text document clustering. We review some of these works related to our research in this section.

Traditional document content similarity is leveraged by integrating Wikipedia based semantic relations in [51]. A concept thesaurus is first created from Wikipedia articles. It includes semantic relations like synonymy, polysemy, hypernymy, and associative relations extracted from anchor texts, disambiguation pages, categories, and hyperlinks in Wikipedia. Given a document, its text content is first mapped to the most relevant Wikipedia articles. The category hierarchy of those articles is then used to form a category vector. A concept vector is also created based on the concepts mentioned in the relevant articles using the thesaurus synonym and associative relations. The document similarity measure is then defined as:

$$S_{\text{combination}} = (1 - \alpha - \beta)S_{content} + \alpha S_{cat} + \beta S_{conc} \tag{2.33}$$

where, $S_{content}$ is the cosine similarity based on content vectors, S_{cat} is based on the category vectors, and S_{conc} is based on the concept vectors. Equal weights ($\alpha = \beta = 1/3$) are considered in empirical experiments. Given a few labeled documents, a parameter optimization is also performed to find the optimal values for α and β. The optimal weights could improve the clustering performance further.

A similar approach is proposed to enrich the document representation using Wikipedia concepts in [52]. A document collection is represented as a document-term matrix, a document-concept matrix, and a document-category matrix. The traditional document cosine similarity is then enhanced by using the concept and category matrices as:

$$\begin{aligned} sim(d_i, d_j) = sim(d_i, d_j)^{term} + \alpha sim(d_i, d_j)^{concept} \\ +\beta sim(d_i, d_j)^{category} \end{aligned} \tag{2.34}$$

where d_i and d_j are two documents. No optimization method is proposed to find the optimal values for α and β. After using different values, the best improvement is reported in experimental results. Compared to the method proposed in [51], lower weights are considered for concepts and categories. An interesting observation in experimental results is that document-concept and document-category representations used alone never outperform the document-term representation in partitional document clustering.

A concept based similarity measure is proposed in [58], which considers the semantic relatedness among concepts. The semantic similarity between two documents d_i and d_j is defined as:

$$Sim(d_i, d_j)^{sem} = \frac{\sum_{c_k, c_l} w(c_k, d_i) w(c_l, d_j) SIM(c_k, c_l)}{\sum_{c_k, c_l} w(c_k, d_i) w(c_l, d_j)} \tag{2.35}$$

where $SIM(c_k, c_l)$ is the semantic relatedness between two concepts c_k and c_l extracted from Wikipedia articles, and $w(c_k, d_i)$ is the weight of concept c_k in vector d_i in document-concept representation. The overall document similarity is then defined as the linear combination of the semantic similarity and the cosine similarity based on document contents. Lower weights are considered for semantic similarity compared to content similarity. The main contribution of the work is in Eq. (2.35), where the semantic similarity between documents is enhanced by using the semantic relatedness between concepts ($SIM(c_k, c_l)$).

Wikipedia categories are also used in [96] to enhance text document clustering. Four different document representation techniques are evaluated for this purpose:

1. A document is represented in the *BOW* model
2. A document is represented using Wikipedia categories
3. A document is represented only by the top 20 keyterms with the highest weight values
4. A document is represented as a combination of the categories and the top 20 keyterms.

The worst results are obtained when documents are only represented by Wikipedia categories. The *BOW* model and the combined representation result in the same clusters and no significant improvement is achieved by integrating Wikipedia categories.

The titles of relevant Wikipedia articles are appended to the content of documents in [8]. The *BOW* model is then used to represent documents with doubling the weights of the terms appearing in the titles. The proposed clustering method is applied on short texts and significant improvement is obtained after integrating the Wikipedia articles. An interesting observation is that the weight of Wikipedia title terms is double, while a lower or at most equal weights are considered in the other methods reviewed so far. The work demonstrated that integrating Wikipedia in document representation can be very effective, when documents are in the form of short texts like tweets or snippets.

A random walk model is proposed in [117] to measure the semantic relatedness among documents. Wikipedia articles are first mapped to a graph, where nodes correspond to articles and edges derived from either hyperlinks among articles or the cosine similarity among the articles' text. A document is then mapped to the 10 closest nodes (articles) based on the cosine similarity of its content and the articles' text. The semantic relatedness among documents is then computed as a Visiting probability of a random walk on the graph. The Visiting probability is computed using the probability of transition t_{ij} between nodes s_i and s_j and is defined as:

$$P(t_{ij}) = \frac{W(i,j)}{\sum_{k=1}^{n} W(i,k)} \tag{2.36}$$

where n is the number of nodes in the article graph, and $W(i, j)$ is the weight of the edge between s_i and s_j. A document similarity matrix is then formed by using the Visiting probability computed on the graph. The similarity matrix is fed into a relational k-means algorithm to cluster documents. The best improvement is obtained by computing the Visiting probability over a combination of the hyperlinks and the lexical similarity among the articles' text.

A conceptual hierarchical clustering using the relevant concepts extracted from Wikipedia is proposed in [103]. Given a document, its noun-phrases are initially mapped to the relevant Wikipedia concepts. Besides concepts, relevant Wikipedia articles are also extracted in order to define the following conceptual features:

- The frequency of a concept in a document.
- The number of common links presented in the article of a relevant concept and in the articles of all relevant concepts.
- The cosine similarity between the document content and the article text.
- The position of a relevant concept in the document.
- The importance of a concept in Wikipedia regardless of the document referred to.

A linear combination of the above features is then used to measure the importance of a particular concept in the respective document. Given the conceptual representation, a hierarchical clustering algorithm is used to cluster documents.

We have also proposed an ensemble approach in Chapter 6 to integrate Wikipedia concepts in document clustering. Our experimental results show the effectiveness of our approach even when the quality of clusters generated by the Wikipedia concepts is poor.

2.5 User Supervised Clustering

Finding the best clustering algorithm to partition a document collection is not a trivial task in practice. No ground truth about the collection is usually available and there is no guarantee that the automatically-generated clustering matches the user's

preferences. We need to involve users in the clustering process and let them evaluate the generated clusters. We have reviewed some user-supervised document clustering algorithms in this section.

2.5.1 Document Level Supervision

The input of the COP-KMEANS algorithm is a dataset plus some *must-link* or *cannot-link* pairwise constraints [110]. A *must-link* constraint specifies that a pair of instances must be in same cluster. Two instances with a *cannot-link* constraint must be always in different clusters. COP-KMEANS is similar to k-means except in assigning instances to the closest centers. During the steps of COP-KMEANS, the constraints are never broken. For each constrained instance, COP-KMEANS tries to assign it to the closest centers such that none of the constraints are violated. An empty partitioning will be returned if there is no way of preserving constraints. The constraints can be determined either by asking directly from users or by eliciting them from the seed documents labeled by users.

Two semi-supervised variants of k-means are proposed in [9]. A set of training documents is chosen and the user is asked to label them. The labeled documents are then used in the k-means algorithm as seed documents to initialize the cluster centers. In one algorithm, seed documents are only used to initialize k-means and the label of seed documents might change in the subsequent steps. In the other one, after initialization by the seed documents, their labels never re-computed during the clustering process. The algorithm preserves the label of seed documents during the step of assigning instances to the cluster centers. No method is proposed to actively select the training documents. The user supervision is limited to the initial supervision and no further interaction is considered.

The MPCK-MEANS algorithm allows violation of the *must-link* and *cannot-link* constraints [15]. Each violation has a penalty cost, which is added to the objective function of k-means. Let ML be a set of *must-link* and CL be a set of *cannot-link* constraints. Moreover, w and $\bar{w}$ are the sets of penalty costs for violating constraints in ML and CL, respectively. The objective function of the algorithm is then defined

as:

$$\zeta = \sum_{x_i} \| x_i - \mu_{c_i} \|^2 + \sum_{x_i, x_j \in ML, c_i \neq c_j} w_{ij} + \sum_{x_i, x_j \in CL, c_i = c_j} \bar{w}_{ij} \tag{2.37}$$

The penalty costs are independent of the distance of instances given the above objective function. For instance, the same penalty cost is considered for two close instances and two far instances.

The MPCK-MEANS algorithm considers the distance of instances in case a constraint violation occurs. If a *must-link* constraint is broken and the instances are far apart, there is a larger penalty cost. If a *cannot-link* constraint is broken and the instances are so close, there is a larger penalty cost. The violation costs are thus multiplied by the distance of instances from each other. The strength of the algorithm is that it is not restricted to preserve all the constraints. However, no method is proposed to specify the penalty costs w and $\bar{w}$ a priori.

An active method is proposed in [10] to select the best pairs of instances to elicit pair-wise constraints from users. The method is based on a *farthest-first* policy. A starting instance is first selected randomly and added to the supervision set. The next instance, which is the farthest from the set is then selected and added to the set. The distance of an instance to the set is the minimum distance of the instance to the instances that exist in the set. This process stops after a pre-defined number of instances are selected.

The user is then asked to label the selected instances. The method then generates a set of *must-link* and *cannot-link* constraints from the seed instances. The constraints are then combined in the objective function of the k-means algorithm, Eq. (2.37), where violation is allowed in exchange of penalty cost. The strength of the algorithm is that there is an active selection method to create a training set. However, the method is sensitive to the outliers, which are often far from the other instances. The user interaction is limited to the initial supervision and no further interaction is considered in this algorithm.

Pairwise constraints are also used to adjust the distance metric in [25, 65]. The idea behind adjusting the distance metric is that if documents d_i and d_j are constrained to be in the same cluster but the clustering algorithm puts them into separate clusters, we need to decrease the distance between d_i and d_j such that they fall into the same cluster. Similarly, if documents d_i and d_j must be in separate clusters, we need to

increase the distance between them if they fall in the same cluster by the algorithm.

The Constrained Complete-link algorithm [65] adjusts the pairwise document distance matrix to ensure the constraints. All the respective *must-link* entries are set to zero and the *cannot-link* entries are set to the maximum value of the matrix plus one. A hierarchical clustering algorithm is used to cluster instances.

A semi-supervised version of the EM algorithm is proposed in [25] so as to involve the pairwise constraints in the distance metric. Each term has a weight in the distance between documents d_i and d_j. Once the algorithm encounter a constraint imposed between d_i and d_j, the terms' weights are adjusted in order to increase or decrease the respective distance.

The pairwise constraints are also used for dimensionality reduction in the clustering algorithm proposed in [106]. The instances are projected into a lower dimensional space provided that the instances involved in the *cannot-link* constraints get farther and the instances involved in the *must-link* constraints get closer. The objective function that should be maximized for this purpose is defined as:

$$f = \sum_{x_i, x_j \in CL} \parallel F^T(x_i - x_j) \parallel^2 - \sum_{x_i, x_j \in ML} \parallel F^T(x_i - x_j) \parallel^2 \tag{2.38}$$

where F^T is the projection matrix that should be learned, CL is the set of *cannot-link* constraints, and ML is the set of *must-link* constraints.

Given the instances in the lower dimensional space, a constrained k-means algorithm clusters the instances such that violation is prohibited. A greedy approach is used in the k-means algorithm. First, all *must-link* constraints are replaced by some equivalent *cannot-link* constraints. Hence, for each pair of instances (x_i, x_j) in CL, two clusters μ_i, μ_j are chosen such that the following sum is maximized:

$$\cos(x_i, \mu_i) + \cos(x_j, \mu_j) \tag{2.39}$$

the other instances are assigned to the closest cluster. The optimal dimensionality of the projected space is not discussed in the paper.

The other kind of user feedbacks used in text document clustering are in the form of split/merge requests [6]. Starting with a cluster including all instances, the user either requests to split a cluster into two, or she requests to merge two clusters into one cluster. In another version of the proposed method, the user starts with one

cluster for each instance and iteratively creates a hierarchy of the clusters with split or merge requests.

The strength of the split/merge requests is that the supervision is not limited to the starting point of the clustering and the user can interact with the clustering process until the end. However, it would be more useful to include other kind of interactions to the algorithm, like changing the cluster of an instance or creating a new cluster.

2.5.2 Term Level Supervision

An active text document clustering algorithm based on frequent itemsets is proposed in [80]. A frequent itemset consists of those terms, which co-occur in documents more than a threshold value called minimum support. For a data collection with $D = \{d_1, d_2, ..., d_N\}$ documents, a set of itemsets $F = \{f_1, f_2, ..., f_p\}$ is obtained initially. Each itemset f is represented using the following formula:

$$f_v = \frac{1}{|S_f|} \sum_{\forall d \in S_f} d \tag{2.40}$$

where S_f is the set of documents in which the itemset f occurred. Given the above presentation, the algorithms clusters documents through the following two steps:

1. The k-means algorithm is first used so as to cluster the frequent itemsets. The center c of a frequent itemsets cluster C is computed using the following formula:

$$c = \frac{1}{|C|} \sum_{\forall f_v \in C} f_v \tag{2.41}$$

2. In the second step, an active learning method is used to refine the frequent itemsets clusters. For each cluster, a list of top itemsets is extracted and the user is asked to select the best itemset from the list. The itemsets of a cluster are ranked based on their distances to the cluster center and the size of S_f:

$$\text{score}(f_v, c) = |S_f| \cos(f_v, c) \tag{2.42}$$

where *cos* is the cosine similarity. The center of the cluster is then adjusted based on the selected itemset. For this purpose, the center of the cluster is

adjusted using the following formula:

$$c^{new} = c^{old} + f_v^{\text{selected\_Itemset}} \tag{2.43}$$

After refining step is over, each document is assigned to the nearest frequent itemsets cluster using the cosine similarity.

A noise-free oracle is presented in the evaluation of this algorithm to simulate user interactions. From the top itemsets of each cluster, the oracle returns the best itemset based on the true label of documents. The strength of the algorithm is that the users can adjust the cluster centers based on the desired itemsets. It means that the users are allowed to change the clusters based on the keyterms of the selected itemset. However, the users cannot create a new itemset or change their keyterms. The simulated users are assumed to never make any mistake and no experiment is provided in the paper to evaluate the effect of noisy user feedback. It is also not discussed how sensitive the quality of clusters is to the value of minimum support.

In our interactive clustering algorithm proposed in Chapters 7 and 8, we represent each group of related keyterms as a term cloud. The advantage of our approach is that the user can change the keyterms of a term cloud or even create a cloud anew. We let the user choose her desired number of document clusters. The supervised number of term clouds specifies the number document clusters in our algorithm. We have provided a comprehensive experiment to evaluate the performance of our algorithm in case of noisy user feedback.

A text clustering algorithm is proposed in [28], which is capable of producing multiple clusterings of the same collection based on different points of view. The clusterings are generated using the following way. Following a spectral clustering algorithm [101], a *Laplacian* matrix is generated using the cosine similarity among documents:

$$L = D^{-0.5}(D - S)D^{-0.5} \tag{2.44}$$

$$D_{i,i} = \sum_j S_{i,j} \tag{2.45}$$

$$S_{i,j} = \cos(d_i, d_j) \tag{2.46}$$

where D is a diagonal matrix, S is the pairwise document similarity matrix, and $D^{-0.5}$ means each entry of D is raised to the power of (-0.5). It has already been proposed that clustering the second eigenvector of L would generate a two-cluster partitioning [87] . The contribution of this paper is that it is not limited to the second eigenvector. The assumption is that clustering each eigenvector of matrix L creates a suboptimal partitioning of the documents.

Eigenvectors corresponding to the smallest second to $(m+1)$-th eigenvalues of the matrix are thus computed for this purpose. The k-means algorithm with $k = 2$ is applied on each of m eigenvectors to produce m two-cluster partitionings. For each partitioning, the top terms associated with each of two clusters are then extracted and displayed to the user. The score of each term in each cluster is computed using the following formula:

$$P(t_i|C_j) \log \frac{P(t_i|C_j)}{P(t_i|\neg C_j)} \tag{2.47}$$

where the probabilities are computed using term frequencies.

The user is then asked to inspect these terms and select those eigenvectors that resulted in meaningful clusters. The combination of the accepted eigenvectors are subsequently used to generate document clusterings. The main steps of the algorithm are given below:

1. Create the *Laplacian* matrix L.
2. Compute $E = \{e_2, e_3, ..., e_{m+1}\}$, the smallest second to $(m+1)$-th eigenvectors of L.
3. Apply k-means with $k = 2$ on each e_j so as to generate two clusters.
4. Extract the keyterms of the clusters using Eq. (2.47).
5. Ask the user to select a subset E' of E based on the extracted top keyterms.
6. Apply k-means with $k = 2$ on the documents in the space formed by the eigenvectors of E'.

The main contribution of the algorithm is that a new feature space can be generated for document clustering. In other words, the user can indirectly select the discriminative terms that should be involved in the clustering process. However, this interaction

is limited to accept or reject an eigenvector. There is no way to directly specify discriminative terms. The other issue is that the algorithm is evaluated only for bi-clustering ($k = 2$). No experiment provided in the paper in case of clustering to more than two clusters.

Our proposed algorithm lets the user directly specify the keyterms of clusters instead of just accepting or rejecting them. She can even create a new term cloud and ask the algorithm to create a corresponding document cluster.

An interactive feature selection framework for text document clustering is proposed in [53]. Given the document clusters of the current iteration, their top keyterms are extracted and displayed to the user. The user specifies discriminative ones and a feature space is subsequently formed to cluster documents in the next iteration. The main steps of the algorithm are given below:

1. Initial document clusters are obtained using k-means with the top m terms selected by the mean-*TFIDF* method. The mean-*TFIDF* is an unsupervised feature selection method described in Section 2.1.2.

2. Treating the current document clusters as classes, the χ^2 statistic is then used to create a ranked list of terms.

3. The top terms of this list are then presented to the user for supervision. The user gives one of two answers regarding each term presented:

   (a) She accepts the term as a discriminative one

   (b) She says *"I do not know"*

4. The feature set of the next iteration consists of m terms including those terms accepted by the user and the remaining top terms from the ranked list.

A noiseless feature oracle is used to simulate user interactions in this paper. Using the class label of documents, a reference feature set is first formed. The reference set consists of the top m terms based on the χ^2 statistic. The simulated user gives the answer *"accept"* if the term is in the reference set. Otherwise, the answer is *"I do not know"*.

The strength of the framework is that the user can form her desired feature space by specifying discriminative terms. The user is also involved in whole clustering

process from beginning till the end. The only issue with the proposed framework is that the simulated users are assumed to never make any mistake. However, the user may specify a noisy term as discriminative or ignore a discriminative term in practice. Therefore, an experiment is needed to evaluate the sensitivity of the proposed framework in presence of noisy feedbacks.

A feature (term) selection method to enhance semi-supervised document clustering algorithm is proposed in [54]. The user is asked to specify discriminative terms during the phase of document supervision. The supervised documents with the selected discriminative terms are then used to guide the clustering algorithm. The supervised documents are used either as seed documents to initialize k-means or they are used to extract some *must-link* and *cannot-link* constraints. The discriminative terms are in turn used to form a feature space for k-means. A feature re-weighting method is also used to bold the importance of the discriminative terms:

$$\text{TFIDF}_j = \begin{cases} \text{TFIDF}_j \times g & \text{if} \quad t_j \quad \text{is discriminative} \\ \text{TFIDF}_j & \text{otherwise} \end{cases} \tag{2.48}$$

where TFIDF_j are the feature values of term j in documents, and g is a user-defined parameter.

The strength of this algorithm is that the users have a chance for dual supervision, document level and term level supervision. Moreover, a probabilistic feature oracle is used in the evaluation of the algorithm, which simulates the user's possible mistakes in supervision. However, there are two issues regarding the proposed supervised approach. First, similar to most semi-supervised algorithms [9, 10, 15, 25, 30, 65, 106, 110], the feature selection of this algorithm is just performed once in the beginning of the clustering process and the users do not have any chance to interact with the process ever after. Second, the training terms are exposed to the user one by one. It would be better if the training terms are displayed in groups. A group of terms better reveals the topics of documents and the supervision would be in turn easier.

The experiments in Chapter 7 show that with a comparable amount of user effort, our proposed term labeling is more effective than the term selection method mentioned above.

Our work is also different from multi-view clustering approach [71, 22, 14]. Multi-view clustering is applicable to the collections where multiple views of data are available.

For instance, it can be used in clustering web pages or Wikipedia articles where one view is textual contents and the other view is hyperlinks [22]. Or, it can be used in clustering a document collection where documents' contents are available in multiple languages [71].

The idea behind multi-view clustering is that multiple independent feature spaces are available to represent data objects such that each feature space independently suffices for clustering [14]. For instance, if documents of a collection are available in multiple languages, each document can then be represented by the terms that exist in the respective language.

The input of our clustering algorithms in this thesis is a set of text documents, each represented based on the Bag of Words model. The importance of terms in documents are also measured using *TFIDF*. Documents are thus represented in one feature space formed by the terms that exist in the collection. We propose a graphical interface in Chapter 8 based on a user-supervised algorithm, which is proposed in Chapter 7. Using the interface, users are able to generate clusterings based on different points of view by specifying the topics of desired clusters in the form of term clouds.

2.5.3 Interactive Clustering Visualization

In this section, we review some graphical interfaces, which have been proposed for interactive text document clustering.

An interface to support semi-supervised clustering is proposed in [77]. The background algorithm of the interface is the semi-supervised clustering algorithm [10] reviewed in Section 2.5.1. The user should specify pairwise constraints between instances. These constraints are then formulated in the objective function of k-means, Eq. (2.37), where violation is penalized with pre-defined costs. The user interaction consists of the following steps:

- All instances of the collection are displayed to the user as shown in Fig. 2.2. The user starts moving the instances on the screen. Each time a pair of instances is selected and moved. The screen distance between the instances is then computed. If the distance is greater than α, a *cannot-link* constraint is created. If the distance is less than β, a *must-link* constraint is created. The constrained clustering algorithm begins after the user has moved two instances.

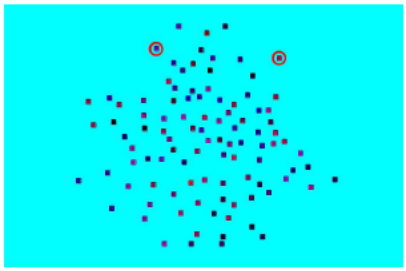

Figure 2.2: A snapshot of the interface implemented to elicit pairwise constraints between instances [77]. The user selects a pair of instances and moves them on the screen. The screen distance between the instances indicates whether a pairwise constraint should be generated.

- The clustering algorithm with the generated constraints is run.
- The screen is then updated based on the obtained clusters. The center of clusters are first mapped to the screen. All the instances are then mapped based on their distances to the centers.

The main contribution of the interface is that the user can easily generate the pairwise constraints by moving the instances on the screen. We believe that this is the easiest way to elicit pairwise constraints from users. However, it is not clear what kind of metadata is displayed on each instance in Fig. 2.2. It is necessary to show some information about the selected instances, for example their keyterms in case of document clustering.

iVisclustering is a visualization interface proposed in [44] to support document level and term level supervisions. The background clustering algorithm of this interface is the *LDA* model, reviewed in Section 2.3.3. Several views are embedded in the interface so as to provide a complete insight into a document collection. A snapshot of the interface is shown in Fig. 2.3. The interface is updated based on the clusters generated by *LDA*:

- Part A of the interface is a cluster relation view representing a hard clustering of documents. Each node of the graph is a document, which is connected to the other similar documents. Edge weights are computed based on the cosine similarity among document contents. Each color in this graph corresponds to a document cluster. Top keyterms of each cluster are shown on top of each cluster.

- Part B of the interface shows a hierarchy of the clusters as a folder tree. The root of the tree contains all the documents and each subtree presents a cluster. The user can merge two clusters by drag and dropping them in a same subtree. The user can also make a new subtree or delete them. This view lets the user change the number of document clusters that she prefers to generate.

- Part C of the interface shows the top keyterms of each cluster. Whenever the user clicks on a rectangle in this view, all the documents assigned to the respective cluster will be shown in Part G. The term weights in the respective document cluster is also shown in part E.

- Given the term weights in part E, the user can perform term level supervision. The user can increase or decrease the weight of terms in different clusters.

- Part G shows all the documents assigned to the selected cluster along with their contents. The top keyterms of the clusters are highlighted in the content.

- The Parallel Coordinate view in part D shows the membership of documents in the clusters. The horizontal axis presents the document clusters, and the vertical axis, which is scaled between zero and one presents the membership of documents in clusters. Each line in this view is a document. By selecting a line in this view, the user can easily figure out the membership of the document in different clusters.

- After user supervision, the *LDA* model is re-learned and the interface is updated based on the new clusters. Part F of the interface depicts the changes made in the clusters after user supervision. Each entry ij of the matrix consists of documents that were in cluster i before user supervision, and they are now in cluster j after user supervision. This view helps the user to find out which documents changed their clusters after supervision.

Although there is no novelty in the interface and the background clustering algorithm has already been proposed, this interface is the best visualization implemented for interactive text document clustering based on our knowledge. It easily lets the user perform document and term level supervision simultaneously. By changing the tree view of Part B, the user can also change the number of document clusters. We believe

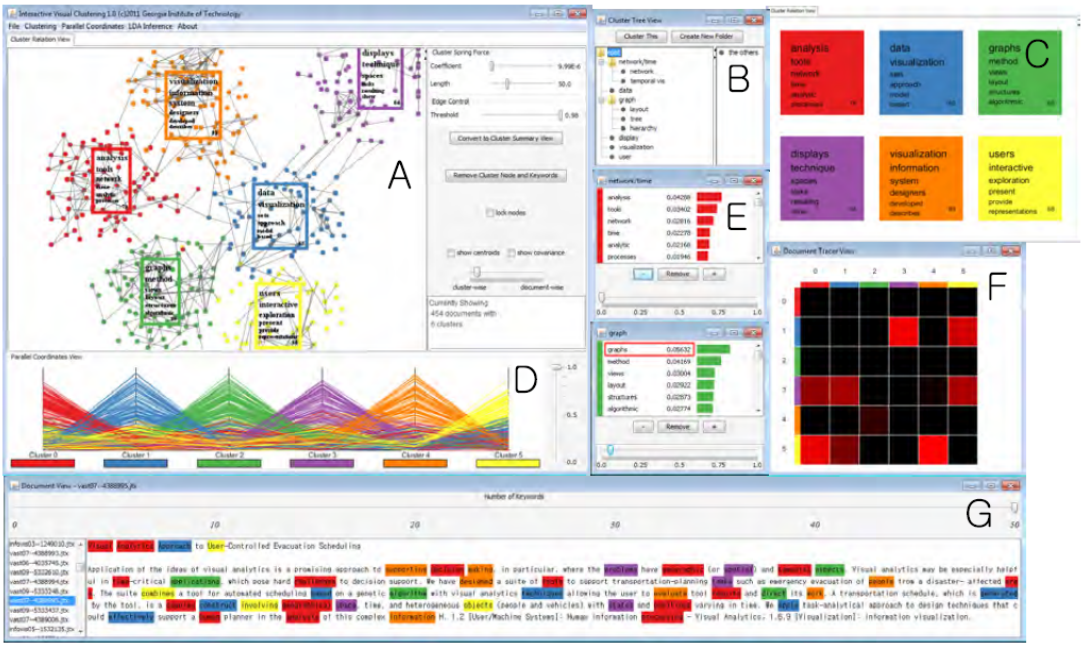

Figure 2.3: *iVisClustering*: A visual interface to support *LDA* topic modeling [44]

that the interface would be more useful if the keyterms of clusters were displayed in another way like in the form of term clouds.

The latent topics of the *LDA* model are visualized by using term clouds in [41, 35, 74]. The probability distribution of terms in each topic is used for this purpose. The user can explore the topics and gather information about their documents. However, no interactive clustering algorithm is proposed and the visualizations are only designed to display and explore the latent topics

A matrix view representation technique is used to display the topic-term distribution of *LDA* in *Termite* [24]. Rows of the matrix correspond to terms and columns correspond to topics. The term probabilities are then displayed as circle in this matrix as shown in Fig. 2.4. The user selects a topic and the interface visualizes its terms' frequencies along with the most representative documents. The interface is designed solely to display the latent topics and no interactive clustering algorithm is proposed.

A visualization interface to support semi-supervised document clustering algorithms is proposed in [55]. The interface elicits the user's preferences through both document and term level supervisions. A snapshot of the interface is shown in Fig. 2.5. The interface is divided into four view panels:

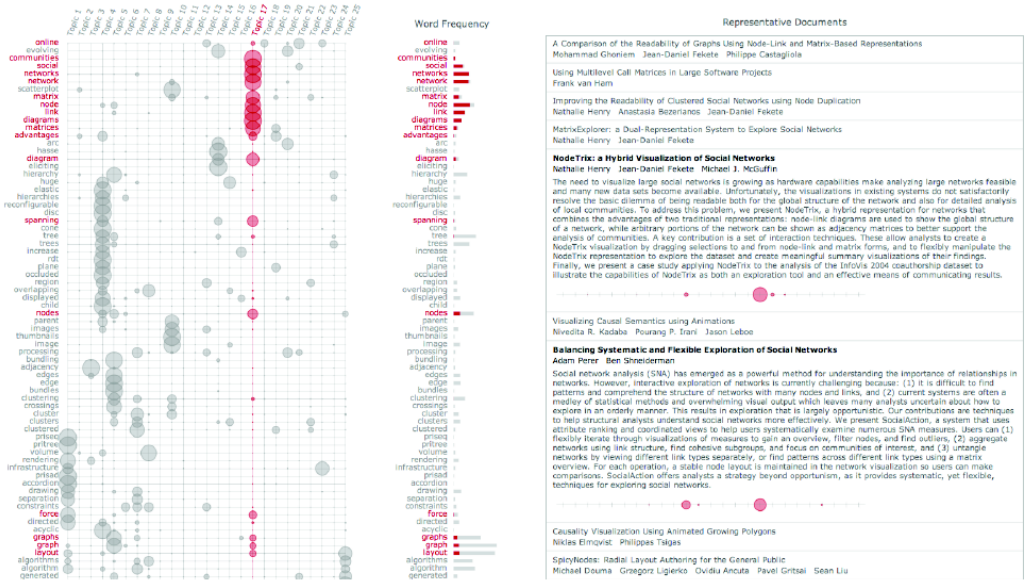

Figure 2.4: *Termite*: A visual interface to display topic-term distribution of the *LDA* model [24]

1. *Supervision Panel* on the top left side is a circle to show the labeled documents. Each sector of this circle is a document cluster where each yellow slice is a document. A sector named *New Cluster* provided in this circle to let the user create a new cluster. The user can drag a document from the top right panel and drop it into this sector to make a new cluster. The user can also merge two clusters by dragging a sector into another. A document can also be moved from one sector into another. There is also a sector named *Trash*, where the user puts outlier documents in. This panel helps the user to label documents in the document supervision phase of semi-supervised clustering algorithms.

2. *To be Labeled Document View* on the top right side of the interface displays all the documents, which are not assigned to any cluster yet. The panel shows the important keyterms of each document in red color along with its text content. The user is able to select a term and make it a keyterm or she can click on a keyterm and make it non-keyterm. By dragging the document into one of the sectors in the *Supervision Panel*, the user provides a seed document for the clustering algorithm.

3. The user can select a document from a cluster by clicking on a yellow line inside

a sector in the *Supervision Panel*. The interface then shows the content and the keyterms of the selected document in the *Labeled Document View* in the bottom right panel. The user can add or remove the keyterms of the selected document in this panel.

4. The user can select a cluster from the circle by clicking on a sector. The interface then shows the keyterms of the selected cluster in the *Cluster View* in the bottom left panel. The user can add or remove the keyterms of the selected cluster in this panel.

The strength of the interface is that the user can specify the keyterms during document labeling. The keyterms are then used to form a feature space for the clustering algorithm. Moreover, this visualization easily lets the user specify her desired number of clusters by dragging and dropping documents or clusters in the circle provided in the interface.

This visualization is not an iterative clustering interface. The interface is designed only to elicit the user's preferences and no interaction is provided ever after.

An interactive visualization to explore text document collections and sense making is proposed in [43]. Text mining techniques like document clustering, document summarization, and sentiment analysis are integrated in this visualization. The k-means algorithm based on cosine distance is used as the clustering algorithm. The user specifies the number of clusters and she may also choose initial seed documents. The output document clusters are labeled with three most-occurring terms in clusters as shown in Fig. 2.6. The user selects documents in this view and sees their contents and related metadata in the document view as shown in Fig. 2.7. This visualization is not designed as an iterative clustering interface and no interaction is provided after initializing k-means.

Word cloud visualization is equipped with natural language processing technique in *Word Cloud Explorer* [46]. The visualization is not designed for clustering text data but it is useful in extracting information from a single document. After a text document is loaded, tokenization, sentence segmentation, part-of-speech tagging, lemmatization, and name-entity recognition are performed in pre-processing steps. A word cloud is then created based on the frequencies of terms within the document as shown in Fig. 2.8. Information and filter panels in this visualization let the user

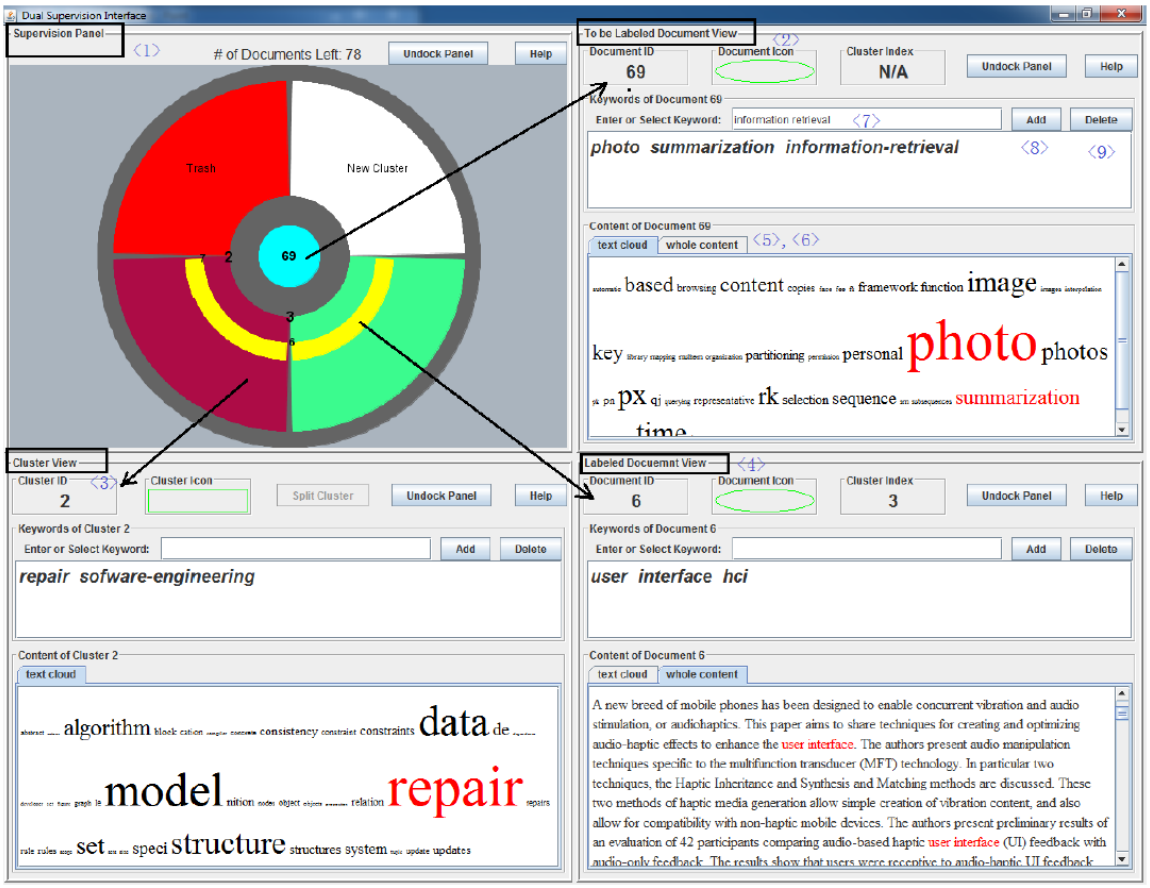

Figure 2.5: The interface proposed in [55] elicits the user's preferences through document and term level supervisions

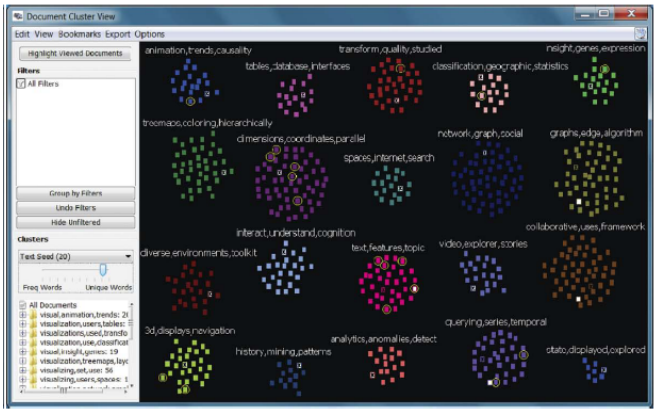

Figure 2.6: The document cluster view of the visualization proposed in [43]. Each cluster is assigned a color along with three most-occurred terms in its documents.

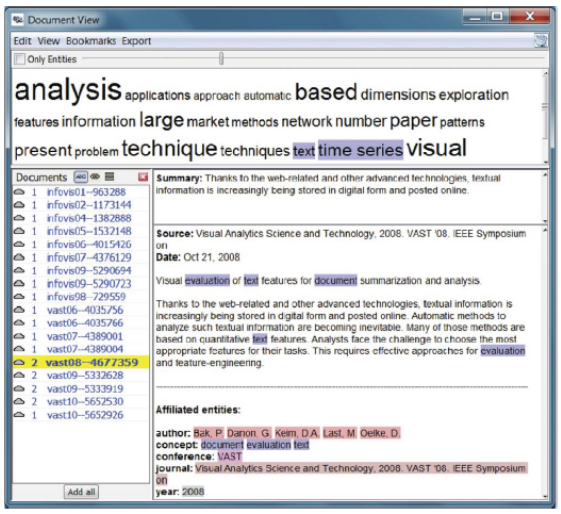

Figure 2.7: The document view of the visualization proposed in [43]. The selected documents are listed in this view. A one-sentence summary along with text content and metadata are presented for each document. A word cloud shows the most-occurred terms in the selected documents on the top side.

extract information from the document. The visualization consists of the following panels:

- Panel 'a': The term cloud is shown in this panel. The user can hover over a term to highlight its related terms. Two terms are related if they co-occur in the same sentence. Related terms are highlighted in yellow. The user can select terms by clicking on them. The selected terms are then added to panel 'b', term filter panel.

- Panel 'b': This panel includes filter terms. The word cloud only displays those terms that are related to the filter terms.

- Panel 'c': This is the search panel. The user can search for a term and add it to the term filter list. If the term is in the cloud, it will be highlighted too.

- Panel 'd': This panel is the term statistics panel. It shows the frequency of the focused term in the document, its frequency in the filtered sentences, the number of terms in the cloud, the number of terms in the document, and the number of sentences in which the term occurs.

- Panel 'e': This is the term info panel. It shows different forms of the focus term

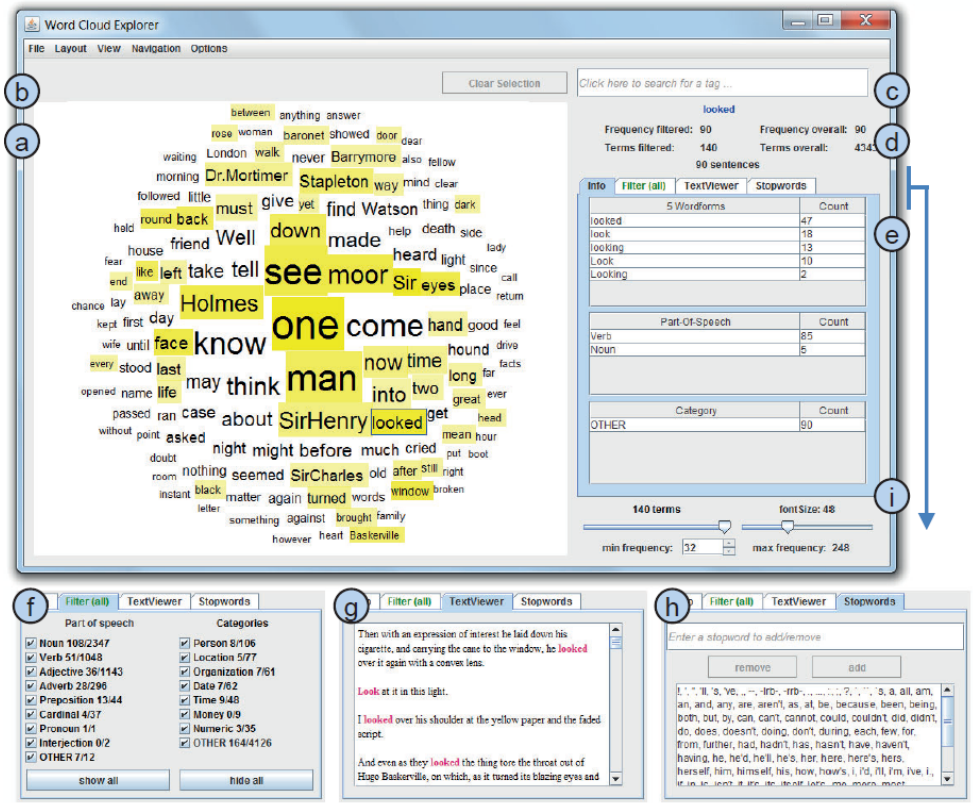

Figure 2.8: *Word Cloud Explorer* consists of a word cloud visualization along with filter and information panels [46].

in the document along with their frequencies. It also shows the part-of-speech tags and name-entity types of the term.

- Panel 'f': This is the linguistic filter panel. It enables the user to determine her desired part-of-speech tags or name-entity types. The terms within the term cloud are filtered based on the selected tags and types.

- Panel 'g': This is the text view panel. It shows all the sentences in which the focus term or its different forms occur.

- Panel 'h': This is the stopword list panel. The user can edit the list of stopwords.

- Panel 'i': This is the control panel. The user specifies the maximum number of terms in the cloud, the minimum frequencies of terms, and the maximum font size.

To the best of our knowledge, no one has used term clouds in interactive text document clustering. We have proposed an interactive interface for text document clustering in Chapter 8. The interface lets the user specify a document cluster by creating a term cloud. The keyterms inside the term cloud should reflect the topic of the document cluster for this purpose. The user can explore document contents or use the list of keyterms extracted from the collection to create term clouds in this interface. The number of term clouds created by the user specifies her desired number of document clusters.

Chapter 3

Experiment Setup

We review characteristics of the datasets used in experiments of this thesis in this chapter. We explain how the datasets are pre-processed using Natural Language Processing methods. The evaluation methods used to measure the quality of clusterings are also presented in this chapter.

3.1 Datasets

Eight document collections are used in our experiments so as to compare clustering algorithms. From these collections, fifteen datasets with different dimensionality and number of clusters are generated. We give a brief review along with a list of datasets generated from each collection in below:

1. *20Newsgroups*: This dataset consists of approximately 20000 news articles grouped into 20 different topics[1]. We removed all articles duplicated in multiple groups. We then created six datasets from this collection according to the experiments performed in [11, 54]. Each dataset is created by selecting a subset of the whole collection. The characteristics of the datasets are mentioned below and also summarized in Table 3.1. The *Reduced No. of Terms* column is number of terms after a pre-processing step. The second last column of the tables in this section show the percentage of zero values that exists in the datasets.

    - *News-sim3* includes all articles in three similar classes (topics) including *comp.graphics*, *comp.os.ms-windows.misc*, and *comp.windows.x*.
    - *News-rel3* includes all articles in three related topics including *talk-politics-misc*, *talk-politics-guns*, and *talk-politics-mideast*.
    - There are articles with seven topics in *News-multi7* including *alt.atheism*,

[1] `http://qwone.com/~jason/20Newsgroups/`

Table 3.1: Summary of the text datasets generated from *20Newsgroups* collection

Dataset Name	No. of Documents	No. of Terms	Reduced No. of Terms	No. of Classes	Sparsity Percentage	Stemmed
News-sim3	2924	20753	4697	3	99.7%	✔
News-rel3	2624	21659	4947	3	98.20%	✔
News-multi7	6632	33469	7006	7	99.09%	✔
News-multi10	9586	43620	8827	10	99.30%	✔
20ng-subset	7528	58814	12780	20	99.43%	✔
20ng-whole	18821	92587	18045	20	99.13%	✖

comp.sys.mac.hardware, *misc.forsale*, *rec.sport.hockey*, *sci.crypt*, *alt.politics.guns*, and *soc.religion.Christian*. All the articles in the mentioned topics are included in the dataset.

- *News-multi10* includes all articles with ten topics including *alt-atheism*, *comp-sys-mac-hardware*, *misc-forsale*, *rec-autos*, *rec-sport-hockey*, *sci-crypt*, *sci-med*, *sci-electronics*, *sci-space*, and *talk-politics-guns*.
- *20ng-subset* is a subset of the whole collection including all topics. Only 40 percent of documents of each topic are included in the dataset.
- *20ng-whole* includes all the documents in all topics.

2. *Reuters-21578*: The documents in the *Reuters-21578*[2] dataset appeared on the *Reuters* newswire in 1987 and labeled manually. We created two datasets by selecting a subset of topics in the collection. The characteristics of the datasets are mentioned below and also summarized in Table 3.2.

   - *Reuters8-subset* is a subset of *Reuters-21578* dataset consisting of documents in eight categories including *acq*, *crude*, *earn*, *grain*, *interest*, *money-fx*, *ship*, and *trade*. Only 70 percent of documents in each category is included.
   - *Reuters8-whole* is the same subset as *Reuters8-subset* but all the documents in the eight categories are included.

3. *SMART* data repository [3] contains abstracts of papers about medical, information retrieval, aerodynamics, and computing algorithms. Two datasets created

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`
[3] `ftp://ftp.cs.cornell.edu/pub/smart/`

Table 3.2: Summary of the text datasets generated from the *Reuters-21578* collection

Dataset Name	No. of Documents	No. of Terms	Reduced No. of Terms	No. of Classes	Sparsity Percentage	Stemmed
Reuters8-subset	5485	19367	4519	8	99.26%	✖
Reuters8-whole	7674	22750	5101	8	99.43%	✖

Table 3.3: Summary of the text datasets generated from *SMART* data repository

Dataset Name	No. of Documents	No. of Terms	Reduced No. of Terms	No. of Classes	Sparsity Percentage	Stemmed
Classic4	7095	41681	5099	4	99.5%	✖✔
Cacmcisi	4663	14409	2528	2	99.4%	✔

from this repository and used in our experiments. The characteristics of the datasets are summarized in Table 3.3.

- *Classic4* includes all the abstracts in medical, information retrieval, aerodynamics, and computing algorithms. We have stemmed and non-stemmed versions of this dataset in our experiments.
- *Cacmcisi* includes all the abstracts in computing algorithms and information retrieval only.

4. *LA-Times* consists of newspaper articles in *Los Angeles Times* with six topics including *Financial*, *Foreign*, *National*, *Metro*, *Sports*, and *Entertainment*. The dataset is created from the TREC-9 newspaper collection[4]. The characteristic of the dataset is summarized in Table 3.4.

5. *WebKB* consists of webpages collected from four computer science departments in *Cornell*, *Texas*, *Washington*, *Wisconsin* universities. The collection is created in the World Wide Knowledge Base (Web-Kb) project in the *CMU* text learning group[5]. The webpages are manually labeled into seven categories: student, faculty, staff, departments, course, project, other. We selected only four categories of student, faculty, course, and project which have more documents than the other categories. The characteristic of the dataset is summarized in Table 3.4.

[4] `http://trec.nist.gov/data/qa/t9_qadata.html`
[5] `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/`

Table 3.4: Summary of the datasets *LA-Times*, *WebKB*, *SMS*, *Cade*, and *Reviews*

Dataset Name	No. of Documents	No. of Terms	Reduced No. of Terms	No. of Classes	Sparsity Percentage	Stemmed
LA Times	6279	31472	6845	6	98.23%	✔
WebKB	4168	7675	2106	4	99.48%	✔
SMS	5479	7288	1676	2	99.26%	✖
Cade	13644	100105	16431	12	98.45%	✔
Reviews	4069	36746	7724	5	97.9%	✔

6. *SMS Spam Collection* consists of a public set of text messages labeled as *spam* or *non-spam*. The collection is used in experiments of the algorithm proposed in [26] and is publicly available[6]. The characteristic of the dataset is summarized in Table 3.4.

7. *Reviews* includes news articles about movies, food, restaurants, music, and radio gathered from *San Jose Mercury News*[7]. The characteristics of this dataset are summarized in Table 3.4.

8. *Cade* is gathered from the content of Brazilian web pages. The web pages are in the Portuguese language. A pre-processed version of the dataset[8] is generated by the *Universidade Federal de Minas Gerais* in Brazil and is used in [18]. The dataset is labeled by human experts.

Non-stemmed datasets are used in integrating Wikipedia concepts in document clustering in Chapter 6. This is mainly because Wikipedia articles are not stemmed and wikify methods require non-stemmed text as input.

3.2 Pre-processing

After extracting document contents (terms), we pre-process them in the following steps:

1. All the stop words are removed from the document contents.

[6] `http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/`
[7] `http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview`
[8] `http://web.ist.utl.pt/~acardoso/datasets/`

2. Porter stemming [93] is applied on the vocabulary of the collection. This process is not performed on datasets *SMS*, *Reuters8-subset*, *Reuters8-whole*, and *20ng-whole*. We also have stemmed and non-stemmed versions of *Classic4*.

3. Each document is then presented as a vector of terms based on the *BOW* model. The importance of terms in documents are measured using *TFIDF*.

4. The whole collection is then represented as a document-term matrix.

5. The dimensionality reduction step, described in Section 3.4, is performed on the matrix.

6. The effect of document length is finally reduced by using the $L2$ norm to normalize the length of document vectors to one.

3.3 Evaluation Measures

The clusterings obtained in the experiments of this thesis are evaluated based on the pre-defined class labels of documents. All text datasets in this thesis are labeled by human experts. This kind of evaluation demonstrates whether the clusterings are consistent with the human understanding of the document collections [56]. Besides, since we simulate user interactions in user-supervised algorithms of Chapter 7 based on the class labels, it is meaningful to use the class labels in performance evaluations.

On the other hand, no ground truth is usually available about class labels of documents in practice, as in our user study. In this case, the performance of clustering algorithms can be evaluated using intrinsic measures like cluster coherence in the form of within-cluster distances and between-clusters separations [86]. However, good performance based on intrinsic measures does not necessarily guarantee good performance in practice [79]. Instead of using intrinsic measures in this case, we use human judgments about document clusters in our user study. We asked the participants how consistent the topics of document clusters are with their understanding of the document collections.

Except for the user study, evaluation of document clusterings is performed in the following way in this thesis. A confusion matrix is formed after each clustering

process. Each element of this matrix shows the number of common documents between the corresponding cluster and class. The confusion matrix is used to compute two evaluation measures:

1. *Fmeasure* [72] is a commonly used measure in information retrieval. We used its balanced version which is the harmonic mean of precision and recall:

$$Fmeasure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.1}$$

   The maximum value of *Fmeasure* is desired.

2. Normalized Mutual Information (NMI) measures the amount of information we get about classes if we have clusters [79]. It has a maximum value of one when the clustering process recreates classes perfectly and it has a minimum of zero. In this study we used the following formula mentioned in [79]:

$$NMI = \frac{I(W,C)}{[H(W) + H(C)]/2} \tag{3.2}$$

$$I(W,C) = \sum_k \sum_j \frac{|w_k \cap c_j|}{N} \log \frac{N|w_k \cap c_j|}{|w_k||c_j|} \tag{3.3}$$

$$H(W) = -\sum_k \frac{|w_k|}{N} \log(\frac{|w_k|}{N}) \tag{3.4}$$

$$H(C) = -\sum_k \frac{|c_k|}{N} \log(\frac{|c_k|}{N}) \tag{3.5}$$

   where $W = \{w_1, w_2, .., w_k\}$ and $C = \{c_1, c_2, ..., c_k\}$ denote clusters and classes respectively, $|w_k \cap c_j|$ is the number of common instances between w_k and c_j, and N is the number of documents.

3.4 Dimensionality Reduction

Document representation based on the *BOW* model results in a high-dimensional document-term matrix. We use feature selection technique in order to reduce the

Table 3.5: The dimensionality of the datasets after dimensionality reduction steps

Dataset Name	No. of Documents	No. of Terms before dimensionality reduction	No. of Terms after dimensionality reduction
Reviews	4069	36746	7724
Classic4	7095	41681	5099
Cacmcisi	4663	14409	2528
News-sim3	2924	20753	4697
News-multi7	6632	33469	7006

dimensionality of this matrix. Among all terms of a collection, we keep only high-variance ones. The *Var-TFIDF* feature selection method describe in Section 2.1.2 is used for this purpose. Our dimensionality reduction approach has three steps:

1. *Var-TFIDF* assigns a score to each term using Eq. (2.5).

2. An average score is then computed for the collection using the following formula:

$$avgVar = \frac{1}{M}\sum_{j=1}^{M} \text{Var-TFIDF}_j \tag{3.6}$$

3. Those terms whose scores are larger than $avgVar$ are kept and the remaining terms are discarded from document contents.

To measure the effect of our dimensionality reduction method, we have conducted an experiment by using the *Naive Bayes* classifier implemented in *WEKA* [9]. We have run the classifier on the datasets of Table 3.5 before and after dimensionality reduction. On average 80% of the terms are removed from these datasets. The quality of classes obtained based on 10-fold cross validation method are measured by the *NMI* criterion and are reported in Table 3.6.

The dimensionality reduction method improved the performance of the classifier in this experiment except for the dataset *Reviews*, where the *NMI* values are the same. The largest improvements obtained for datasets *News-sim3* and *News-multi7*.

The conclusion of this section is that we can reduce the dimensionality of our document-term matrices, by removing those terms whose scores are smaller than the mean-score of all terms that exist in the collection, and still get the same or better

[9] `http://www.cs.waikato.ac.nz/ml/weka/`

Table 3.6: Average *NMI*s obtained by running *Naive Bayes classifier* on the datasets of Table 3.5 before and after dimensionality reduction.

Dataset Name	Before Removing Low-variance Terms	After Removing Low-variance Terms
Reviews	0.651	0.651
Classic4	0.918	0.958
Cacmcisi	0.903	0.944
News-sim3	0.292	0.539
News-multi7	0.668	0.833

classes in terms of quality. This is consistent with the experiments performed in [75], where around 90% percent of terms are removed with either improvement or no loss in clustering performance. This reduction can lead to a significant speed up in the clustering algorithms as well.

Chapter 4

Evolutionary Double Clustering

An instance of a clustering problem is described by a two-dimensional matrix. The rows of the matrix correspond to the objects to be clustered. The columns correspond to attributes. The entries of the matrix are numerical values that indicate the degree of presence of an attribute in the respective object. For a matrix with N rows and M columns, each object corresponds to a point in M-dimensional attribute space. The goal of clustering is to partition the objects into groups such that objects within a group are similar so that they have short pairwise distances from each other, while objects in different groups are separated by large distances.

If the number of attributes M is large, then basic clustering techniques, such as k-means, may fail to generate useful results. Meaningful groups of objects may be identified by the presence or absence of a few relevant attributes. The objects are close in the subspace corresponding to those attributes. However, that proximity fails to register if distances are computed in the full-dimensional space as it is hidden in the noise from the remaining attributes [69, 85].

The number of terms in a text corpus is typically large. Each topic of the corpus can be represented by some keyterms. The other terms are usually too general to discriminate topics and act as noisy attributes in document clustering. Therefore, it might be beneficial to first focus on term clusters and the keyterms that represent topics. There are two general approaches to use term clusters in document clustering: Double Clustering and Co-clustering.

Double clustering refers to algorithms that perform term clustering before document clustering [78, 102]. Simultaneous clustering of terms and documents is referred to *co-clustering* in the literature [7, 23, 34].

The number of document clusters is usually fixed to the original number of classes in experimental evaluations of double and co-clustering algorithms [7, 23, 34, 102]. However, finding an optimal number of term clusters is not a trivial task [34]. No

solution to find the optimal number of terms clusters is proposed in these algorithms. For this reason, the quality of clusters are analyzed for different numbers of term clusters using the ground truth [7, 23, 34, 102, 112]. This method of finding the number of term clusters raises the following questions:

1. What range of numbers should be tested in order to find an optimal number of term clusters?

2. How should one find a good value for this parameter when no ground truth is available?

3. Since the dimensionality of document-term matrices is often large, is it practical to run algorithms multiple times on each dataset to find the best value for this parameter?

We propose a new way of clustering that does not attempt to find the optimal number of term clusters. The proposed Feature Selective Double Clustering (*FSDC*) algorithm outperformed the competitive co-clusterers and double clusterers in our experiments.

On the other hand, term clusters are used in the co-clustering algorithms without any pre-analysis [7, 23, 34, 102, 112]. We used a MultiObjective Genetic Algorithm (MOGA) to find topic keyterms that exist in each term cluster. The evolutionary module distills term clusters by keeping only discriminative terms.

FSDC clusters documents in the following way. Terms that exist in the corpus are first clustered. The term clusters are then distilled to keep only discriminative terms. Representative documents are then extracted for each distilled term cluster. The representative documents are then used as seeds to cluster documents.

In detail, the main contributions of this work are mentioned below:

1. We propose a new way to extract topic keyterms. An evolutionary module is used to distill a term cluster by removing its non-discriminative terms.

2. Given the distilled term clusters, we then propose a method to find their representative documents, which are used later as seeds to cluster document. In our clustering algorithm, there is no need to evaluate the quality of clusters multiple times in order to find the optimal number of term clusters.

3. We finally propose a user-supervised version of *FSDC*. The keyterms of term clusters are extracted and shown to the users. The users are then asked to interactively label the keyterms. The labeled keyterms are then fed into the term clusterer to re-cluster terms. The user supervision is repeated for a few iterations until the term clusters match the user's preferences.

The remainder of this chapter is organized as follows. Section 4.1 provide a background about the proposed evolutionary algorithm. Section 4.2 explains the algorithm in detail. Experimental results on some real text datasets are shown in Section 4.3. Section 4.4 explains the user-supervised version of the algorithm and experimental results. Section 4.5 presents conclusions and future work. It is better to mention that the terms "representative" and "seed" might be used interchangeably in the rest of this report.

4.1 Background

The fuzzy c-means algorithm [13] is used in this chapter for term clustering. We review this algorithm in the first part of this section along with its benefit over k-means. We then explain how a probabilistic cluster assignment can be generated for documents in partitional clustering algorithm like k-means. In the second part of this section, we propose a new method to extract seed documents of each term cluster. We explain how zero values in document-term matrices are used for this purpose.

4.1.1 Fuzzy c-means

In classical set theory, each element either belongs or does not belong to a set. The set is called crisp and the membership value of an element is either zero or one. On the other hand, there is fuzzy set theory. In fuzzy set theory, each element has a degree of membership in a fuzzy set. The set is called fuzzy and the membership value of an element is a value between zero and one.

The idea behind fuzzy c-means is that an instance might be related to multiple clusters with different degrees of membership. In each iteration, the membership of an instance is computed based on its distances to the cluster centers. The degrees of membership are subsequently used in computing the cluster centers.

Instances around a center have more contribution than farther ones in computing the center. An instance between two centers have equal contribution in both centers. This is the main advantage of fuzzy c-means over k-means, where even outliers have the same contribution as close instances in computing centers.

Let $T = \{t_1, t_2, ..., t_M\}$ denote the data set and $t_i = \{t_{i_1}, t_{i_2}, ..., t_{i_N}\}$ is an instance presented in an N-dimensional space. Fuzzy c-means generate k clusters $\{TC_p\}_{p=1}^{k}$ such that the following objective function is locally minimized [13]:

$$F_z = \sum_{i=1}^{M} \sum_{p=1}^{k} u_{ip}^{z} dist^2(t_i, \mu_p) \tag{4.1}$$

where z is a real number larger than 1, u_{ip} is the degree of membership of t_i in TC_p, and μ_p is the centroid of TC_p. The matrix $U = \{u_{ip}\}_{M \times k}$ is the membership matrix of the algorithm with the following assumption:

$$\sum_{p=1}^{k} u_{ip} = 1 \quad \text{for} \quad \forall i = 1, 2, ..., M \tag{4.2}$$

where a binary U converts fuzzy c-means to the k-means algorithm.

During each iteration of fuzzy c-means, the membership matrix U and the centroids $\{\mu_p\}$ are updated using the following formula:

$$u_{ip} = \sum_{j=1}^{k} \left(\frac{dist(t_i, \mu_p)}{dist(t_i, \mu_j)} \right)^{2/(1-z)} \tag{4.3}$$

$$\mu_p = \frac{\sum_{i=1}^{M} u_{ip}^{z} t_i}{\sum_{i=1}^{M} u_{ip}^{z}} \tag{4.4}$$

where those instances with higher membership u_{ip} have more contribution than outliers or far instances in μ_p. The steps of fuzzy c-means is shown in Algorithm 1.

The final matrix U is fed to a defuzzification module in order to generate a soft or hard partitioning. There are many method for defuzzification [13]. An instance can be assigned to the cluster with the largest membership value so as to generate a hard partitioning. To generate a soft partitioning, the instance can be assigned to multiple clusters if its membership values are larger than a threshold.

The same membership matrix can be generated from the output of a partitional clustering algorithm, like k-means. Given the clusters $\{TC_p\}_{p=1}^{k}$, we compute the

Algorithm 1 Fuzzy c-means clustering algorithm

Input: *a document-term matrix* M_{DT}, $k > 1$, $z > 1$

Output: *membership matrix* U

1: *Initialize the membership matrix* U *randomly.*

2: *Compute the cluster centroids* μ_p *using:*

$$\mu_p = \frac{\sum_{i=1}^{M} u_{ip}^z t_i}{\sum_{i=1}^{M} u_{ip}^z}$$

3: *Update the membership matrix* U *using:*

$$u_{ip} = \sum_{j=1}^{k} \left(\frac{dist(t_i, \mu_p)}{dist(t_i, \mu_j)} \right)^{2/(1-z)}$$

4: *If* $\| U_s - U_{s-1} \| < \delta$ *or the maximum number of iterations is reached stop, otherwise go to step 2.*

probability of cluster TC_p given t_i using the following formula:

$$P(TC_p|t_i) = \frac{sim(t_i, \mu_p)}{\sum_{j=1}^{k} sim(t_i, \mu_j)} \tag{4.5}$$

where μ_p is the center of cluster TC_p, and $\sum_{p=1}^{k} P(TC_p|t_i) = 1$. The $P(TC_p|t_i)$ values can be used to generate a soft partitioning of the terms or to show how related a term is to the term clusters. A similar approach can be used for documents in document clusters.

4.1.2 Extracting Representative (Seed) Documents

In any partitional clustering algorithm like the one we have proposed in Chapter 5, finding a good set of seed documents is crucial. We conducted an experiment to see whether zero values in a document-term matrix can be used for this purpose. It is worthy to mention that on average more than 95 percent of entries of the document-term matrices in our experiments are zero.

The experiment is performed in the following way. We created a dataset by selecting documents from four classes of the *20Newsgroups* collection. Given the class labels of the documents, we used the χ^2 statistic to find the top 20 keyterms of each class. We then create a document-term matrix given the keyterms. Documents of the same class sit next to each other in this matrix. The dimensionality of this matrix is $N_s \times 80$, where $N_s = 400$ is the number of documents, 100 documents from each class.

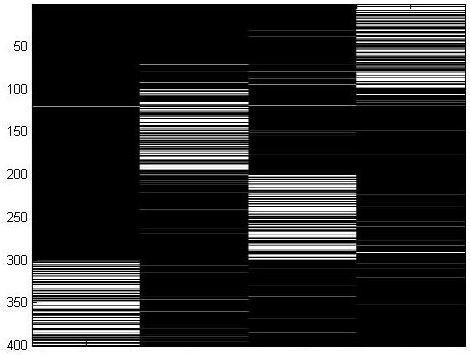

Figure 4.1: The document-centroid matrix is mapped to a gray-scale image. The block structure of the representative documents (light areas) is clear in this image.

Given the document-term matrix, we created another matrix with dimensionality of $N_s \times 4$, where each column of the new matrix is the column average of 20 respective keyterms. Each column of this matrix is like a centroid of the respective keyterms. We call this matrix, document-centroid matrix in the rest of this section.

A gray-scale image is then generated based on the *TFIDF* values of the document-centroid matrix. The maximum *TFIDF* value in this matrix is mapped to a white pixel ($g = 255$), and the minimum value is mapped to a black pixel ($g = 0$). The other values are scaled to a gray pixel ($g \in [1 \quad 255]$). The obtained image is shown in Fig. 4.1, where the documents of each class are reordered to sit next to each other.

Each column of the image consists of a dark area along with a light area. Each light area corresponds to the representative documents of a class in which the respective keyterms have larger *TFIDF* values. The dark areas corresponds to the non-representative documents in which the respective keyterms have near-zero *TFIDF* values. By grouping *TFIDF* values of each column into two clusters, one can extract the representative documents. One cluster includes instances with near-zero values and the other one includes instances with larger values.

We extract seed documents in our partitional clustering algorithms in the following way:

1. Given the term clusters, we apply the k-means algorithm with $k = 2$ on each term centroid of term clusters to partition its elements into two clusters.

2. We then find the clusters with the larger values to extract the representative documents.

It is noteworthy to mention that a representative document might be extracted from multiple term clusters. It means that the document is related to multiple topics in the corpus.

At first glance, having zero values in the document-term matrices seem troublesome. However, we have shown how to utilize these zero values so as to extract seed documents provided that there is a way to extract the keyterms. We have used this method in our proposed algorithms in this thesis.

4.2 Methodology

The proposed evolutionary algorithm (*FSDC*) is based on the idea that before finding document clusters, it is better to focus on term clusters and try to find the keyterms representing their topics. Having the keyterms, representative documents for each term cluster are extracted using the method proposed in Section 4.1.2. The representative documents are subsequently used as seeds to cluster documents. Document representation is based on the *BOW* model in this algorithm. The structure of *FSDC* and its main steps are depicted in Fig. 4.2 and Algorithm 2, respectively. *FSDC* consists of the following phases:

1. **Term clustering**: In this phase, fuzzy c-means is used to cluster terms (columns of the document-term matrix). Fuzzy c-means groups the terms into k term clusters. The value of k is user-defined. We suppose that the number of document clusters and term clusters is the same. The output of this phase is thus k term clusters.

2. **Topic keyterm selection**: The number of keyterms needed to present a topic depends on the clustering algorithm and the dataset [20]. This value is not known in advance. In this phase, a multiobjective genetic algorithm is used to select the keyterms of each term cluster. The *MOGA* module are applied on the term clusters independently. The output of each module are several distilled clusters (corresponding to non-dominated solutions on the Pareto front) with different numbers of keyterms inside them. These distilled term clusters are the input of Phase 3. The goal of Phase 3 is to select the best distilled term clusters.

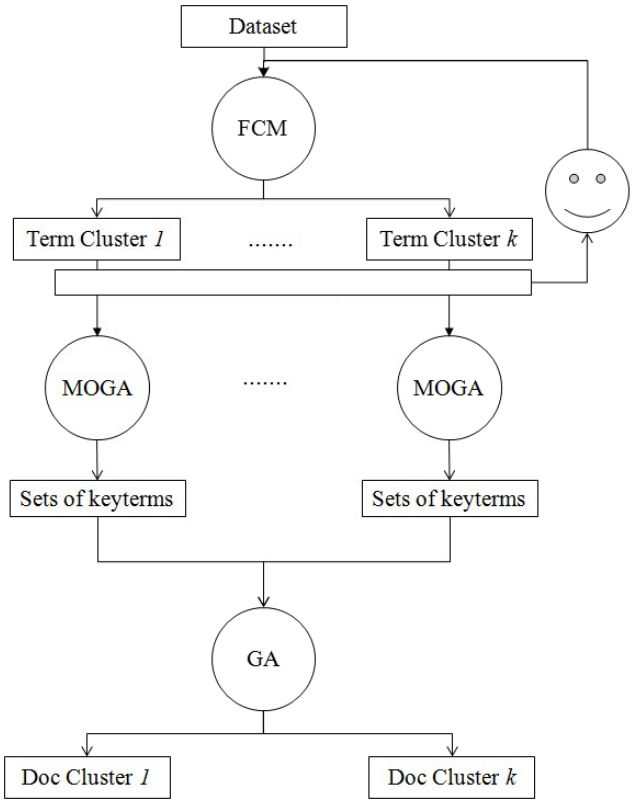

Figure 4.2: The main steps of (user-supervised) *FSDC*. Fuzzy c-means is used for term clustering. The *MOGA* modules distills term cluster to remove non-discriminative terms. Distilled term clusters are then used to extract seed documents and cluster documents.

3. **Document clustering**: Given the distilled term clusters, their representative documents are extracted in this phase. A document centroid is then computed for each distilled term cluster. A single-objective genetic algorithm (GA) finally chooses the best k document centroids to cluster documents.

4.2.1 Term Clustering

Having similar topics is a common case in text clustering. Similar topics share common terms and each term usually belongs to multiple topics with different degrees of relevance. This is a key issue in term clustering.

The integration of the fuzzy paradigm with the simplicity and efficiency of k-means, make fuzzy c-means a good candidate for term clustering. Given a document-term matrix, we apply fuzzy c-means on the term vectors to generate k term clusters.

Algorithm 2 Feature Selective Double Clustering (*FSDC*)

Input: *a document-term matrix* M_{DT}, k

Output: k *document clusters* $\{W_p\}_{p=1}^{k}$

1: *use fuzzy c-means, Algorithm 1, to generate k term clusters* $\{TC_p\}_{p=1}^{k}$
2: **for** *each term cluster* TC_p **do**
3: *apply the MOGA to prune non-discriminative terms*
4: **for** *each non-dominated solution (distilled term cluster)* **do**
5: *extract the representative documents* rep_s
6: *compute a document centroid over its representative documents* dc_s
7: **end for**
8: **end for**
9: *apply the GA to identify the best document centroids dc*
10: *cluster documents using the selected document centroids:*
11: ***for*** *each document in the dataset* d_i ***do***
12: *measure its similarities to the document centroids*
13: *assign the document to each document centroid with a membership value:*

$$P(W_p|d_i) = \frac{sim(d_i, dc_p)}{\sum_{j=1}^{k} sim(d_i, dc_j)}$$

14: ***end for***

After fuzzy clustering is done, we use the maximum method to defuzzify the obtained membership matrix [13]. The maximum method assigns each term to the cluster with the largest membership value. The output of this method is thus a hard term clustering. This phase of the algorithm has time complexity of O(NMK^2I), where I is the maximum number of iterations of fuzzy c-means [42].

4.2.2 Topic Keyterm Selection

The input of this phase are k term clusters. We propose a multiobjective genetic algorithm in this section so as to distill term clusters and get rid of general terms. The algorithm is inspired by the Genetic algorithms proposed in [88, 109]. We explain the components of this genetic algorithm in the following.

- An integer representation is used for individuals. Each individual includes a subset of terms that exist in a term cluster in the form of:

$$ind_i = (t_1, t_2, ..., t_{T_i}) \tag{4.6}$$

  where t_j is the column number of a term vector in the document-term matrix, and T_i is the number of terms used in the i^{th} individual.

  Since it is not clear how many keyterms should be used to represent a topic, a variable-length integer representation is considered for individuals. This representation lets the genetic algorithm find the number of keyterms. However, variable-length representations suffer from bloating, the uncontrolled growth of the length of individuals [76]. We thus restricted the length of individuals to a value between 20 and $\frac{1}{4}$ size_of(term cluster).

- The goodness of an individual depends on the goodness of the subspace spanned by its terms. Finding a good subspace can be defined as a multiobjective task using the following criteria:

  1. The number of similar documents in the subspace.
  2. The degree of similarity between the documents and the size of subspace in which similarities are measured.

The best subspace has the minimum number of terms while characterizing the maximum number of similar documents. We call these similar documents, representative documents in the rest of this chapter. Representative documents are extracted based on the method proposed in Section 4.1.2 as given below:

1. A term-centroid is computed for each individual. A term-centroid is the average of the column vectors corresponding to the terms included in an individual. A term-centroid is a vector with dimensionality equal to the number of documents.
2. The k-means algorithm with $k = 2$ is applied on each term-centroid (in a one-dimensional space) to partition its elements into two clusters. One cluster includes elements with near-zero values and the other one includes elements with larger values. The cluster with the larger values includes the representative documents.

After finding the representative documents of each individual, two objective function values are computed for fitness assignment:

1. *Number of representative documents*: the number of representative documents of an individual. This objective should be maximized since we use the representative documents as seeds to cluster all documents later. Having a larger number of these documents results in better document clusters.
2. *Distortion*: The representative documents of an individual should be similar to each other. Distortion or intra-cluster variation is a common criterion used to evaluate the quality of clusterings [78, 109]. The distortion of the i^{th} individual is computed as:

$$\text{Distortion}(ind_i) = \frac{1}{T_i} \sum_{d_j \in rep_i} \| d_j - dc_i \|^2_{SP_i} \tag{4.7}$$

where rep_i is the set of representative documents associated with the i^{th} individual, dc_i is the document-centroid computed over the documents of rep_i, and SP_i is the feature space spanned by the terms of the i^{th} individual.

The distances are normalized by the size of subspace T_i. This objective function should be minimized.

- Individuals are initialized by using the high-variance terms (Eq. 2.5). The initial size of individuals is chosen randomly. Parent selection is performed by tournament selection[83] and population size is fixed to 50. The probability of uniform crossover [50] is 0.7 and we perform the following mutations on all offspring:

  1. Add-term: this operator increases the length of an individual by 1 and adds a new randomly selected term.
  2. Remove-term: this operator decreases the length of an individual by 1 and removes a randomly selected term.
  3. Replace-term: this operator randomly replaces a term by a new term.

The output of each *MOGA* module is a set of distilled term clusters with different numbers of terms. The goal of the next phase, Document Clustering, is to determine the best term clusters from these distilled term clusters.

A sample output of this phase is shown in Fig. 4.3. In this figure, terms are already clustered into two term clusters. The topic of the first term cluster is *Electrical* and the topic of the second one is *Car*. Some terms in this example are too general to discriminate a topic like Canada, September, Please, and Number. We applied the MOGA on each term cluster individually to remove these general terms.

FSDC is independent of the multiobjective optimization algorithm and any kind of multiobjective method can be used. In this study, we used a Matlab implementation[1] of *NSGA-II* [31]. It is also noteworthy to mention that this phase of *FSDC* can be run independently on each term cluster in parallel as shown in Fig. 4.2.

This phase has time complexity of $O(PNMKI+P^2KI)$, where P is the population size. The time complexity of *NSGA-II* in forming Pareto fronts in case of two objective functions is $O(P^2)$ [31].

[1] http://www.mathworks.com/matlabcentral/fileexchange/10429-nsga-ii-a-multi-objective

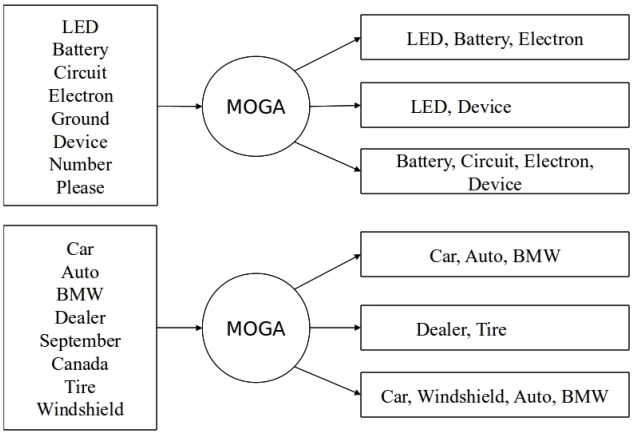

Figure 4.3: A sample output of Phase 2 (Topic keyterm selection). Two term clusters are fed into the MOGA so as to prune non-discriminative terms. Each output is a distilled term cluster including some keyterms.

4.2.3 Document Clustering

Multiple distilled term clusters are obtained for each term cluster in Phase 2. In this phase, we want to identify the best distilled term clusters.

Each distilled term cluster characterizes a set of representative documents. We compute a document centroid over the representative documents of each distilled term cluster. A document centroid is the row average of document vectors corresponding to the representative documents. Since *MOGA* produces multiple distilled term clusters, there are multiple document centroids for each term cluster. We need to choose only k document centroids to cluster all documents.

Suppose that the number of document centroids of the i^{th} term cluster is tk_i. To find the best document centroids, we need to examine $tk_1 \times tk_2 \times ... \times tk_k$ cases. This is very time-consuming where k or the number of distilled term clusters is large.

To search for the best document centroids, a single-objective *GA* is used in this phase. In this *GA*, the length of individuals is fixed to k. Each entry of an individual is dedicated to one term cluster. The value of an entry indicates one of the document centroids of the respective distilled term cluster.

For fitness assignment, all documents are assigned to the nearest document centroids indicated in an individual to generate k clusters $\{C_i\}_{i=1}^{k}$. We then compute the

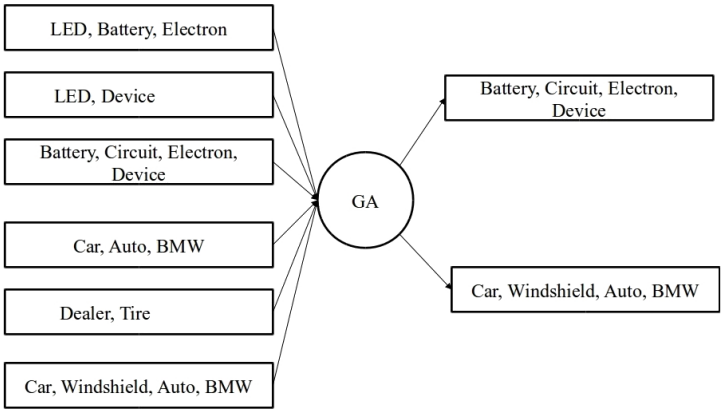

Figure 4.4: A sample output of Phase 3 (Document clustering). The *GA* identified the best distilled term clusters, which are used to cluster documents.

Separation (inter-cluster distances) of the clusters as the fitness value of the individual:

$$\text{Separation} = \sum_{\mu_i,\mu_j} \parallel \mu_i - \mu_j \parallel^2 \tag{4.8}$$

where μ_i is the centroid of cluster C_i. After a predefined number of iterations, the best individual in the population generates k document clusters.

Parent selection is performed by tournament selection and no crossover is considered. The population size is fixed to 25. The only genetic operator is a replacement mutator which randomly replaces one document centroid with another one. The number of iterations is 50 in this algorithm.

A sample output of this phase is shown in Fig. 4.4. As shown in this figure, from all distilled term clusters, the third distilled clusters of term clusters are chosen for document clustering.

This phase has time complexity of $\mathrm{O}(PNMKI + PMK^2I)$. If we assume the same number I of iterations for Genetic algorithms and fuzzy c-means, the time complexity of *FSDC* is $\mathrm{O}(NMK^2I + PNMKI + PMK^2I + P^2KI)$.

4.3 Experimental Results

This section provides performance comparisons of some co-clustering and direct clustering algorithms. We show the benefits of *FSDC* as compared to seven co-clustering algorithms introduced in Section 2.3.1:

- Information-Theoretic Co-clustering (*ITCC*) [34],
- Euclidean Co-clustering (*ECC*) [23],
- Minimum Sum-squared Residue Co-clustering (*MSRCC*) [23],
- Square Euclidean co-clustering and I-divergence co-clustering with bases $C2$ and $C5$ (*SECC2*, *SECC5*, *IdivCC2*, *IdivCC5*) [7],

We did not include the clusterer of [102] in our experiments since it is shown that *ITCC* outperforms the clusterer on some subsets of the *20-Newsgroups* dataset [34]. We also performed some experiments using the following direct (without term clustering) clustering algorithms:

- Standard k-means,
- Fuzzy c-means (*FCM*) [13],
- Four greedy direct clustering algorithms described in Section 2.3.2 called:
  - Bisecting I2 (*BI2*),
  - Bisecting H2 (*BH2*),
  - Direct I2 (*DI2*),
  - Direct H2 (*DH2*)
- The unsupervised version of the clustering algorithm proposed in [78] (*UVFCM*).

4.3.1 Evaluation Measures and Datasets

Given the class labels of documents, we measure the quality of clusterings using *Fmeasure* (Eq. 3.1) and *NMI* (Eq. 3.2). We used five text datasets whose characteristics are summarized in Table 4.1.

We use Euclidean distance to cluster document vectors since their length are normalized to one and Cosine similarity to cluster term vectors.

Table 4.1: Summary of the text datasets used in our experiments

Dataset Name	No. of Documents	Reduced No. of Terms	No. of Classes	Sparsity
Reviews	4069	7724	5	97.9%
Classic4	7095	5099	4	99.5%
Cacmcisi	4663	2528	2	99.4%
News-sim3	2924	4697	3	99.7%
News-multi7	6632	7006	7	99.8%

Table 4.2: Characteristics of the competitors used in our experiments

Clustering Algorithm	Term Clustering	Number of Document Cluster	Number of Term Cluster
FSDC	✔	k	k
UVFCM	✔	k	k
ITCC	✔	k	*variable*
ECC	✔	k	*variable*
MSRCC	✔	k	*variable*
SECC2	✔	k	*variable*
SECC5	✔	k	*variable*
IdivCC2	✔	k	*variable*
IdivCC5	✔	k	*variable*
BI2	✖	k	✖
BH2	✖	k	✖
DI2	✖	k	✖
DH2	✖	k	✖
k-means	✖	k	✖
FCM	✖	k	✖

4.3.2 Results and Discussion

We have evaluated the performance of 15 clustering algorithms on five datasets using two evaluation measures. Nine algorithms including *FSDC*, *UVFCM*, and seven co-clusterers (*ITCC*, *ECC*, *MSRCC*, *SECC2*, *SECC5*, *IdivCC2*, *IdivCC5*) use term clustering to generate document clusters as shown in Table 4.2. The number of term clusters for *FSDC* and *UVFCM* is the same as the number of document clusters.

For the co-clustering algorithms, we used different numbers of term clusters. For *ITCC*, *ECC*, and *MSRCC* the number of term clusters are $1, 2, 4, 8, ..., 128$ as suggested in [34]. These numbers are $5, 10, 15, ..., 50$ for *SECC2*, *SECC*, *IdivCC2*, and *IdivCC5* as in the experiments performed in [7].

Table 4.3: Standard deviations of *NMI*s obtained for the competitors in 20 runs

	Cacmcisi	Reviews	Classic4	News-sim3	News-multi7
FSDC	2×10^{-5}	5×10^{-4}	0.005	0.003	2×10^{-4}
UVFCM	0.162	0.003	0.004	0.05	0.09
ITCC	0.0063	0.0228	0.0423	0.0017	0.0246
ECC	0.0060	0.0197	0.0656	0.0057	0.0225
MSRCC	0.0098	0.0139	0.0245	0.0079	0.0061
SECC2	0.0626	0.0740	0.0168	0.0179	0.0530
SECC5	0.0022	0.0656	0.0121	0.0604	0.0346
IdivCC2	0.1480	0.0331	0.0623	0.0093	0.0351
IdivCC5	0.0337	0.0610	0.0531	0.0228	0.0331

We first compared *FSDC* to the co-clustering algorithms. For each dataset, we ran the experiments in the following way. Each co-clusterer is run 20 times for each number of term clusters and the average *Fmeasure* and *NMI* of these 20 runs are computed. The other algorithms are also run 20 times and the average *Fmeasure* and *NMI* are shown in Fig. 4.5 to Fig. 4.9. The number of term clusters in *FSDC* and *UVFCM* is fixed and their average results are thus shown as horizontal lines in the plots. The standard deviations of *NMI*s are also shown in Table 4.3. The standard deviations of the co-clusterers regarding the best number of term clusters are computed for this purpose.

Our experimental results reveal that *FSDC* outperformed the competitors on all the datasets. This is more evident on *Classic4*, and *News-sim3*. It is noteworthy to mention that *News-sim3* is the most difficult dataset to cluster in our experiments since it includes three similar topics. *FSDC* has also the lowest standard deviations in most cases as shown in Table 4.3. This observation confirms that the number of term clusters being selected as equal to the number of document clusters is a rational assumption and no further attempts for finding this number is needed for *FSDC* in this experiment. Documents in the datasets of this experiment are single-labeled. This assumption is thus rational in this case since each document is related to only one topic. However, further analysis is required in order to evaluate *FSDC* in case of multi-labeled documents.

An important observation in our experiments is that the outputs of the co-clusterers are sometimes sensitive to the number of term clusters. It is quite noticeable for datasets *Reviews* and *Classic4*. Even though it is suggested in [34, 102] to increase the number of term clusters in order to get better results, it did not happen on some

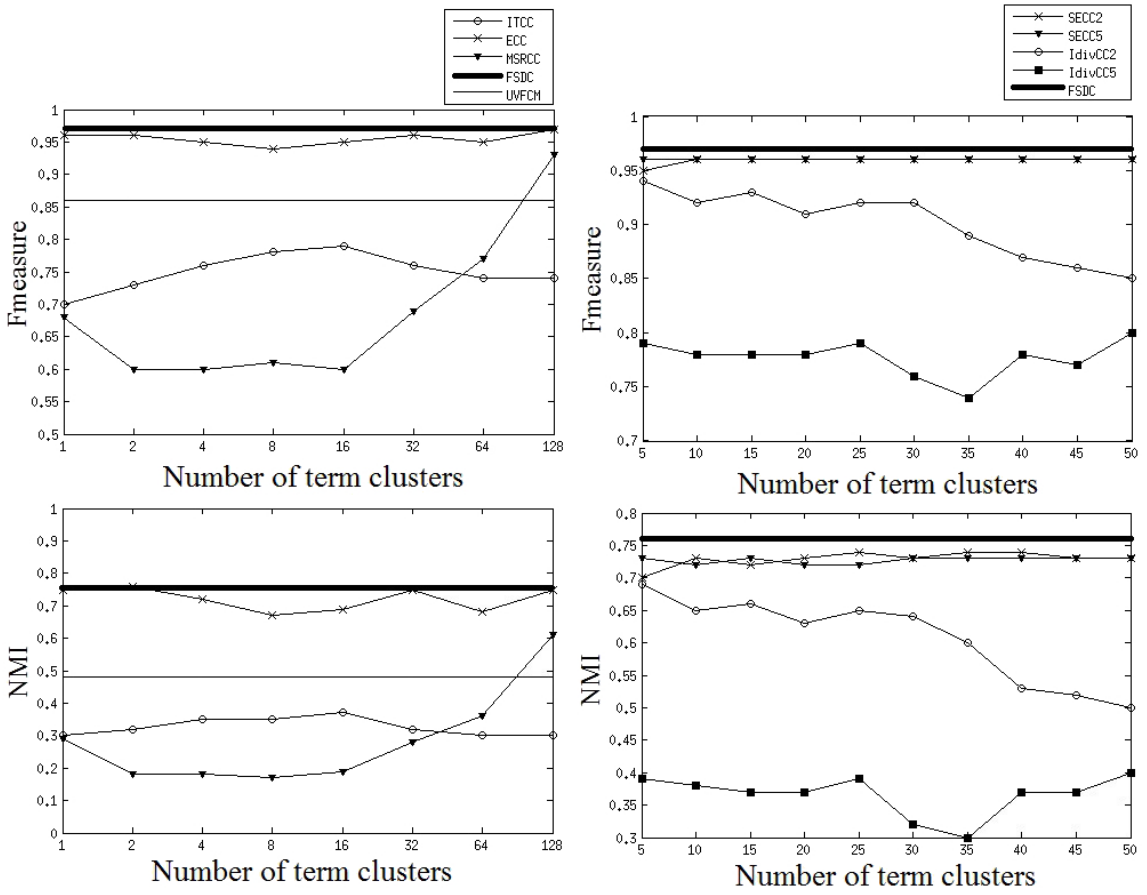

Figure 4.5: The quality of clusters for *FSDC* and the co-clusterers on Cacmcisi. *FSDC* outperformed the co-clusterers. The algorithms *ECC*, *SECC2* and *SECC5* could generate comparable results. The quality of clusters of *IdivCC2* decreases as the number of term clusters increases.

datasets in our experiments.

We have also compared *FSDC* to the direct clustering algorithms. These algorithm are run 20 times and the average of these 20 runs are summarized in Tables 4.4 and 4.5. Each entry in these tables shows the quality of obtained clusters on average along with the standard deviation. The standard deviation of the co-clustering algorithms shows the variation in the quality of clusters with respect to the best numbers of term clusters. For the other algorithms, the standard deviation shows the variation that exists in the quality of clusterings in 20 runs. The average running times of these algorithms in this experiment are shown in Table 4.6.

The quality of clusterings obtained based on the *Cacmcisi* dataset show that *FSDC*

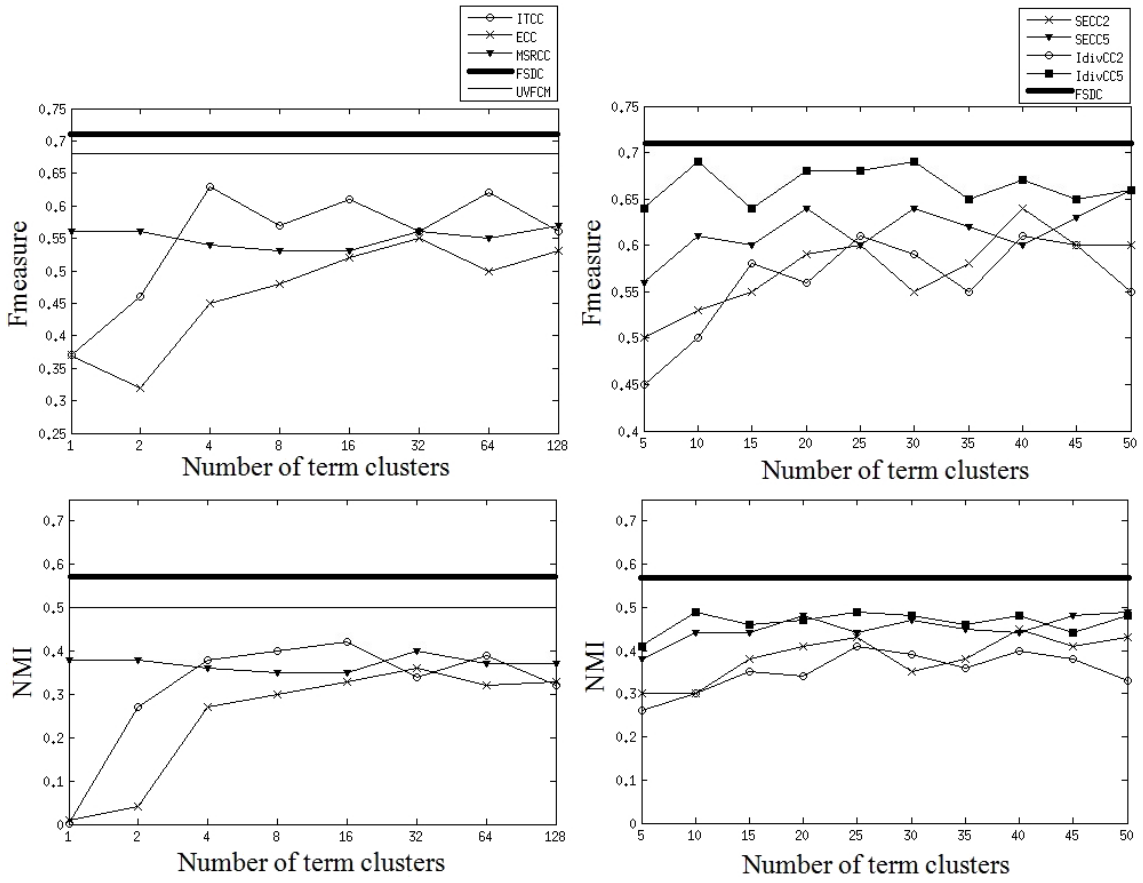

Figure 4.6: The quality of clusters for *FSDC* and the co-clusterers on Reviews. *FSDC* is the best clustering algorithm. The outputs of the co-clusterers (*SECC2*, *SECC5*, *IdivCC2*, *IdivCC5*, *ITCC*) are sensitive to the number of term clusters, based on *Fmeasure*.

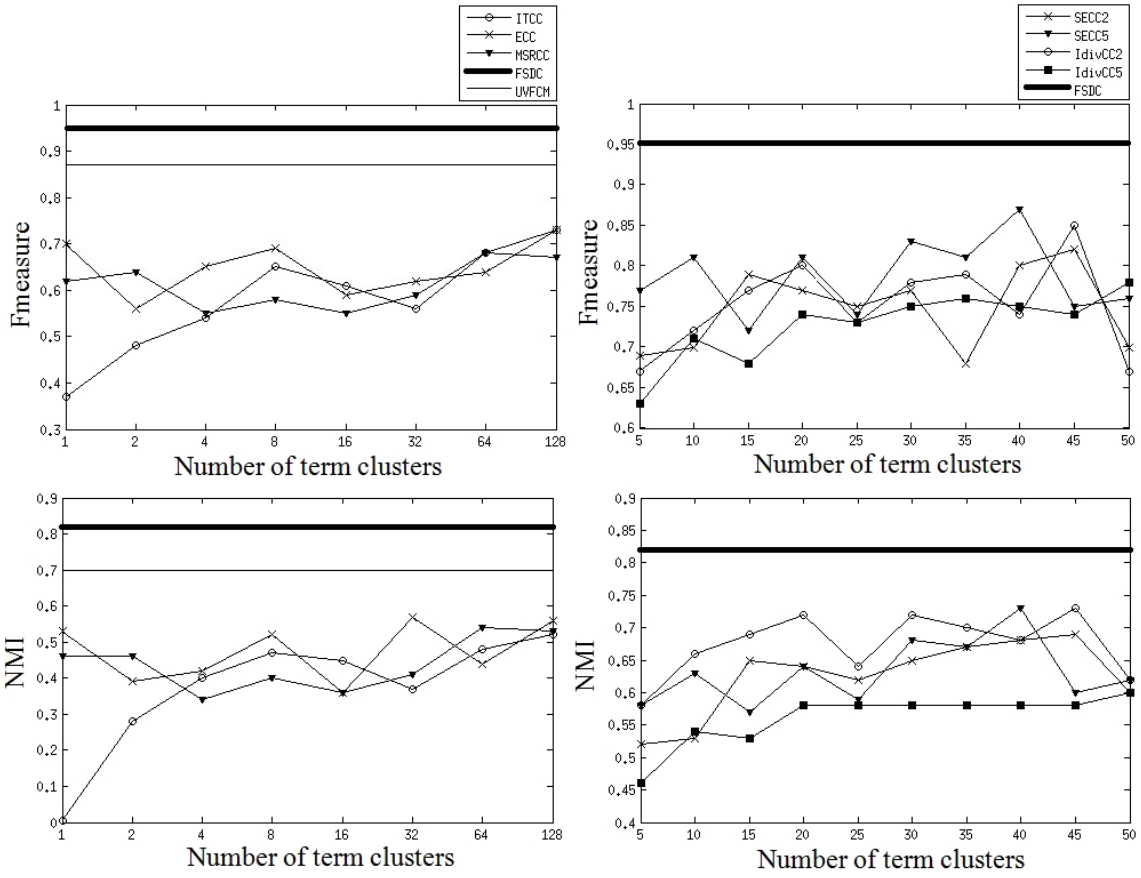

Figure 4.7: The quality of clusters for *FSDC* and the co-clusterers on Classic4. *FSDC* is the best clustering algorithm and the quality of its clusters are much better than those of the competitors. The outputs of the co-clusterers (*SECC2*, *SECC5*, *IdivCC2*) are sensitive to the number of term clusters.

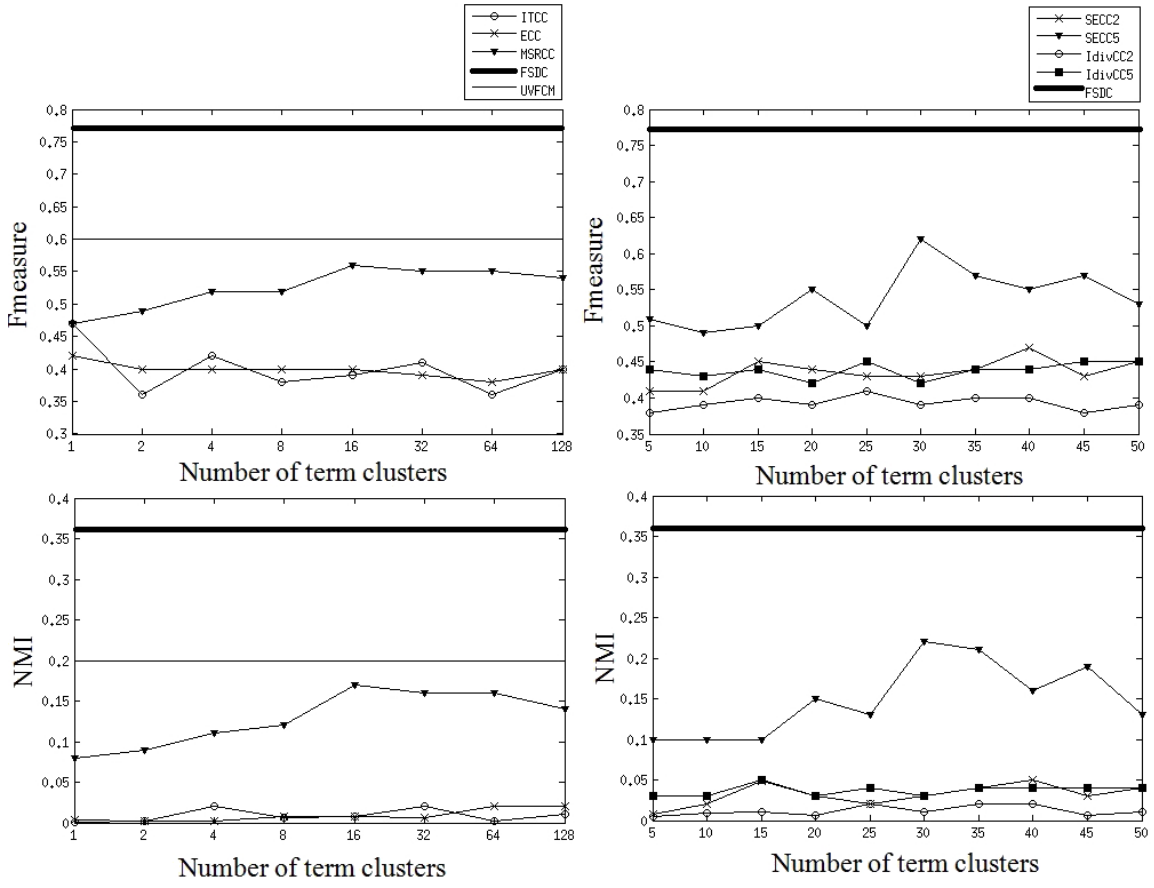

Figure 4.8: The quality of clusters for *FSDC* and the co-clusterers on News-sim3. *FSDC* is the best clustering algorithm and the quality of its clusters are much better than those of the competitors.

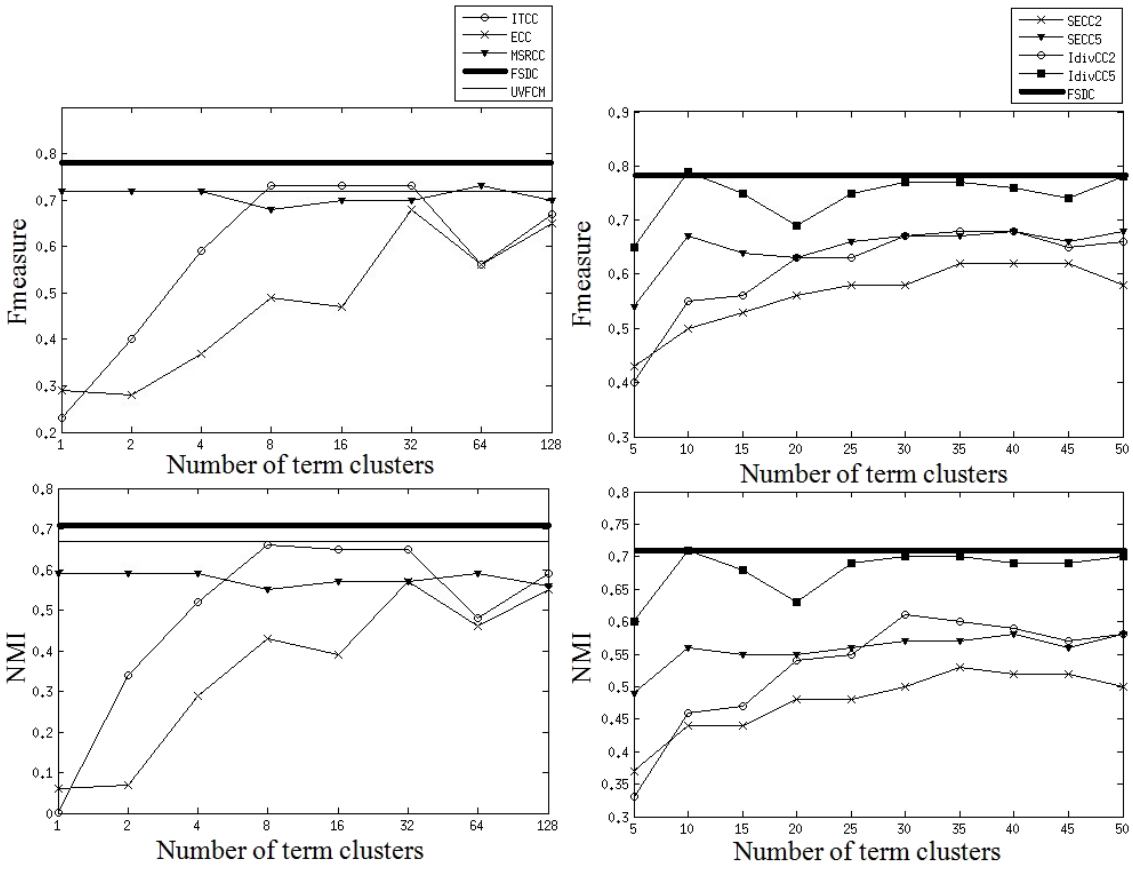

Figure 4.9: The quality of clusters for *FSDC* and the co-clusterers on News-multi7. *FSDC* is the best clustering algorithm. Only *IdivCC5* could generate the same results in terms of quality.

Table 4.4: Averaged *Fmeasure*s and standard deviations obtained for the competitors in 20 runs. *FSDC* is the best clusterer for all datasets in our experiment except for *News-multi7*. Direct clusterers (*BI2*, *BH2*, *DI2*, *DH2*) outperformed the other competitors on *News-multi7*.

	Cacmcisi	Reviews	Classic4	News-sim3	News-multi7
FSDC	$\mathbf{0.97 \pm 10^{-6}}$	$\mathbf{0.71 \pm 2 \times 10^{-4}}$	$\mathbf{0.95 \pm 0.003}$	$\mathbf{0.77 \pm 0.002}$	0.78 ± 0.002
UVFCM	0.86 ± 0.105	0.68 ± 0.003	0.87 ± 0.004	0.60 ± 0.07	0.72 ± 0.057
ITCC	0.79 ± 0.029	0.62 ± 0.089	0.73 ± 0.116	0.47 ± 0.036	0.73 ± 0.18
ECC	$\mathbf{0.96 \pm 0.013}$	0.53 ± 0.081	0.73 ± 0.058	0.42 ± 0.011	0.68 ± 0.15
MSRCC	0.93 ± 0.116	0.57 ± 0.015	0.67 ± 0.051	0.56 ± 0.031	0.73 ± 0.016
SECC2	$\mathbf{0.96 \pm 0.003}$	0.60 ± 0.041	0.82 ± 0.051	0.47 ± 0.019	0.62 ± 0.061
SECC5	$\mathbf{0.96 \pm 0.003}$	0.64 ± 0.028	0.87 ± 0.047	0.62 ± 0.041	0.68 ± 0.042
IdivCC2	0.92 ± 0.032	0.59 ± 0.051	0.85 ± 0.057	0.41 ± 0.01	0.68 ± 0.087
IdivCC5	0.79 ± 0.017	0.68 ± 0.02	0.78 ± 0.044	0.45 ± 0.012	0.79 ± 0.045
BI2	0.62 ± 0.009	0.70 ± 0.025	0.69 ± 0.047	0.61 ± 0.09	0.82 ± 0.032
BH2	0.74 ± 0.005	$\mathbf{0.71 \pm 0.024}$	0.71 ± 0.029	0.66 ± 0.098	$\mathbf{0.85 \pm 0.057}$
DI2	0.62 ± 0.006	0.70 ± 0.057	0.68 ± 0.033	0.62 ± 0.085	0.84 ± 0.053
DH2	0.74 ± 0.004	$\mathbf{0.71 \pm 0.036}$	0.73 ± 0.036	0.60 ± 0.113	0.84 ± 0.063
k-means	0.92 ± 0.149	0.65 ± 0.054	0.70 ± 0.09	0.52 ± 0.054	0.68 ± 0.05
FCM	0.81 ± 0.088	0.58 ± 0.029	0.64 ± 0.064	0.53 ± 0.072	0.57 ± 0.086

Table 4.5: Averaged *NMI*s and standard deviations obtained for the competitors in 20 runs. *FSDC* is the best clusterer for all datasets in our experiment except for *News-multi7*. Direct clusterers (*BI2*, *BH2*, *DI2*, *DH2*) outperformed the other competitors on *News-multi7*.

	Cacmcisi	Reviews	Classic4	News-sim3	News-multi7
FSDC	$\mathbf{0.76 \pm 2 \times 10^{-5}}$	$\mathbf{0.57 \pm 5 \times 10^{-4}}$	$\mathbf{0.82 \pm 0.005}$	$\mathbf{0.36 \pm 0.003}$	$0.71 \pm 2 \times 10^{-4}$
UVFCM	0.48 ± 0.162	0.50 ± 0.003	0.70 ± 0.004	0.2 ± 0.05	0.67 ± 0.09
ITCC	0.37 ± 0.0063	0.39 ± 0.0228	0.52 ± 0.0423	0.02 ± 0.0017	0.66 ± 0.0246
ECC	$\mathbf{0.76 \pm 0.0060}$	0.33 ± 0.0197	0.56 ± 0.0656	0.02 ± 0.0057	0.57 ± 0.0225
MSRCC	0.61 ± 0.0098	0.37 ± 0.0139	0.53 ± 0.0245	0.17 ± 0.0079	0.59 ± 0.0061
SECC2	0.74 ± 0.0626	0.43 ± 0.0740	0.69 ± 0.0168	0.05 ± 0.0179	0.52 ± 0.0530
SECC5	0.72 ± 0.0022	0.48 ± 0.0656	0.73 ± 0.0121	0.22 ± 0.0604	0.58 ± 0.0346
IdivCC2	0.65 ± 0.148	0.39 ± 0.0331	0.73 ± 0.0623	0.02 ± 0.0093	0.59 ± 0.0351
IdivCC5	0.39 ± 0.0337	0.49 ± 0.061	0.60 ± 0.0531	0.04 ± 0.0228	0.71 ± 0.031
BI2	0.23 ± 0.004	0.52 ± 0.025	0.6 ± 0.057	0.24 ± 0.11	$\mathbf{0.76 \pm 0.02}$
BH2	0.32 ± 0.005	$\mathbf{0.57 \pm 0.017}$	0.59 ± 0.028	0.27 ± 0.11	0.73 ± 0.059
DI2	0.23 ± 0.004	0.55 ± 0.063	0.58 ± 0.05	0.24 ± 0.091	$\mathbf{0.76 \pm 0.03}$
DH2	0.32 ± 0.005	0.56 ± 0.052	0.60 ± 0.028	0.22 ± 0.104	0.75 ± 0.038
k-means	0.58 ± 0.31	0.45 ± 0.069	0.56 ± 0.093	0.14 ± 0.060	0.58 ± 0.056
FCM	0.42 ± 0.156	0.34 ± 0.035	0.43 ± 0.059	0.13 ± 0.065	0.41 ± 0.21

Table 4.6: The averaged running times of the methods in the experiments in 20 runs are reported in seconds.

	Cacmcisi	Reviews	Classic4	News-sim3	News-multi7
FSDC	304.8	612.2	560.6	356	981.7
UVFCM	8.4	32	28.4	22.5	48
ITCC	5.8	12.8	15.8	5.3	20.3
ECC	5.4	11.8	15.5	4.8	19.2
MSRCC	5.2	11.7	16.2	4.9	19.6
SECC2	188.3	827.1	752.8	315.8	1416.8
SECC5	287.5	921.4	864.2	345.3	1200.4
IdivCC2	139.5	345.1	344.9	205	378.2
IdivCC5	189.2	514.6	525.6	264.4	532
BI2	5.4	20.2	28	7.8	31.7
BH2	5.9	25.2	29.6	8.4	33.2
DI2	5.1	17.2	20	5.4	26.6
DH2	5.5	19.6	21.1	5.9	27.8
k-means	3	20.7	22.8	11.4	75
FCM	5	32.8	27.2	18	54.4

and some co-clustering algorithms outperform the direct clusterers. This fact implies that term clustering can enhance the performance of document clustering for *Cacmcisi*. *K-means* and *FCM* are the best algorithms among the direct clusterers. *FSDC* and *ECC* have generated similar results but *ECC* has a lower running time.

The quality of clusterings obtained based on *Reviews* leads us to a contrary conclusion. For this dataset, most direct clusterers (*BI2*, *BH2*, *DI2*, and *DH2*) outperform the co-clustering algorithms. This means that direct clustering is a better choice for this dataset. *FSDC* and *BH2* have generated similar clusters but *BH2* has a lower running time.

Experiments on *Classic4* return the superiority to the algorithms that use term clustering. Based on *NMI* and *Fmeasure* the best algorithms for this dataset use term clustering. *FSDC* significantly outperformed the other competitors but its running time is larger than the others except for *SECC2* and *SECC5*.

FSDC significantly outperformed the other algorithms on *News-sim3* but it has the largest running time. The documents of this dataset have similar topics. Direct clusterers generate better clusters in this case.

Four direct clusterers (*BI2*, *BH2*, *DI2*, and *DH2*) outperform the other algorithms on *News-multi7*. They have lower running times compared to *FSDC*. The main difficulty of clustering this dataset is in separating the two classes *comp.sys.mac.hardware* and *misc.forsale*. Most co-clustering algorithms put these two classes into a single cluster. This difficulty for *FSDC* originates from term clustering, where fuzzy *c*-means clusters keyterms of these two classes into one term cluster.

We conjecture that the advantage of the clusterers *BI2*, *BH2*, *DI2*, and *DH2*, in clustering *News-multi7*, come from their iterative refinement phase [118]. During this phase, each document is revisited and moved to another cluster provided that it results in an improvement in the criterion function.

Overall, we had 15 algorithms in our experiments. Nine algorithms use term clustering and six algorithms cluster documents directly. There is no clusterer that can outperform the others on all datasets. Besides, co-clustering and double clustering, or using term clusters in general do not result in better clusterings in all cases. Despite the fact that term clusters can improve the quality of clusters for some datasets, there are other datasets where using direct clustering is a better option. The experimental

results confirm the fact that "there is no best clustering algorithm" [63].

4.4 User-Supervised *FSDC*

In this section, we want to show how term labeling is effective in clustering *News-multi7*. As mentioned in the previous section, most clustering algorithms have difficulty in separating two classes of *News-multi7*. Experimental results of this section demonstrate that the quality of clusterings obtained by *FSDC* can be improved with a few iterations of term labeling.

We first show how *FSDC* can be used interactively in the form of term labeling. The interaction phase starts after term clusters are generated as shown in Fig. 4.2. Top keyterms of the term clusters are extracted and displayed to the user and she labels the keyterms (groups them in meaningful clusters). The labeled keyterms are then fed into fuzzy c-means to re-cluster terms. This interaction can be performed for a few iterations until the user decides to terminate.

We choose f high-variance terms from each term cluster to make a list of candidate terms. The list is shown to the user and she labels the terms using the following actions:

1. Assign a term to a cluster correctly
2. Assign a term to a cluster randomly
3. Remove a term from the list

The labeled terms are then fed into fuzzy c-means to re-cluster terms. The labeled terms are used to initialize the membership matrix U of fuzzy c-means . The membership matrix U has M rows corresponding to the M term and k columns corresponding to the k term clusters in this case.

We initialize the membership matrix in a way that all entries are set randomly except for the labeled terms. We set the corresponding entry of a labeled term and its cluster to one in the matrix. The initialized matrix is then used to re-run fuzzy c-means. We perform term labeling for a few iterations until term clusters match the user's preferences.

To evaluate the user-supervised *FSDC*, we use the oracle introduced in [54]. The oracle knows the true label of terms based on the χ^2 statistic. These labels are computed in the following way:

1. Given the class label of documents, the χ^2 statistic computes the relevance of each term in each class.

2. Each term is then assigned to the class with maximum relevance and the class label is saved in the oracle.

The oracle has a parameter P_{exp} that indicates the degree of user's expertness. Based on this parameter, the user either labels a term correctly or she says *"I do not know"*. She might remove the term from the list or assign it to another class randomly in case of *"I do not know"*. The probability of remove is set to 0.5 and the probability of random assignment is set to $0.5/k$. The main steps of the simulated term labeling are shown in Algorithm 3.

Algorithm 3 Simulated term labeling

```
for each term on the list do
    if rand[0,1] < P_exp then
        assign label correctly
    else if rand[0,1] < 0.5 then
        remove the term from the list
    else
        generate a random number between 1 and k
        assign the random label
    end if
end for
```

In the first experiment we evaluated the effect of P_{exp} on the quality of clusters. The results of this experiment are shown in Fig. 4.10. For each value of P_{exp}, the algorithm is run 20 times and the average *NMI*s and *Fmeasure*s are reported. The plots of quality measures show that a user with minimum expertness of 0.5 can improve the quality of clusters in this case. The number of iterations is set to three, and the number of keyterms of each term cluster is set to 20 in this experiment.

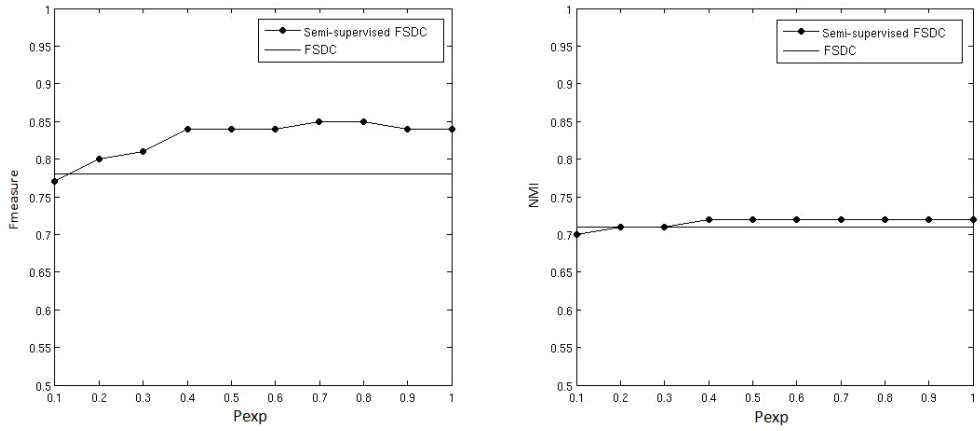

Figure 4.10: The effect of the degree of user expertness (P_{exp}) on the quality of clusters. The minimum expertness of 0.5 results in an stable improvement.

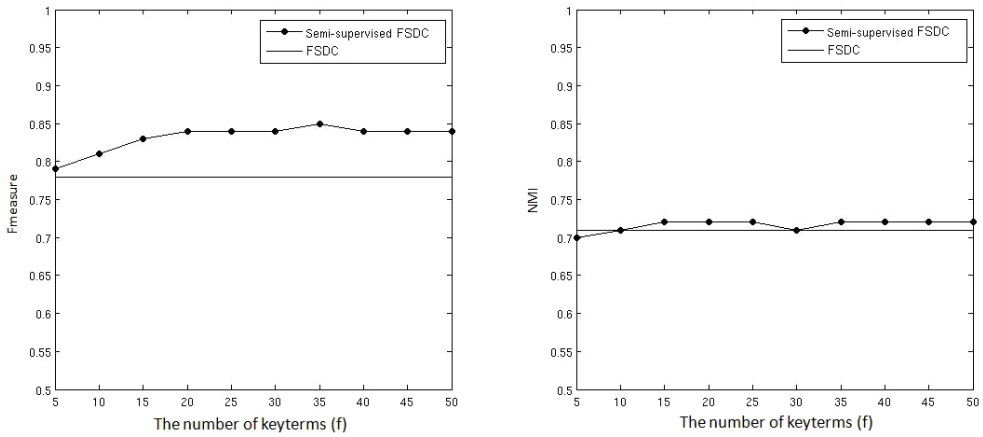

Figure 4.11: The effect of the number of keyterms extracted from each term cluster (f) for term labeling. 15 to 20 keyterms are enough to improve the quality of clusters for *News-multi7*.

In the second experiment, we evaluated the effect of f (the number of keyterms of each term cluster) on the quality of clusters. The results are shown in Fig. 4.11. For each value of f, the algorithm is run 20 times and the average *NMI*s and *Fmeasure*s are reported. As shown in this figure, 15 to 20 keyterms are enough for term labeling of *News-multi7*. The number of iteration is set to three and P_{exp} is set to 0.5 in this experiment.

In the third experiment, we examined the number of iterations needed to get the best results. The number of keyterms (f) is set to 20 and P_{exp} is set to 0.5 in this experiment. The effect of different numbers of iterations is shown in Fig. 4.12. For each number of iterations, the algorithm is run 20 times and the average measures are

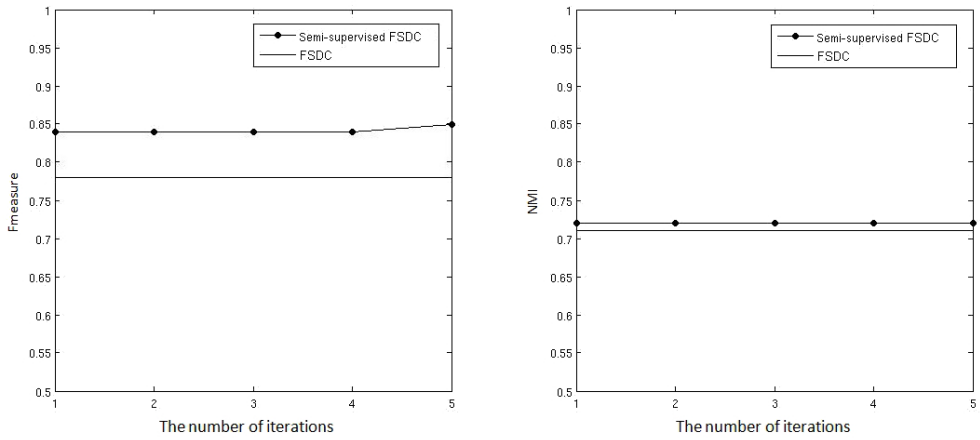

Figure 4.12: The quality of clusters after different number of rounds of user supervision. Evan after one iteration of supervision, the quality of clusters increases.

reported. We can see that even after one iteration *FSDC* generates better results for *News-multi7*. It means that Fuzzy c-means can learn the user's desired term clustering after one iteration in this case.

The conclusion we get from these experiments is that if the degree of simulated user's expertness is at least 0.5 and 15 to 20 keyterms are extracted from each term cluster, the simulated user can improve the quality of clusters for *News-multi7* even after one iteration of term labeling.

We have used the conclusions of these experiments to propose a user-supervised partitional algorithm. The algorithm is explained in detail in Chapter 7.

4.5 Conclusion

We proposed a multiobjective genetic algorithm so as to distill term clusters. For each term cluster, the algorithm keeps only those terms whose subspace includes the maximum number of representative documents.

To find the representative documents, we proposed a method based on the nature of text datasets. We take advantage of zero values that exist in most entries of document-term matrices. For this purpose, the k-means algorithm is applied on the centroids of distilled term clusters.

The output of the multiobjective algorithm is a set of non-dominated solutions for each term cluster. Each non-dominated solution corresponds to a distilled term

cluster. The representative documents of these distilled term clusters are used as seeds to cluster documents. Only one distilled term cluster of each term cluster is needed for this purpose. To find the best distilled clusters, we used a standard genetic algorithm.

Our experimental results demonstrated that distilling term clusters can result in better document clusters compared to the competitors in which term clusters are used without any prior analysis.

A user-supervised algorithm is also proposed in this chapter. The experimental results show that with a few iterations of term labeling, the simulated users can improve the quality of document clusters when the unsupervised algorithm has difficulty in separating classes.

The main drawback of *FSDC* is that the supervision is based on the terms extracted from term clusters not from document clusters. It would provide a better insight into the topics of a collection if the terms are extracted from the document clusters. We propose an interactive clustering algorithm in Chapter 7 along with more comprehensive experiments to show the effectiveness of term labeling in document clustering.

Chapter 5

Partitional Double Clustering

We propose a non-evolutionary partitional clustering algorithm in this chapter. Compared to the evolutionary algorithm (*FSDC*) proposed in Chapter 4, this partitional algorithm has the following benefits:

- A greedy approach is used to distill term clusters instead of using the *MOGA* module described in Section 4.2.2. The greedy approach removes non-discriminative terms using a feature selection method. This change decreases the running time of *LDC* significantly.

- A soft term clustering is generated in the partitional algorithm, while a hard term clustering is used in *FSDC*. We use a different defuzzification method for this purpose. Since similar topics share common terms, a term can be related to multiple term clusters in this way.

- The weight of seed documents in document clusters is not considered in *FSDC*. A seed document might be related to multiple document clusters as described in Section 4.1.2. We use the weight of seeds in computing document centroids.

- Since the MOGA module is replaced by a greedy approach, there is only one distilled term cluster for each term cluster in the partitional algorithm. Therefore, the genetic algorithm used in *FSDC*, to find the best non-dominated solutions, is not needed in this algorithm.

Removing the genetic algorithms from our clustering method results in a significant improvement in running time. We have conducted an experiment to compare the proposed partitional clustering to *FSDC*. The experiment reveals how much the quality of clusterings is deteriorated due to replacing the *MOGA* module by the greedy approach.

The remainder of this chapter is organized as follows. Section 5.1 explains the proposed algorithm in detail. Section 5.2 covers the experiments performed to evaluate

this algorithm. This section includes the experiments performed to compare this algorithm to *FSDC*. It also includes the experiments performed to compare this algorithm to the *LDA* model. The experiments conducted to compare Google based distance and *TFIDF* based cosine similarity are also reported in this section. We conclude this chapter with conclusions and future work.

5.1 Methodology

A text corpus is represented as a document-term matrix using the *BOW* model in this algorithm. The importance of terms in documents are computed using *TFIDF*. The proposed partitional algorithm consists of the following three phases:

1. Term clustering and topic keyterm selection
2. Finding lexical seed documents
3. Document clustering

The main steps of the algorithm are depicted in Fig. 5.1 and Algorithm 4. Since the algorithm is based only on document contents, we call it Lexical Double Clustering (*LDC*) in the rest of this report.

LDC clusters documents in the following way. Fuzzy c-means groups the terms into k term clusters. It then removes general terms from the term clusters since these terms deteriorate the performance of our clustering algorithm [90]. We consider the remaining terms in term clusters as topic keyterms.

Given the topic keyterms as the input in Phase 2, it extracts the representative documents, which are used as seeds to cluster all documents later.

In Phase 3, a document centroid is computed based on the seed documents of each term cluster. The distances among documents and the centroids are then used to cluster documents. The time complexity of *LDC* is $O(NMK^2I)$, where I is the maximum number of iterations of fuzzy c-means, which is fixed to 50 in this thesis.

5.1.1 Term Clustering and Topic Keyterm Selection

Given a document-term matrix, we apply fuzzy c-means on the term vectors to generate k term clusters. After fuzzy c-means is done, each term cluster includes only terms

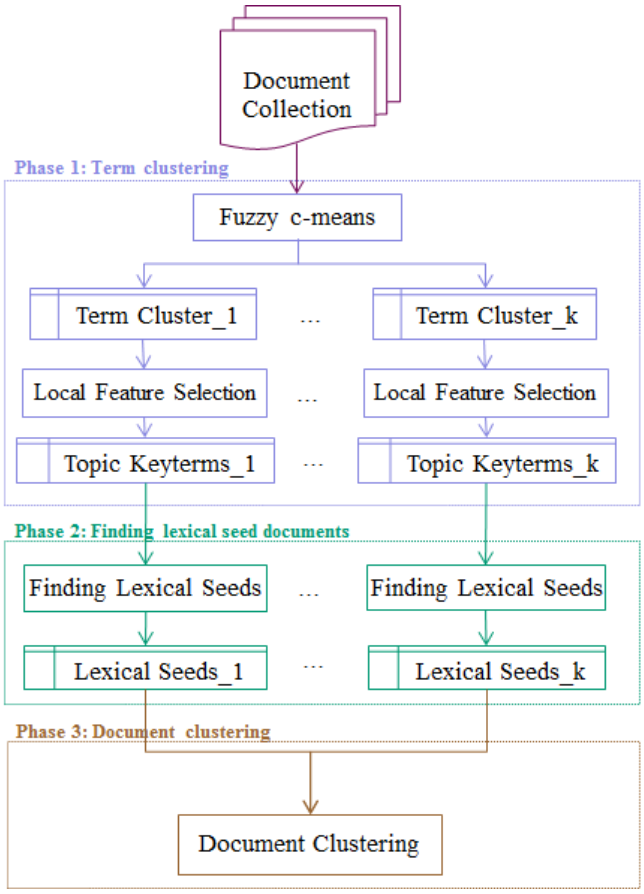

Figure 5.1: The structure of the lexical clustering algorithm, Algorithm 4. Fuzzy c-means is used for term clustering. A greedy approach distills the term clusters through feature selection in order to remove non-discriminative terms. Representative documents associated with each term cluster are then extracted and used as seeds to cluster all documents. No user interaction is involved and document clustering is unsupervised.

Algorithm 4 Lexical Double Clustering (*LDC*)

Input: *a document-term matrix* M_{DT}, *k*

Output: *k document clusters* $\{W_p\}_{p=1}^{k}$

1: *use fuzzy c-means, Algorithm 1, to generate k term clusters* $\{TC_p\}_{p=1}^{k}$
2: *remove non-discriminative terms from the term clusters*
3: **for** *each distilled term cluster* TC_p **do**
4: *extract lexical representative (seed) documents*
5: *compute a document centroid for the representative documents:*

$$lexicalCentroid_p = \frac{\sum_{d_i \in lexicalSeeds(TC_p)} w_i * d_i}{|lexicalSeeds(TC_p)|}$$

6: **end for**
7: **for** *each document in the dataset* d_i **do**
8: *measure its similarities to the document centroids*
9: *assign the document to each document centroid with a membership value:*

$$P(W_p|d_i) = \frac{sim(d_i, lexicalCentroid_p)}{\sum_{j=1}^{k} sim(d_i, lexicalCentroid_j)}$$

10: **end for**

whose memberships are greater than $1/k$. This defuzzification method results in a soft term clustering. The value of $1/k$ makes sense here since if a term is equally related to all clusters, it would have the same membership value of $1/k$ in all clusters. The time complexity of fuzzy *c*-means is $O(NMK^2I)$, where I is the number of iterations [42].

We assume that only a few terms represent topics and the other terms are non-discriminative [67]. We propose a greedy approach to extract these topic keyterms. Our greedy approach consists of the following steps:

1. For each term, we first compute its score using the *Var-TFIDF* feature selection method described in Section 2.1.2.

2. We then compute an average score for each term cluster. The average score is the mean of scores of terms that exist in a term cluster.

3. In each term cluster, those terms whose scores are smaller than the cluster average score are removed. The remaining terms are considered as topic keyterms and used in the next phase.

The time complexity of computing *Var-TFIDF* scores is $O(NM)$. The output of this

phase are k term clusters distilled by the greedy approach and it has a time complexity of O(NMK^2I).

5.1.2 Finding Lexical Seed Documents

The input of this phase consists of k distilled term clusters, each including a set of topic keyterms. Each set characterizes a set of seed documents. The seed documents are those documents that are close to each other in the subspace spanned by the topic keyterms. We extract the seed documents of each term cluster using the method described in 4.1.2. The method consists of the following steps:

1. A term-centroid is first generated for each term cluster based on the document-term matrix representation. The term-centroid is the column average of the term vectors corresponding to the terms included in a term cluster. It is a vector with dimensionality equal to the number of documents. The time complexity of this step is O(NMK), where M is the number of terms in the distilled term clusters.

2. The k-means algorithm with $k = 2$ is then applied on the term-centroid (treated as a one-dimensional space) to partition its elements into two clusters. One cluster includes elements with near-zero values and the other cluster includes elements with larger non-zero values. The near-zero values correspond to the documents in which the terms of a term cluster have low frequencies. The elements with larger non-zero values correspond to the seed documents. The time complexity of this step is O(NKI).

It is noteworthy to mention that a seed document can be linked with more than one term cluster in this phase. However, the *TFIDF* value of the document in a term-centroid indicates the weight of the document in each term cluster. We use these weight values in computing document centroids. These extracted seed documents are called lexical seeds in the rest of the report. The time complexity of this phase is O($NFK + NKI$).

5.1.3 Document Clustering

The input of this phase is the lexical seed documents of the term clusters. Given the lexical seeds, we use the following steps to cluster documents:

1. For each term cluster TC_p, we compute a document centroid over its seed documents. A document centroid is the row average of the document vectors corresponding to the seed documents:

$$\text{lexicalCentroid}_p = \frac{\sum_{d_i \in \text{lexicalSeeds}(TC_p)} w_i * d_i}{|\text{lexicalSeeds}(TC_p)|} \tag{5.1}$$

   where w_i is the averaged *TFIDF* value of d_i in the term-centroid of TC_p, and $||$ indicates the cardinality of a set. The time complexity of this step is O(NMK).

2. The distances of each document to the lexical centroids are then computed. The memberships of documents in document clusters are then measured as the inverse of these distances. In this way, each document has a membership value in each document cluster. The time complexity of this step is also O(NMK).

A hard partitioning of a collection can be generated by assigning each document to the closest centroid. The time complexity of this Phase is O(NMK).

5.2 Experimental Results

We have conducted several experiments to evaluate the performance of *LDC* algorithm. We have compared *LDC* to the *FSDC* algorithm proposed in Chapter 4. We have then compared *LDC* to the *LDA* model. An experiment has also been conducted to see whether a Google Ngram based distance is better than Cosine similarity in term clustering. We have evaluated the performance of *LDC* based on different feature selection methods described in Section 2.1.2. We have finally compared Euclidean distance to Cosine similarity metric based on *LDC*.

5.2.1 Comparison to the *FSDC* Algorithm

The experiment of this section reveals whether replacing the *MOGA* module with the feature selection technique in distilling term clusters, results in deterioration in the

Table 5.1: Comparison between the *LDC* and the *FSDC* algorithms. Each algorithm is run 20 times on datasets *Cacmcisi*, *Reviews*, *Classic4*, *News-multi7*, and *News-sim3*. The average quality of clusterings in 20 runs are reported in the form of *Fmeasure*s (*NMI*s).

	Cacmcisi	Reviews	Classic4	News-multi7	News-sim3
FSDC	0.97(0.76)	0.71(0.57)	0.95 (0.82)	0.78 (0.71)	0.77 (0.36)
LDC	0.96(0.73)	0.68(0.53)	0.93 (0.81)	0.77 (0.69)	0.68 (0.29)

Table 5.2: Comparison between the *LDC* and the *FSDC* algorithms based on running times. Each algorithm is run 20 times on datasets *Cacmcisi*, *Reviews*, *Classic4*, *News-multi7*, and *News-sim3*. The average running times of these algorithms in 20 runs are reported in seconds.

	Cacmcisi	Reviews	Classic4	News-multi7	News-sim3
FSDC	304.8	612.2	560.6	981.7	356
LDC	8.2	48.3	34.2	59.4	21

quality of clusterings. We have run *FSDC* and LDC 20 times on five datasets. The average performance of algorithms are reported in Table 5.1.

The experiment demonstrates that the *FSDC* algorithm always outperforms *LDC* especially on the dataset *News-sim3*, which has three similar topics. This confirms that the *MOGA* module is better than the feature selection method used in *LDC* in order to distill term clusters. This advantage is because that a population of solutions is iteratively evolved in evolutionary algorithms. However, *LDC* is faster than *FSDC* since no evolutionary algorithms are used in. The average running times of these algorithms are shown in Table 5.2. On average, *LDC* is 21 times faster than *FSDC* in this experiment. The other advantage of *LDC* is that it can be easily enhanced with user supervisions as proposed in Chapter 7.

5.2.2 Comparison to the *LDA* Model

We compare *LDC* to the *LDA* model using eight text datasets. Since the generative process of *LDA* is based on the co-occurrence of terms in the documents [17], *LDA* is run on a version of datasets which are only preprocessed by stop-word removal and stemming steps. No feature selection is performed on the datasets for *LDA*. We used

a C++ implementation[1] of this model in this experiment. The number of iterations for the model is set to 10,000 and the number of topics is set to k. The value of k is user-defined. After 10,000 iterations, we assign each document to the topic (cluster) with maximum probability. The maximum number of iterations for fuzzy c-means is set to 50 in *LDC*. Each document is similarly assigned to the closest cluster in *LDC* (hard partitioning).

We also measured the running time of *LDC* for each dataset in the worst case when fuzzy c-means is run for all 50 iterations. We then let the *LDA* model run for these running times instead of 10,000 iterations. We call this algorithm *LDA*[RelTime] in experimental results. By running both algorithms for the same time on the same machine, we provided a more fair performance comparison. We have run these algorithms 50 times. The average and the standard deviation of *NMI*s and *Fmeasure*s in these 50 runs are depicted in Fig. 5.2 and Fig. 5.3.

This experiment clearly demonstrated the performance of *LDC*. *LDC* can create similar clusterings to the state-of-the-art text clustering method, the *LDA* model. There is no significant difference in the quality of clusterings except for two datasets, *Cade* and *SMS*. Based on *NMI*, *LDA* models significantly outperformed *LDC* on *Cade*. On the other hand, *LDC* significantly outperformed *LDA* models on dataset *SMS* based on *Fmeasure* and *NMI*. Based on the empirical results obtained in this experiment, we can conclude that *LDC* and the *LDA* model generate similar results, especially if they run for the same time. The time complexity of *LDC* is $O(NMK^2I)$ and the time complexity of Gibbs sampling based *LDA* is $O(KVI)$ [107, 114], where V is the number of terms (tokens) in a collection and I is the number of iterations. The number of iterations in *LDC* is the maximum number of iterations of fuzzy c-means, which is fixed to 50 in this thesis.

5.2.3 Google Ngram Distance vs. Cosine Distance

We compare two distance functions for term clustering in this experiment. The first distance function is the cosine distance based on *TFIDF* that has been used so far in our work. The second distance function is based on the Google Corpus [61]. We use these distances only for term clustering, Phase 1 of the *LDC* algorithm. The goal of

[1] http://gibbslda.sourceforge.net/

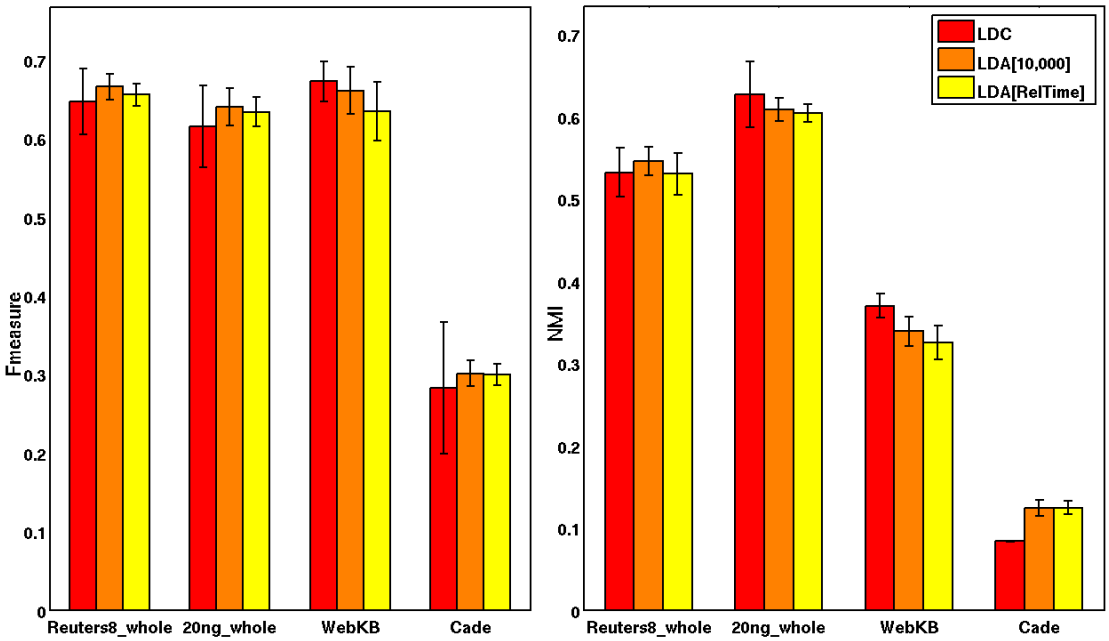

Figure 5.2: The quality of clusters obtained from *LDC*, Algorithm 4 and the LDA models in 50 runs. None of the algorithms in this experiment outperforms the others on all datasets.

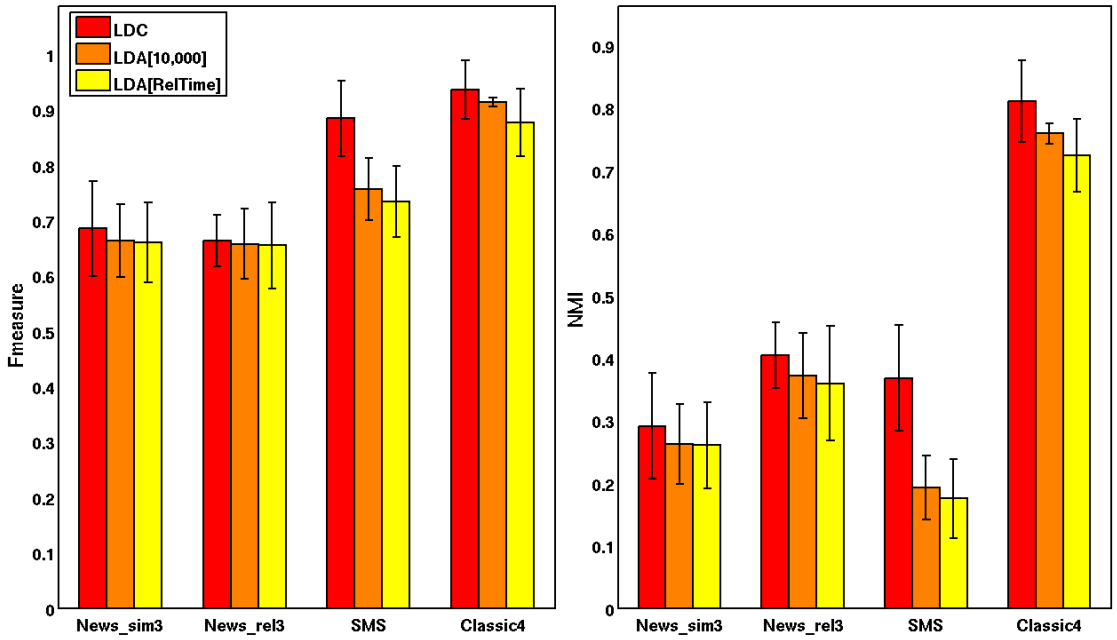

Figure 5.3: The quality of clusters obtained from *LDC*, Algorithm 4 and the LDA models in 50 runs. *LDC* outperforms both *LDA* models on the *SMS* dataset, where documents are short. Otherwise, they generate similar clusters.

this experiment is that we like to find out whether a similarity based on an external source (the Google Ngrams) results in a better term clustering than the document vector representation of terms.

A method is proposed in [61] to measure the similarity of texts using Google word Ngrams. The novelty of the proposed method is in measuring the similarity of a pair of words based on the Google Corpus. The idea is based on the frequency of the tri-grams that start and end with the given pair of words. The tri-gram frequency is then normalized by the frequencies of uni-grams. Given a pair of words (w_a, w_b), the similarity $Sim(w_a, w_b) \in [0, 1]$ is measured using the following formula:

$$Sim(w_a, w_b) = \begin{cases} \frac{\log \frac{\mu(w_a,n_1,w_b,n_2)C^2}{c(w_a)c(w_b)\min(c(w_a),c(w_b))}}{-2*\log \frac{\min(c(w_a),c(w_b))}{C}} & \text{if } \frac{\mu(w_a,n_1,w_b,n_2)C^2}{c(w_a)c(w_b)\min(c(w_a),c(w_b))} > 1 \\ \frac{\log(1.01)}{-2*\log \frac{\min(c(w_a),c(w_b))}{C}} & \text{if } \frac{\mu(w_a,n_1,w_b,n_2)C^2}{c(w_a)c(w_b)\min(c(w_a),c(w_b))} \leq 1 \\ 0 & \text{if } \mu(w_a, n_1, w_b, n_2) = 0 \end{cases} \tag{5.2}$$

$$\mu(w_a, n_1, w_b, n_2) = \frac{1}{2}(\sum_{i=1}^{n_1} c(w_a w_i w_b) + \sum_{i=1}^{n_2} c(w_b w_i w_a)) \tag{5.3}$$

where C is the maximum frequency among all Google uni-grams, $c(w)$ is the frequency of word w in Google uni-grams, n_1 is the number of tri-grams start with w_a and end with w_b, and n_2 is the number of tri-grams start with w_b and end with w_a. The distance between w_a and w_b is computed as $1 - Sim(w_a, w_b)$. We use this distance function in fuzzy c-means to generate term clusters.

According to the formula of the objective function of fuzzy c-means (Eq. 4.1), the Google Ngram distance cannot be used directly in term clustering. This is simply because the centroid (μ) of each term cluster is not a real term itself (Eq. 4.4). It is a weighted mean vector of term vectors. It is thus impossible to compute the Google Ngram distance of a term to a centroid using Eq. 5.2.

On solution is to use fuzzy c-medoids instead of fuzzy c-means. The fuzzy c-medoids algorithm [70] is a version of fuzzy c-means, where each cluster is represented by an instance (medoid) instead of an average mean vector (centroid). Rather than computing the distances of instances to centroids, the distances of instances to medoids are computed. The fuzzy c-medoids algorithm locally minimizes the following function:

$$J_z = \sum_{i=1}^{M} \sum_{p=1}^{k} u_{ip}^{z} dist^2(t_i, v_p) \tag{5.4}$$

where v_p is an representation instance (medoid) of cluster C_p.

Given the membership matrix U, those instances that minimize Eq. 5.4 are selected as medoids in each iteration. The new membership matrix is then computed based on the selected medoids using the following formula:

$$u_{ij} = \frac{(dist(x_i, v_j))^{2/(1-z)}}{\sum_{p=1}^{k} (dist(x_i, v_p))^{2/(1-z)}} \tag{5.5}$$

Using the fuzzy c-medoids algorithm, we can use the Google distance function in term clustering since the representative of a term cluster is itself a real word. However, a single term should not be used as a medoid in term clustering. This is because a single term does not contain enough data to represent a term cluster [2]. This is due to the nature of text, where each term may appear only in a fraction of all documents that exist in a corpus.

A solution of this problem is multi-medoids representation. A cluster can be represented by multiple instances instead of one instance. It seems more reasonable to represent each term cluster by multiple terms. Instead of using the distance of a term to a single medoid, its average distance to medoids is used in this case. Therefore, the distance term of Eq. 5.4 and Eq. 5.5 is replaced by the following formula:

$$dist(t_i, v_p) = \frac{1}{m_p} \sum_{l=1}^{m_p} dist(t_i, v_{pl}) \tag{5.6}$$

where m_p is the number of medoids used to represent C_p, and v_{pl} is its l^{th} medoid. We have used this fuzzy c-medoids algorithm in order to examine Google distance function in term clustering in the following way.

We have run *LDC* 50 times using fuzzy c-medoids once based on Google Ngram distance and once based on Cosine similarity. Different number of medoids is used in this experiment for Google Ngram based distance, $m_p = \{10, 20, ..., 150\}$. We have run *LDC* 50 times for each value of m_p. The average and standard deviation of the evaluation measures obtained in this experiment are depicted in Fig. 5.4 to Fig. 5.8 and in Fig. A.1 to Fig. A.3. Eight datasets are used in this experiment, where six datasets are stemmed and two datasets *Reuters8-subset* and *20ng-subset* are non-stemmed.

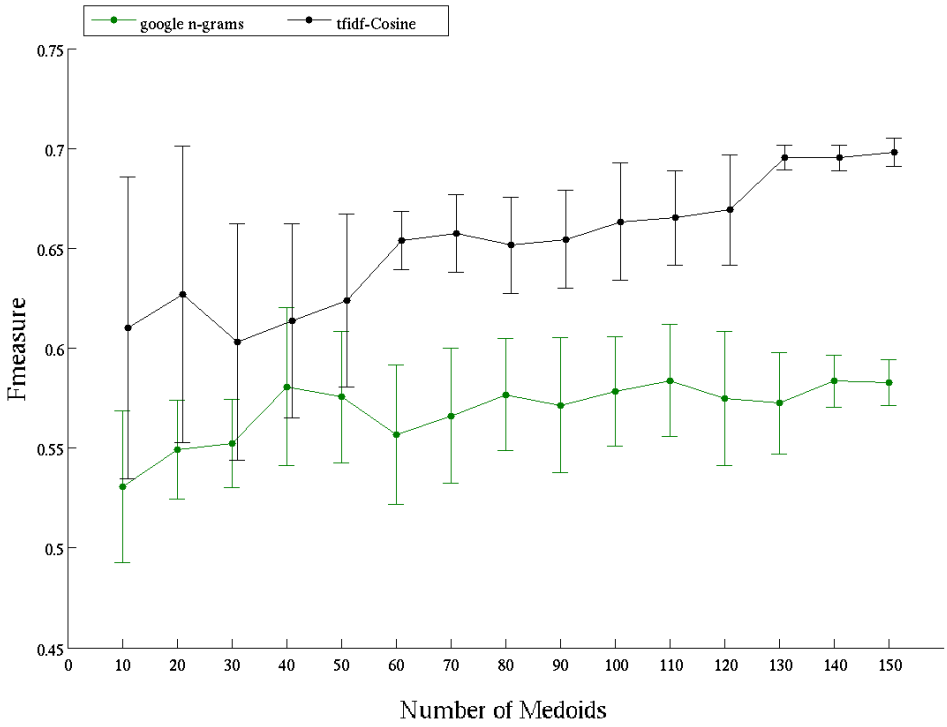

Figure 5.4: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *News-rel3*. The Cosine similarity generate better results than the Google Ngram distance.

The evident observation in this experiment is that the Cosine similarity based clusterings are superior to the Google Ngram distance based clusterings, regardless of the similarity of text topics or the number of document clusters.

According to the experiments performed in this section, we can conclude that the proposed text clustering algorithm (*LDC*) generates better clusterings by using the *TFIDF*-based Cosine similarity than the Google Ngram based distance. We have also tried to combine these two distances but no improvement is obtained in our experiments.

5.2.4 Fuzzy c-means vs. Fuzzy c-medoids

In this experiment, we evaluate the performance of *LDC* using two fuzzy term clusterers: fuzzy *c*-means and fuzzy *c*-medoids. The *TFIDF*-based Cosine similarity is used for term clustering. The goal of this experiment is that we intend to figure whether representing term clusters by single terms results in better term clusters than

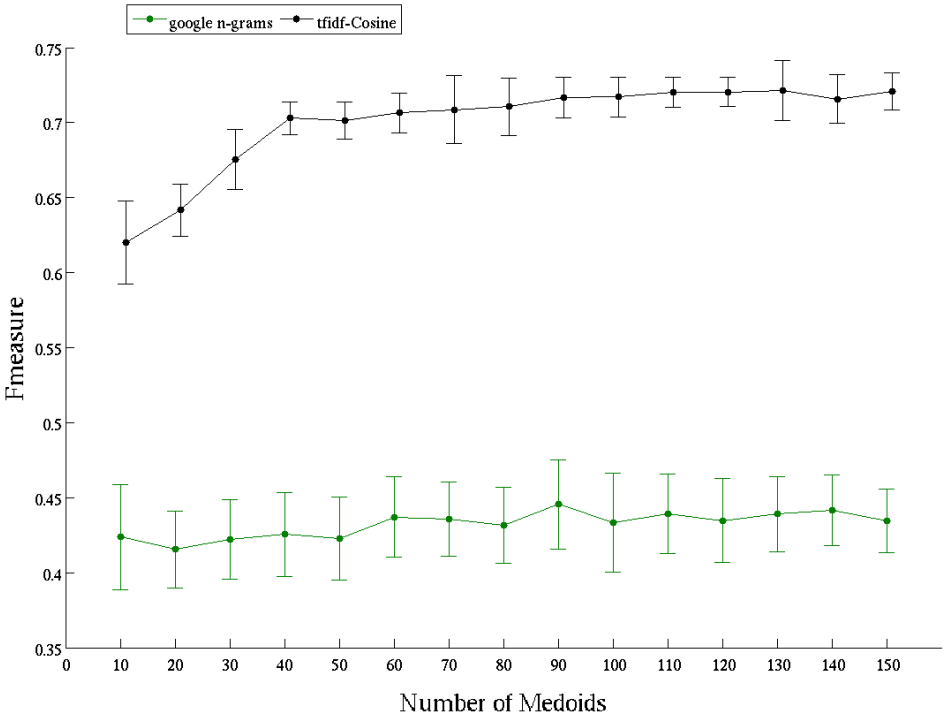

Figure 5.5: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *News-multi10*. Clustering based on Cosine similarity are superior to the clustering based on the Google Ngram distance.

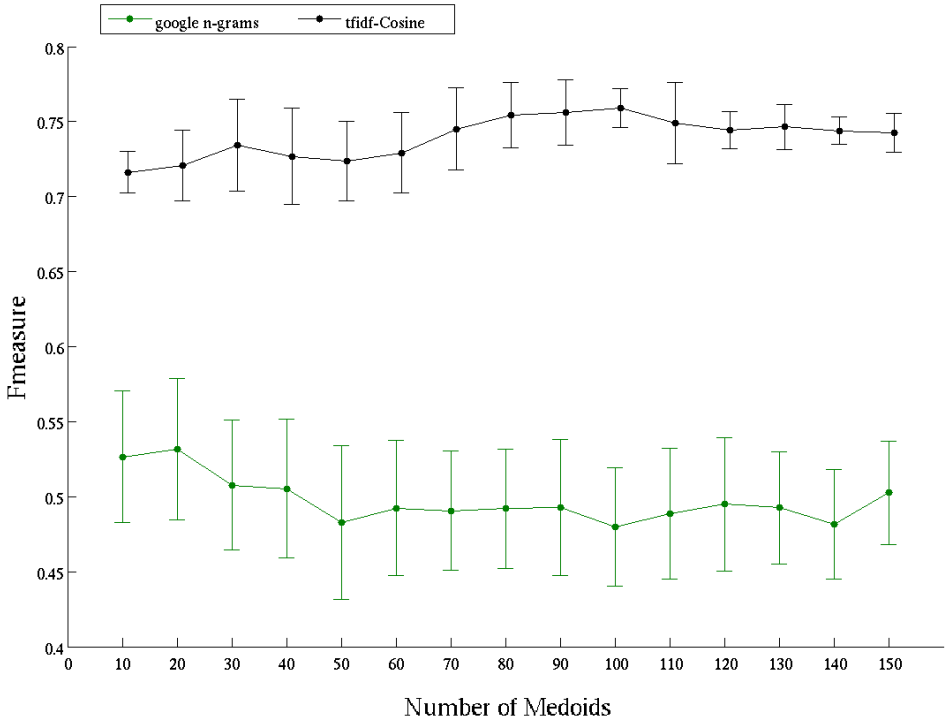

Figure 5.6: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *News-multi7*. Cosine similarity results in better clustering than the Google Ngram distance and its standard deviations are smaller.

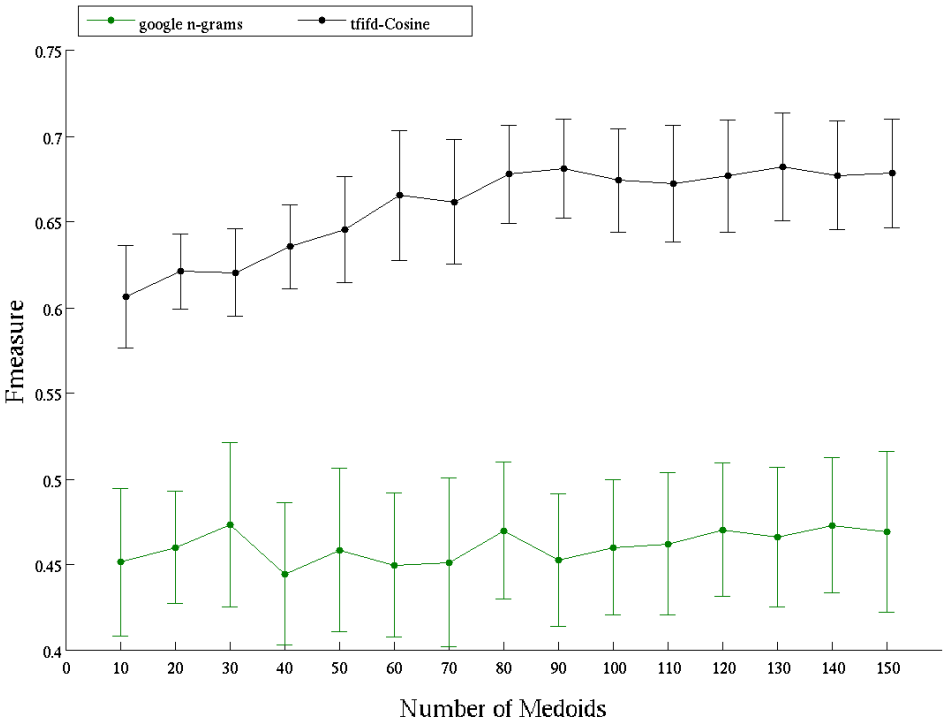

Figure 5.7: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *Reuters8-subset*. The *TFIDF*-based Cosine similarity results in better clustering than the Google Ngram based distance.

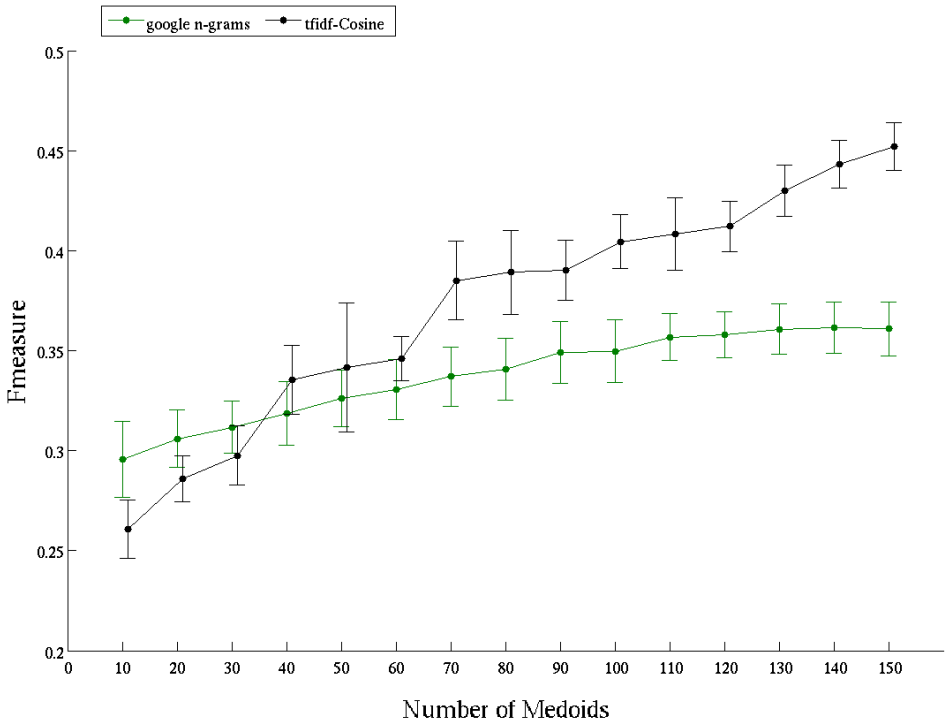

Figure 5.8: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *20ng-subset*. The clusterings obtained based on the Google Ngram distance is better than those obtained based on Cosine similarity if the number of medoids is less than 40. As the number of medoids increases from 40 to 150, the *TFIDF*-based Cosine similarity results in better clusterings.

representing them by term centroids.

The outputs of this experiment are plotted in Fig. 5.9 to Fig. 5.12 and in Fig. B.1 to Fig. B.4. *LDC* is run 50 times on each dataset using fuzzy *c*-means and the average *Fmeasure*s are depicted as a horizontal line with a standard deviation as the error bar. It is also run 50 times using fuzzy *c*-medoids with different number of medoids. The average *Fmeasure*s and standard deviations of the respective number of medoids are depicted in the plots.

The main observation in this experiment is that as the number of medoids increases, the quality of clusters obtained from fuzzy *c*-medoids increases. However, fuzzy *c*-means results in better document clusters than fuzzy *c*-medoids in terms of quality in most cases. Fuzzy *c*-medoids outperforms fuzzy *c*-means only on *Reuters8-subset* and on *News-rel3*, where the number of medoids is more than 100. We can conclude that representing term clusters by centroids is better than representing them by medoids in this experiment. This is due to the nature of text where a single term alone is not a good representative for a term cluster.

5.2.5 Evaluating Feature Selection Methods

The goal of this experiment is to evaluate the performance of *LDC* based on the feature selection methods described in Section 2.1.2. Feature selection is used in Phase 1 of *LDC* so as to distill term clusters by removing low-score terms.

We have run *LDC* 50 times for different feature selection methods. The average and standard deviation of the quality of clusterings obtained are reported in Table 5.3. The first value in each cell corresponds to the average of *Fmeasure*s and the second one corresponds to the average of *NMI*s.

Entropy Rank and Var-*TFIDF* resulted in the best clusterings in this experiment, based on the mean values of evaluation measures. The time complexity of *Entropy Rank* is $O(N^2M^2)$ while Var-*TFIDF* has $O(NM)$ time complexity. The obvious conclusion of this experiment is that either method can be used for distilling term clusters in *LDC* if the running time is not a big concern. Otherwise, Var-*TFIDF* is the better method in case of large datasets.

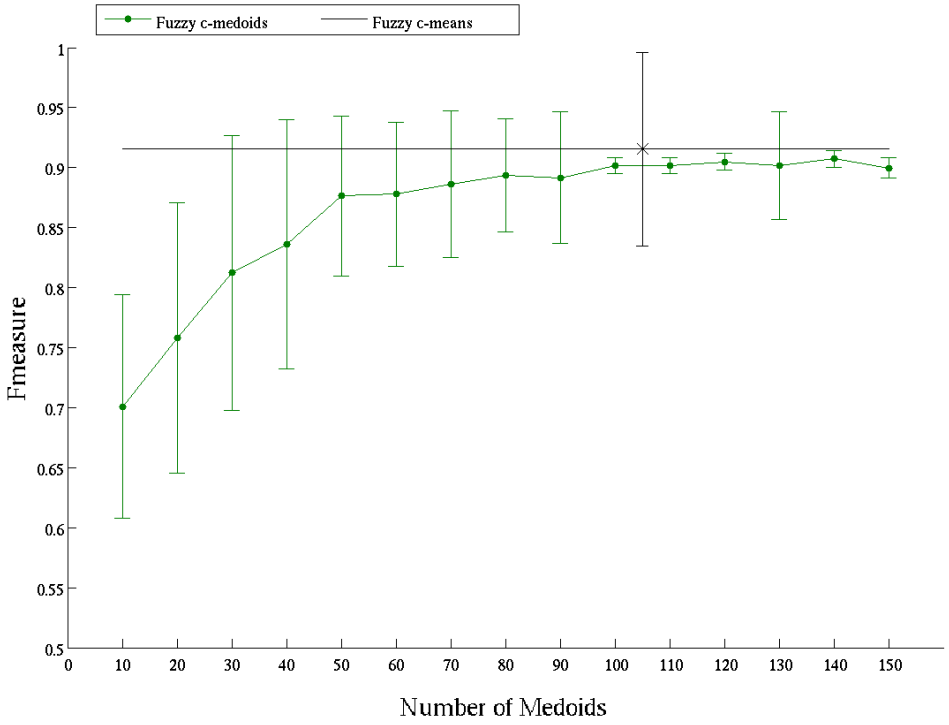

Figure 5.9: The quality of clusters obtained from *LDC* using fuzzy *c*-means and fuzzy *c*-medoids on *Classic4*. As the number of medoids increases to 100, fuzzy *c*-medoids generates comparable results and the standard deviations are smaller.

Table 5.3: Average and standard deviation of the quality of clusterings generated by *LDC* in 50 runs based on different feature selection methods. The first value in each cell corresponds to the average of *Fmeasure*s and the second one corresponds to the average of *NMI*s.

	Mean-*TFIDF*	Var-*TFIDF*	*Entropy Rank*	*Term Contribution*
Classic4	0.93 ± 0.07 0.81 ± 0.08	$\mathbf{0.94 \pm 0.04}$ $\mathbf{0.82 \pm 0.04}$	0.93 ± 0.08 0.81 ± 0.08	0.93 ± 0.06 0.80 ± 0.07
LA Times	0.65 ± 0.04 0.48 ± 0.04	0.66 ± 0.03 0.49 ± 0.03	$\mathbf{0.66 \pm 0.02}$ $\mathbf{0.50 \pm 0.03}$	0.65 ± 0.02 0.48 ± 0.03
News-sim3	0.70 ± 0.07 0.31 ± 0.07	$\mathbf{0.72 \pm 0.06}$ $\mathbf{0.32 \pm 0.06}$	0.70 ± 0.08 0.30 ± 0.08	0.70 ± 0.07 0.30 ± 0.06
News-rel3	0.65 ± 0.05 0.37 ± 0.07	$\mathbf{0.66 \pm 0.05}$ $\mathbf{0.40 \pm 0.06}$	0.66 ± 0.06 0.38 ± 0.08	0.64 ± 0.07 0.35 ± 0.09
News-multi7	0.75 ± 0.04 0.68 ± 0.03	0.76 ± 0.04 0.69 ± 0.02	$\mathbf{0.78 \pm 0.03}$ $\mathbf{0.72 \pm 0.02}$	0.75 ± 0.04 0.67 ± 0.02
News-multi10	0.74 ± 0.03 0.66 ± 0.01	0.76 ± 0.03 0.67 ± 0.01	$\mathbf{0.77 \pm 0.03}$ $\mathbf{0.70 \pm 0.02}$	0.73 ± 0.03 0.65 ± 0.01

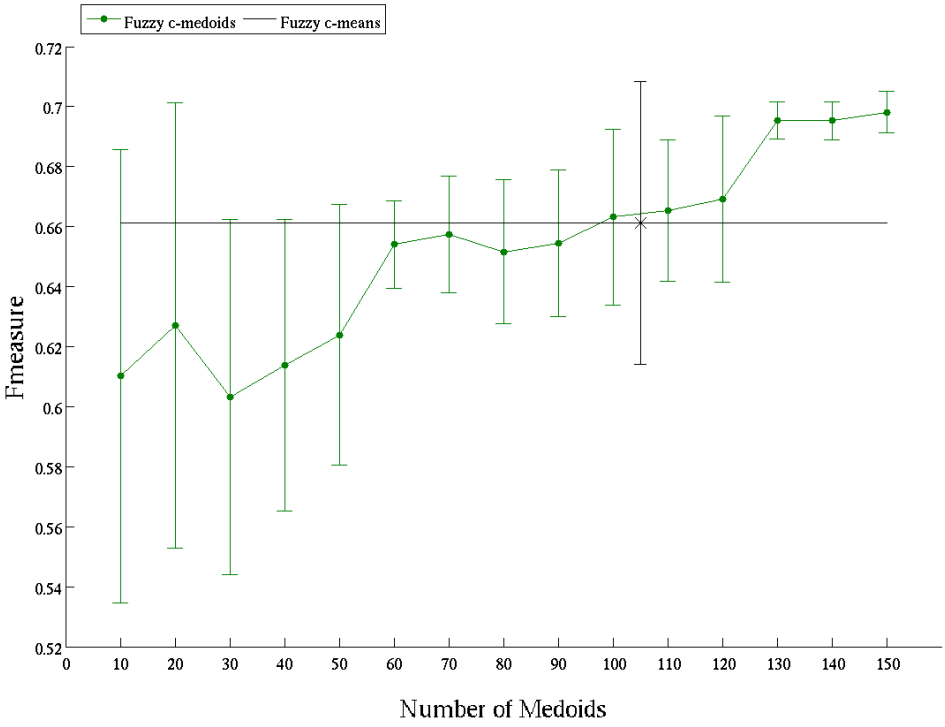

Figure 5.10: The quality of clusters obtained from *LDC* using fuzzy *c*-means and fuzzy *c*-medoids on *News-rel3*. The fuzzy *c*-medoids term clusterer outperforms fuzzy *c*-means if the number of medoids are greater than 100. Its standard deviations are also smaller after 100 medoids.

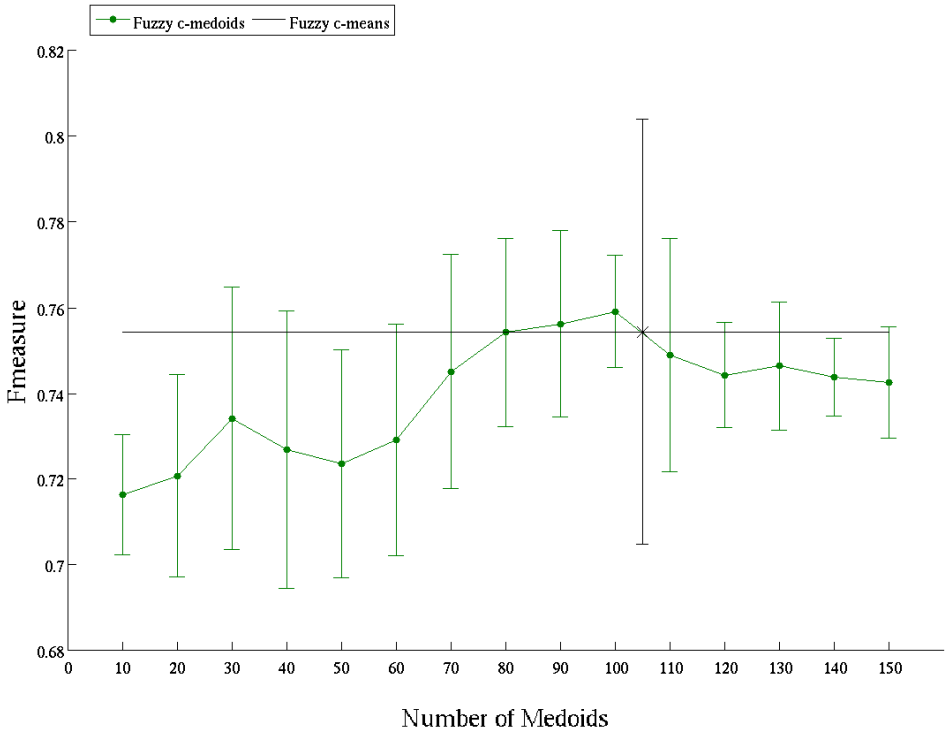

Figure 5.11: The quality of clusters obtained from *LDC* using fuzzy *c*-means and fuzzy *c*-medoids on *News-multi7*. The fuzzy *c*-medoids generates better results if the number of medoids is around 100. The standard deviation of the quality of clusters obtained by using fuzzy *c*-medoids is smaller.

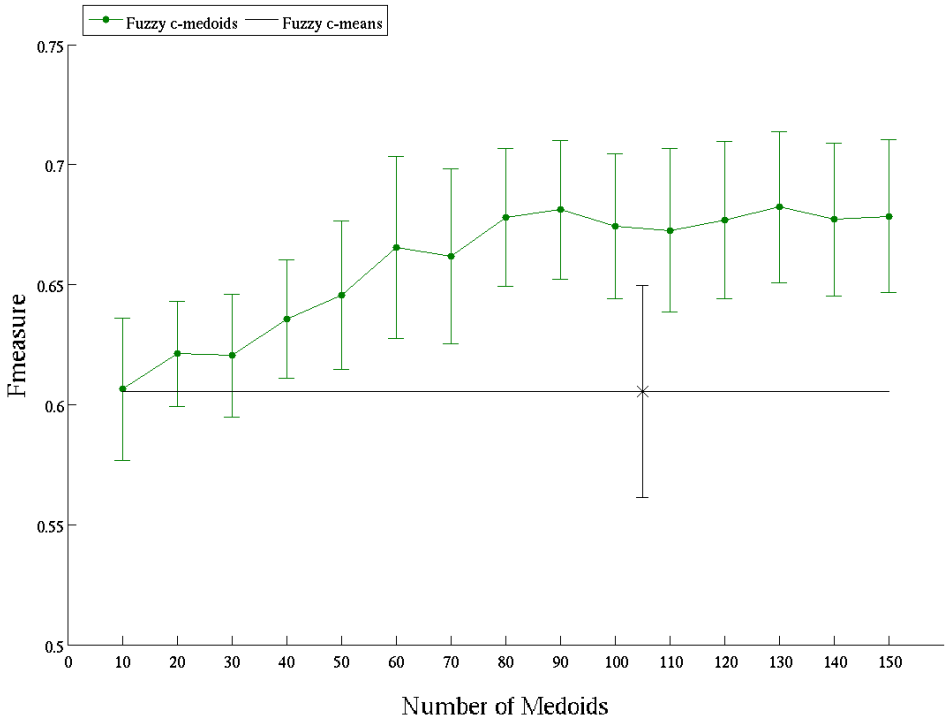

Figure 5.12: The quality of clusters obtained from *LDC* using fuzzy *c*-means and fuzzy *c*-medoids on *Reuters8-subset*. Fuzzy *c*-medoids generates better clusters regardless of the number of medoids.

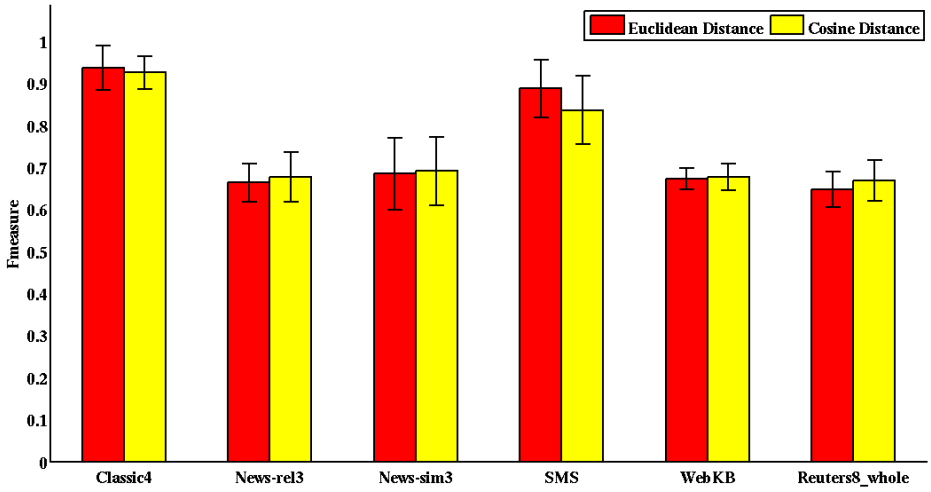

Figure 5.13: The quality of clusters obtained from *LDC* using Euclidean distance and Cosine distance based on *Fmeasure*. Based on paired-sample T-test, there is no statistically significant difference between the quality of clusterings on each dataset.

5.2.6 Euclidean Distance vs. Cosine Similarity

The goal of this experiment is to evaluate the performance of *LDC* when the document similarity is measured based on Euclidean distance and Cosine distance. Term clustering is still based on Cosine distance in this experiment.

We have run *LDC* 50 times based on these two distance metrics. The average and standard deviation of the quality of clusterings obtained are shown in Fig. 5.13 and Fig. 5.14.

The observation of this experiment is that there is no significant difference in the quality of clusterings in this experiment and *LDC* generates similar results based on these two distance metrics.

5.3 Conclusion

We proposed a novel partitional text clustering algorithm in this chapter. The main novelty of this algorithm is that a feature selection method distills term clusters so as to remove non-discriminative terms. The main advantage of this algorithm is that we can easily involve user supervision in the clustering process as proposed in Chapter 7.

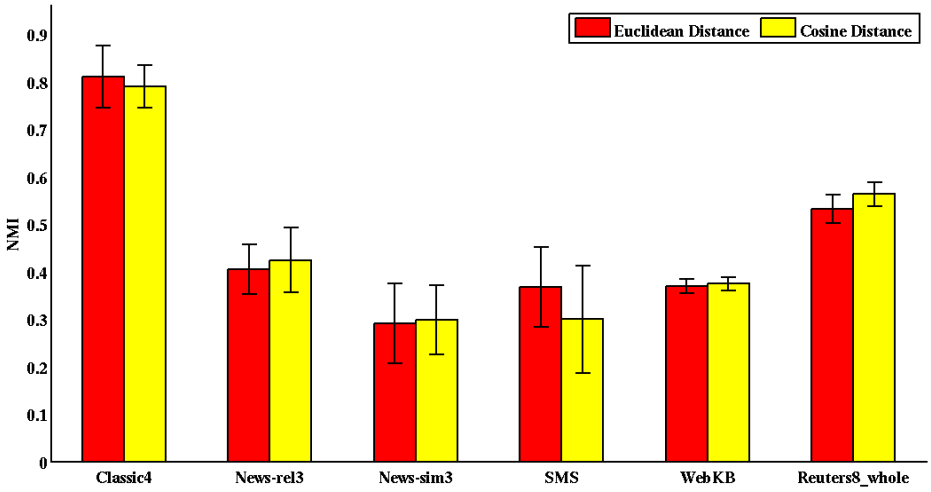

Figure 5.14: The quality of clusters obtained from *LDC* using Euclidean distance and Cosine distance based on *NMI*. Based on paired-sample T-test, there is no statistically significant difference between the quality of clusterings on each dataset.

We have conducted several experiments to evaluate the performance of this clustering algorithm. The experimental results showed that the proposed algorithm can generate comparable results to the *LDA* model, which is one of the top document clustering approaches in the literature.

We have then compared Google Ngram based similarity to the *TFIDF*-based Cosine similarity in term clustering. The experimental results demonstrated that *TFIDF*-based Cosine similarity is a better similarity metric in this algorithm.

An experiment has also been conducted to evaluate the performance of the proposed algorithm based on two fuzzy term clustering algorithms, fuzzy c-means and fuzzy c-medoids. The experimental results revealed that it is better to represent a term cluster by term centroid instead of single terms.

We have finally compared the performance of our proposed algorithm based on different feature selection methods. Var-*TFIDF* was the winner of this experiment based on the computational time complexity and the quality of the clusterings obtained.

Chapter 6

Integrating Wikipedia Concepts in Text Clustering

Traditional text clustering algorithms usually represent a document collection as a document-term matrix in the bag of words (BOW) model. The model is based on the idea that related documents have common terms, while unrelated documents are formed by different vocabulary barely share any terms. The representation is limited to the term frequencies in documents and no semantic relation among terms is considered. For instance, two documents with the same topic would sit in two different clusters if they are formed by different but semantically related terms. One solution to this problem is to enrich the document representation by using the external resources like WordNet and Wikipedia.

Several research works have exploited Wikipedia in text clustering. The document representation of BOW is augmented in [8] utilizing top relevant Wikipedia articles. The title of selected articles are appended to the content of documents and the best performance is obtained by doubling the weights of terms appearing in the titles. A framework is proposed in [51] to enhance the traditional document similarity measures using the semantic relations extracted from Wikipedia. Different combinations of the semantic relations (synonyms, hypernyms, and associated relations) with traditional similarity measures are evaluated in experimental results. A linear combination of cosine similarities based on document-term representation and document-concept representation is also proposed in [58] to enhance the document similarity measure. A similar approach is proposed in [52] to enhance the document similarity measure. Document contents are first mapped to Wikipedia concepts and categories in this algorithm. The document cosine similarity measure is then combined with cosine similarities of document concepts and categories. No significant improvement is obtained for the partitional text clustering and the approach is more effective in hierarchical text clustering.

Wikipedia categories are also used in [96] to enrich document representation.

Experimental results demonstrate that document-category representation is not as good as document-term representation in partitional text clustering. Some improvements are obtained only when a combination of document contents and Wikipedia categories are used.

Wikipedia concepts are used in [57] to actively find pair-wise constraints for a semi-supervised clustering algorithm. A document-concept representation is first created for the collection. All the extracted concepts are then clustered. Those documents with higher weights in concept clusters are then submitted to a noise-free oracle to form *must-link* and *cannot-link* constraints.

A framework to label document clusters is proposed in [19]. The application of the framework is in interactive text clustering, where an interface is provided for users. Instead of representing a document cluster by its top keyterms, the framework exploits categories and titles of the relevant Wikipedia articles to assign a label.

A graph based distance among Wikipedia articles is presented in [117]. Nodes of the graph are articles and edges are weighted by their content or link similarity. The documents are mapped to the nodes based on the cosine similarity among their contents and articles' text. A random walk model is then proposed to measure node distance. Node distance is then used to measure document distance.

The *BOW* document representation is replaced by a concept model representation using the features extracted from Wikipedia articles in [103]. The concept model representation is then used in a hierarchical algorithm to cluster the documents.

Document representation is also enriched by using WordNet. Synonyms and hypernyms extracted from WordNet are used in [49] to represent documents instead of (or in combination with) their contents. However, the coverage of WordNet is limited and it is not comprehensive enough to find all the concepts mentioned in a document collection [52].

Overall, there are three approaches to enrich document representation of the *BOW* model:

1. The *BOW* model is completely replaced by a conceptual model.

2. The *BOW* model is enriched by the information extracted from an external source, like relevant concepts added to the content of documents.

3. The document similarity measure based on *BOW* is enhanced by a similarity measure based on a conceptual model.

To the best of our knowledge, no one has proposed an ensemble clustering algorithm to combine the clusterings generated based on document terms and concepts. This work shows that we can improve the quality of clusters using the proposed ensemble algorithm even when representing documents by Wikipedia concepts solely, results in poor clusterings.

We propose a new framework in order to integrate the Wikipedia concepts in partitional document clustering. We use the method proposed in Section 4.1.2 to find lexical seed documents based on document terms. We propose another method to find semantic seed documents based on concepts. We then generate two clusterings of a collection based on semantic and lexical seeds. An ensemble module finally combines the clusters generated by these seed documents. To evaluate the performance of our method, we performed empirical experiments on some real text datasets. Our experimental results show that the quality of clusters is significantly improved by utilizing Wikipedia concepts.

The remainder of this chapter is organized as follows. Section 6.1.1 describes a partitional clustering algorithm based on Wikipedia concepts. Section 6.1.2 explains our ensemble clustering algorithm. Experimental results on some real text datasets are reported in Section 6.2. Section 6.3 presents conclusions.

6.1 Methodology

In this section, we first show how *LDC* algorithm can be applied on document concepts. We then propose our ensemble clustering algorithm.

6.1.1 Semantic Double Clustering

Besides the document-term representation, we represent a document as a bag of concepts (*BOC*) extracted from Wikipedia. The wikified concepts are the titles of the relevant articles. Each entry of the document-concept matrix is a feature value computed based on *TFIDF*. Given the document-concept matrix, we can use the *LDC* algorithm proposed in Section 5.1 to cluster documents. The semantic clustering

algorithm consists of the following steps:

1. **Concept Clustering**: Fuzzy c-means is used to cluster concepts (columns of the document-concept matrix). It groups the concepts into k concept clusters. After concept clustering is done, each concept cluster includes only concepts whose memberships are greater than $1/k$. This method of defuzzification results in a soft clustering. Non-discriminative concepts are then removed from the concept clusters using the greedy approach proposed in Section 5.1.1

2. **Finding Seed Documents**: Given the distilled concept clusters, seed documents are extracted using the method described in Section 5.1.2.

3. **Document Clustering**: For each distilled concept cluster, we compute a document centroid over its seed documents. The distances of each document to the centroids are then used for document clustering.

The algorithm is quite similar to the *LDC* algorithm where terms are replaced by concepts. The main steps of this Semantic Double Clustering (*SDC*) are shown in Algorithm 5.

Algorithm 5 Semantic Double Clustering (*SDC*)

Input: *a document-concept matrix* M_{DC}, k

Output: k *document clusters* $\{W_p\}_{p=1}^{k}$

1: *use fuzzy c-means, Algorithm 1, to generate k concept clusters* $\{CC_p\}_{p=1}^{k}$
2: *remove non-discriminative concepts from the concept clusters*
3: **for** *each distilled concept cluster* CC_p **do**
4: *extract conceptual representative (seed) documents*
5: *compute a document centroid over the representative documents:*

$$conceptCentroid_p = \frac{\sum_{d_i \in conceptSeeds(CC_p)} w_i * d_i}{|conceptSeeds(CC_p)|}$$

6: **end for**
7: **for** *each document in the dataset* d_i **do**
8: *measure its similarities to the document centroids*
9: *assign the document to each document centroid with a membership value:*

$$P(W_p|d_i) = \frac{sim(d_i, conceptCentroid_p)}{\sum_{j=1}^{k} sim(d_i, conceptCentroid_j)}$$

10: **end for**

6.1.2 Ensemble Lexical-Semantic Double Clustering

Given the documents in the *BOW* and *BOC* models, our ensemble algorithm clusters a document collection in the following phases:

1. Term clustering and topic keyterm selection
2. Finding lexical seed documents
3. Finding semantic seed documents
4. Document clustering using the consensus method

Phases 1 and 2 are the same as Phases 1 and 2 of our partitional algorithm, Section 5.1.1 and Section 5.1.2. Fuzzy c-means clusters terms (columns of the document-term matrix) into k groups. The *Var-TFIDF* feature selection method distills term clusters by removing non-discriminative terms. The lexical seed documents are then extracted based on the document-term representations. Given the lexical seeds, we compute the lexical document centroids using Eq. (5.1).

In Phase 3, we present a method to semantically find seed documents. We first extract the relevant Wikipedia concepts of each distilled term cluster by wikifying its terms. We also have the Wikipedia concepts of each document. We thus check to see if any document has a common concept with the concepts of a term cluster. We consider these documents as semantic seeds. The number of common concepts is also considered as the weight of the semantic seed documents. Document centroids are then computed as the weighted mean of the semantic seed documents. We call these document centroids, semantic centroids in the rest of this chapter. Hence, the output of Phases 2 and 3 are two document centroids, lexical and semantic, for each distilled term cluster.

Given the lexical and semantic centroids, we generate two clusterings of the documents in Phase 4. The first clustering is generated by only using the lexical centroids and the second one is generated by using the semantic centroids.

Reconciling the generated clusters reveals that some documents sit in the same clusters in these two clusterings. It means that these documents belong to the same clusters based on the lexical and conceptual relatedness. We treat these documents as a training set to learn a text classifier. After training the classifier, the remaining

documents are classified. The main steps of this ensemble lexical-semantic double clustering (*ELSDC*) algorithm are shown in Algorithm 6 and Fig. 6.1. The following sections explain Phase 3 and Phase 4 of the ensemble algorithm in detail.

Finding Semantic Seed Documents

The input of this phase are k distilled term clusters. We extract semantic seed documents based on the concepts extracted for documents and for distilled term clusters. The idea is that the semantic seed documents share common concepts with the distilled term clusters. The semantic seed documents of term cluster TC_p are extracted using the following equation:

$$\begin{aligned} \text{semanticSeeds}(TC_p) = \{(d_i, w_i) | \quad w_i = |\text{wikify}(d_i) \cap \\ \text{wikify}(TC_p)| \quad \text{AND} \quad w_i > 0\} \end{aligned} \tag{6.1}$$

where w_i is the weight of seed document d_i. The input of wikify is a set of terms or a document's terms and its output is a set of Wikipedia concepts. The semantic centroid of TC_p is then computed using the following equation:

$$\text{semanticCentroid}_p = \frac{\sum_{d_i \in \text{semanticSeeds}(TC_p)} w_i * d_i}{|\text{semanticSeeds}(TC_p)|} \tag{6.2}$$

where d_i is the vector of the i^{th} document in BOW.

Consensus Method

The input of this phase are two document centroids for each term cluster, a lexical centroid and a semantic centroid. We cluster documents once based on the lexical centroids and once based on the semantic centroids using the method of Section 5.1.3. We then combine the generated clusters to form the final clusters.

The idea behind our aggregation is that if a document in both clusterings belongs to the same clusters, it is more likely that the document is clustered correctly. We find all these documents and treat them as a training set to train a classifier. After training is done, the remaining documents are classified. The algorithm of this phase has the following steps:

- Cluster documents using the lexical centroids

- Cluster documents using the semantic centroids
- Find the documents with the same clusters in both clusterings
- Treat these documents as a training set to train a classifier
- Classify the remaining documents

It is worth mentioning that two clusterings are generated based on the same set of term clusters (or their semantic mappings). So, there is a correspondence between the clusters of the two clusterings by the structure of our ensemble algorithm. In other words, if a document is assigned to the respective document clusters of a term cluster, both lexically and semantically, we say that the document sits in the same clusters. We used the Naive Bayes classifier available in the Matlab statistical toolbox as the text classifier in our experiments.

The idea behind this consensus module may fail to take effect if documents are similar based on bag of words model but dissimilar based on bag of concepts. For instance, document d_i is about Computer Networks and document d_j is about Social Networks. These two documents share some terms but their concepts may be different. If this case happens for many pairs of documents, there is no guarantee to have sufficient training data for the text classifier and *ELSDC* may fail.

6.2 Experimental Results

We show the benefits of integrating Wikipedia concepts in document clustering by comparing three algorithms *LDC*, Algorithm 4, *SDC*, Algorithm 5, and the proposed ensemble algorithm (*ELSDC*), Algorithm 6. Three non-stemmed datasets *Reuters8-whole*, *20ng-whole*, and *Classic4* are used in these experiments. Non-stemmed datasets are used since the Wikipedia articles are not stemmed and input of wikify methods are non-stemmed texts. We used the wikify method proposed in [84] to extract Wikipedia concepts[1].

We first show that *BOW* is much better than *BOC* in terms of clustering quality. For this purpose, we compared *LDC* to *SDC* using the document-term and the document-concept matrices. We ran the algorithms 50 times on each dataset. The

[1] `http://wikipedia-miner.cms.waikato.ac.nz/`

Algorithm 6 Ensemble Lexical-Semantic Double Clustering (*ELSDC*)

Input: *a document collection, k*

Output: *k document clusters* $\{W_p\}_{p=1}^{k}$

for *each document* d_i **do**
    extract relevant Wikipedia concepts
end for
use fuzzy c-means, Algorithm 1, to generate k term clusters $\{TC_p\}_{p=1}^{k}$
remove non-discriminative terms from the term clusters
for *each distilled term cluster* TC_p **do**
    extract relevant Wikipedia concepts
end for
for *each distilled term cluster* TC_p **do**
    find the seed documents based on the BOW model
    compute the lexical document centroids:

$$lexicalCentroid_p = \frac{\sum_{d_i \in lexicalSeeds(TC_p)} w_i * d_i}{|lexicalSeeds(TC_p)|}$$

    find the seed documents based on the BOC model
    compute the semantic document centroids:

$$semanticCentroid_p = \frac{\sum_{d_i \in semanticSeeds(TC_p)} w_i * d_i}{|semanticSeeds(TC_p)|}$$

end for
cluster the document collection using the lexical centroids
cluster the document collection using the semantic centroids
aggregate the clusterings by finding the documents with the same clusters in both clusterings
treat these documents as a training set and train a classifier
classify the remaining documents using the classifier

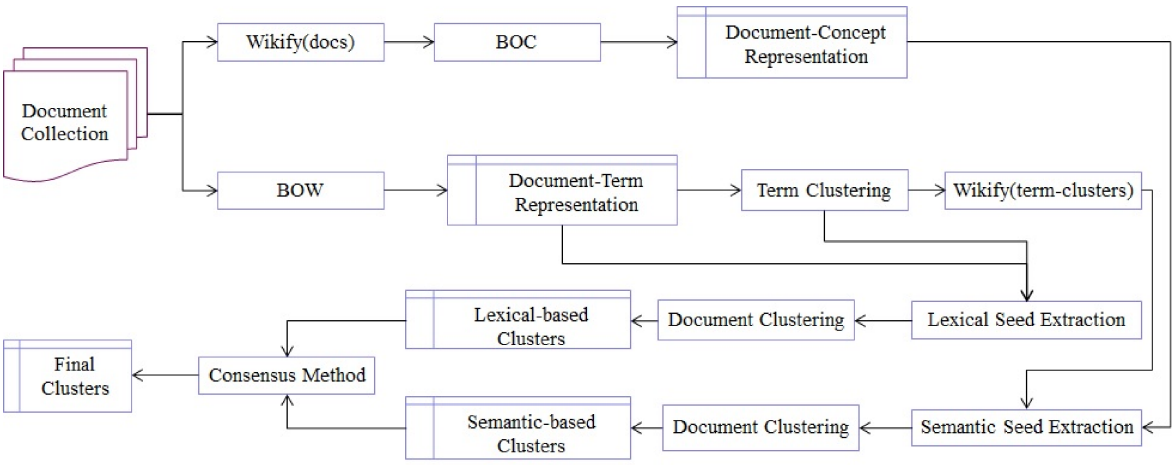

Figure 6.1: The structure of the ensemble lexical-semantic double clustering algorithm, Algorithm 6. *BOW* and *BOC* are used to represent documents. The wikify module extracts relevant concepts from Wikipedia. The consensus method consists of a Naive Bayes classifier, which is trained by using the documents that belong to the same clusters in both clusterings. The final clustering is generated by classifying the remaining documents. The wikify module proposed in [84] is used to extract Wikipedia concepts in this algorithm.

average *NMI*s and *Fmeasure*s of these 50 runs are shown in Table 6.1. Significant improvements according to paired-sample T-test with $p \leq 0.05$ are indicated by "*".

We then show that combining the *BOW* and *BOC* models in the ensemble algorithm (*ELSDC*) generates better results than using *BOW* alone. For this purpose, we compared *LDC* to the proposed *ELSDC* algorithm. We ran the algorithms 50 times on each dataset. The average *NMI*s and *Fmeasure*s of these 50 runs are shown in Table 6.2. Based on the empirical results obtained in this experiment, we observed that:

1. We cannot ignore document terms from the clustering process. The comparison between the *BOW* and *BOC* models reveals that the quality of clusters deteriorates significantly if document contents are replaced by Wikipedia concepts. The first cause originates from term polysemy. A term like "tree" has different meanings in different contexts. Finding the best sense of the disambiguated terms is still a challenging task in extracting semantic relatedness [84]. The second reason is that there are many discriminative terms in document contents that are not shown in the output of wikify modules. Eliminating those terms

Table 6.1: Comparing the *BOW* and *BOC* models by using the lexical document clustering (*LDC*) and the semantic document clustering (*SDC*) algorithms. Each algorithm is run 50 times and the average *NMI*s and *Fmeasure*s are shown. The first value in each cell is *Fmeasure* and the second one is *NMI*. The document representation in *BOW* generates significantly better results in our experiments. This observation is consistent with the results obtained in [52].

Datasets	LDC	SDC
20ng-whole	**0.6203 ± 0.0213*** **0.6328 ± 0.0105***	0.4315 ± 0.0176 0.4143 ± 0.0173
Reuters8-whole	**0.6505 ± 0.0498*** **0.5381 ± 0.0289***	0.5166 ± 0.0229 0.2663 ± 0.0282
Classic4	**0.9268 ± 0.0333*** **0.8145 ± 0.0302***	0.6729 ± 0.0466 0.4695 ± 0.0528

Table 6.2: Comparison between the proposed ensemble algorithm (*ELSDC*) and the lexical document clustering algorithm (*LDC*). *LDC* is based on the *BOW* model, while a consensus method is used in *ELSDC* to combine the results obtained from *BOW* and *BOC*. Each algorithm is run 50 times and the average *NMI*s and *Fmeasure*s are shown. The first value in each cell is *Fmeasure* and the second one is *NMI*. Integrating document concepts extracted from Wikipedia improves the quality of clusters significantly except on two datasets *20ng-whole* and *Classic4* based on *Fmeasure*.

Datasets	*LDC*	*ELSDC*
20ng-whole	0.6203 ± 0.0213 0.6328 ± 0.0105	**0.6301 ± 0.0448** **0.6695 ± 0.0223***
Reuters8-whole	0.6505 ± 0.0498 0.5381 ± 0.0289	**0.7213 ± 0.0417*** **0.5861 ± 0.0222***
Classic4	0.9268 ± 0.0333 0.8145 ± 0.0302	**0.9387 ± 0.0326** **0.8370 ± 0.0184***

was not compensated for by adding Wikipedia concepts in our experiments.

2. The proposed ensemble algorithm (*ELSDC*) has successfully integrated concepts in document clustering. Improvements have been obtained even though the *BOC* model alone resulted in poor clusters in our experiments.

Overall, we conclude that the *ELSDC* algorithm is an effective way to integrate Wikipedia concepts in our partitional clustering algorithm. Besides, no parameter setting is needed in *ELSDC*, compared to the algorithms reviewed in Section 2.4. The proposed consensus method can be used with any partitional clustering algorithm if term clustering is performed before document clustering.

6.3 Conclusion

We proposed a new framework for partitional document clustering to integrate Wikipedia concepts in the *BOW* model. Our framework consists of an ensemble algorithm which combines the clusterings generated from *BOW* and *BOC*. The documents with the same labels in the clusterings are used as a training set to train a text classifier. The trained classifier clusters the remaining documents.

We proposed a method in the framework to extract seed documents semantically. Semantic seed documents are extracted from term clusters and Wikipedia concepts. Documents that share Wikipedia concepts with term clusters are considered as semantic seeds.

Our experimental results demonstrate that the proposed ensemble algorithm can improve the quality of document clusters even if the clusters obtained from the document-concept representation alone are inferior to those obtained from the document-term representation.

The other conclusion is that we should not ignore the documents' terms in text document clustering.

As future work, our work can be extended to integrate other information like the Wikipedia categories or the semantic relatedness among Wikipedia articles.

Chapter 7

User-supervised Document Clustering

One advantage of the clustering algorithms presented in the previous chapters is that they can be easily adapted for interactive use. We propose three user-supervised versions of *LDC*, Algorithm 4, in this chapter:

1. Document-supervised *LDC*: A set of documents is selected randomly from the collection and the user is asked to label them, i.e. assign them to a set of classes. Given the labeled documents, keyterms of each class are extracted using the χ^2 statistic. The keyterms are then used to initialize the term clusterer, fuzzy c-means.

2. Term-supervised *LDC*: A document partitioning of the collection is first generated and the keyterms of each cluster are then extracted using the χ^2 statistic. The user is then asked to label the keyterms, i.e. assign them to a set of classes. The labeled keyterms are then used to initialize the term clusterer and re-run *LDC*.

3. Dual-supervised *LDC*: This algorithm is the combination of the Document and Term supervised algorithms. A set of documents is selected randomly from the collection and the user is asked to label them. Given the labeled documents, keyterms of each class are extracted using the χ^2 statistic. The keyterms are then used to initialize the term clusterer. After generating the document clusters, their top keyterms are extracted using the χ^2 statistic. The user is then asked to label the keyterms. The labeled keyterms are then used to initialize the term clusterer for the next round.

The block structure of unsupervised *LDC* is depicted in Fig. 7.1. No user supervision is involved in *LDC* and the term clusterer is initialized randomly. We use this block structure to present the user-supervised versions of *LDC*.

We have compared the proposed user-supervised algorithms so as to find out which type of supervision is most appropriate for *LDC*.

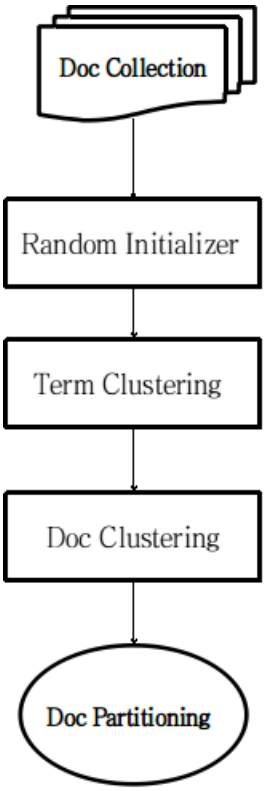

Figure 7.1: ***LDC***: The structure of the *LDC* algorithm. Fuzzy *c*-means is used for term clustering. The term clusters are then used to cluster documents. A greedy approach distills the term clusters in order to remove non-discriminative terms. Representative documents associated with each term cluster are then extracted and used as seeds to cluster all documents.

The remainder of this chapter is organized as follows. Section 7.1 explains the user-supervised clustering algorithms in detail. Experimental results on some real text datasets are reported in Section 7.2. This section includes the experiment performed to compare term labeling with a baseline term selection, and also the experiments performed to compare different supervision approaches proposed for *LDC*. Section 7.3 presents conclusions.

7.1 Methodology

In this section, we present three approaches to involve users in document clustering by using the *LDC* algorithm. The amount of user effort needed in each approach is also discussed.

7.1.1 Document-Supervised *LDC*

Document-supervised *LDC* is quite similar to the semi-supervised partitional clustering algorithms [9, 10], which ask the user to provide seed documents in advance. Semi-supervised clustering is usually performed in the following way. Before starting the clustering process, a set of documents is selected from the collection and the user groups similar documents and assigns a label to each group. Each document is assigned only to one group. The labeled documents are then used as seeds to initialize the clustering algorithm. For instance, the documents are used to generate initial centroids of k-means [9] or generate *"must-link"* or *"cannot-link"* pairwise constraints [10].

Labeled documents cannot be used directly in *LDC* since term clustering precedes document clustering as shown in Fig. 7.1. Hence, we need to use the labeled documents in a different way. For this purpose, we extract the keyterms of the labeled documents using the χ^2 statistic. The keyterms are used to initialize the term clusterer, fuzzy c-means, as shown in Fig. 7.2.

The membership of terms in term clusters are stored in a membership matrix in fuzzy c-means as explained in Section 4.1.1. The membership matrix has k columns corresponding to the k term clusters, and M rows corresponding to the M terms of the collection.

The matrix is initialized randomly in unsupervised *LDC*. In Document-supervised *LDC*, the keyterms extracted from the labeled documents are used to initialize the

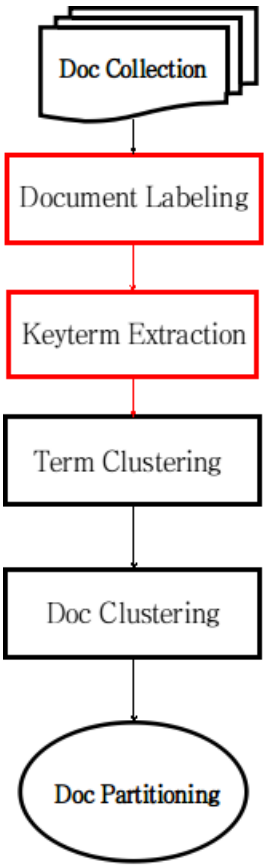

Figure 7.2: **Document-supervised *LDC***: The structure of Document-supervised *LDC*. The user provides some labeled documents. The respective keyterms of the labeled documents are then extracted using the χ^2 statistic. The keyterms are used to initialize the term clusterer. Term clustering and document clustering are then performed to generate a document partitioning.

matrix. For those keyterms, their corresponding entries of term cluster are set to one. The entries of the other terms are initialized randomly. The initialized membership matrix is then fed into fuzzy c-means to generate term clusters.

The user of this algorithm should read at least the title or the first few lines of the documents and should have enough background knowledge to organize them in meaningful clusters. The user effort thus includes 1) reading documents and 2) grouping similar documents into clusters. The interaction is limited to the initialization step in this algorithm. After labeling the documents, the user has no chance to interact with the clustering process. The main steps of Document-supervised *LDC* are shown in Algorithm 7.

Algorithm 7 Document-supervised *LDC*

Input: *a document-term matrix* M_{DT}, *k*

Output: *k document clusters* $\{W_p\}_{p=1}^{k}$

1: *generate a training set by randomly selecting documents*

2: *ask the user to group the training documents*

3: *extract the keyterms of the labeled training documents using the* χ^2 *statistic*

4: *initialize the term clusterer, fuzzy c-means, using the extracted keyterms*

5: *obtain k document clusters using LDC, Algorithm 4*

7.1.2 Term-Supervised *LDC*

The user supervision is in the form of term labeling in Term-supervised *LDC*. Term labeling starts after document clustering is done as shown in the main steps of Term-supervised *LDC* in Fig. 7.3. A hard partitioning of documents is first generated using unsupervised *LDC*. Treating document clusters as true classes, the χ^2 statistic is then used to extract top keyterms. For each document cluster, we sort the χ^2 values in descending order. The first f terms are then considered as the top keyterms of the document cluster.

A term cloud is subsequently created for each document cluster using its respective top keyterms. The term clouds are displayed to the user and she performs term labeling. Term labeling is in the form of modifying term clouds using the following options:

1. Remove a term from a term cloud.

2. Assign a term to another term cloud.

3. Assign a term to multiple term clouds.

It is also possible to merge two or more term clouds, split or even remove them. In this way, the user can specify the number of document clusters she prefers to generate.

After term labeling is done, we use the supervised term clouds to initialize a membership matrix for fuzzy c-means. The matrix is initialized randomly except for those keyterms that exist in the supervised term clouds. For those keyterms, only their corresponding entries of the term clouds are set to one. The initialized membership

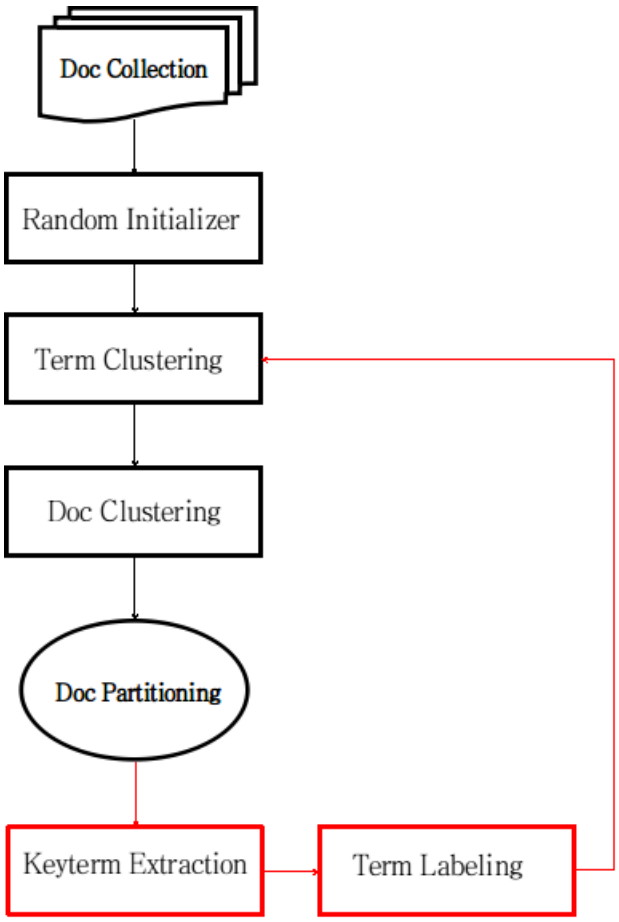

Figure 7.3: **Term-supervised *LDC***: The structure of Term-supervised *LDC*. *LDC* generates a document partitioning. The top keyterms of the current document clusters are then extracted and displayed to the user in the form of term clouds. The user performs term labeling on the term clouds. The supervised term clouds are subsequently used to initialize the term clusterer to re-cluster the terms.

matrix is then fed into fuzzy c-means to re-initialize *LDC* and obtain new document clusters.

We perform these interactions for a few iterations until the term clouds of document clusters satisfy the user. It is necessary to mention that the number of term clouds, term clusters, and document clusters is the same in user-supervised *LDC* algorithms.

The user of this algorithm can continue to interact until the end of the clustering process. The user effort includes only relocating terms among term clouds. She should have enough background knowledge to group terms in meaningful clouds. The main steps of Term-supervised *LDC* is shown in Algorithm 8.

Algorithm 8 Term-supervised *LDC*

Input: *a document-term matrix* M_{DT}, k

Output: k *document clusters* $\{W_p\}_{p=1}^{k}$

1: *obtain initial* k *document clusters using LDC, Algorithm 4*
2: **repeat**
3: **for** *each document cluster* **do**
4: *generate a term cloud using its* f *top terms based on the* χ^2 *values*
5: **end for**
6: *perform term labeling*
7: *use the supervised term clouds to re-initialize LDC and obtain* k *document clusters*
8: **until** *the maximum number of iterations is reached or the user chooses to terminate*

7.1.3 Dual-Supervised *LDC*

A combination of document and term supervisions is used in Dual-supervised *LDC*. Rather than using random initialization in term clustering, the user labels a subset of documents. Given the labeled documents, the χ^2 statistic is used to extract their keyterms. The keyterms are then used to initialize the term clusterer.

After generating a document partitioning, the top keyterms of document clusters are extracted. The user is then asked to label terms by relocating them among term clouds. The keyterms in supervised clouds are then used to initialize the term clusterer to re-cluster terms. The block structure of Dual-supervised *LDC* is shown in Fig. 7.4.

The user of this algorithm can continue to interact until the end of the clustering process. The user effort includes the user effort of Term-supervised *LDC* and the user effort of Document-supervised *LDC*. The user should have enough background knowledge to group terms into meaningful clouds and documents into meaningful clusters. The main steps of Dual-supervised *LDC* is shown in Algorithm 9.

The type of user effort for all these three user-supervised clustering algorithms is shown in Table. 7.1. Term labeling has the minimum effort since the user should only relocate terms among term clouds. Document labeling needs more effort since the user should read at least a few lines of training documents and cluster them properly. Dual supervision needs the most effort since the user should supervise terms and

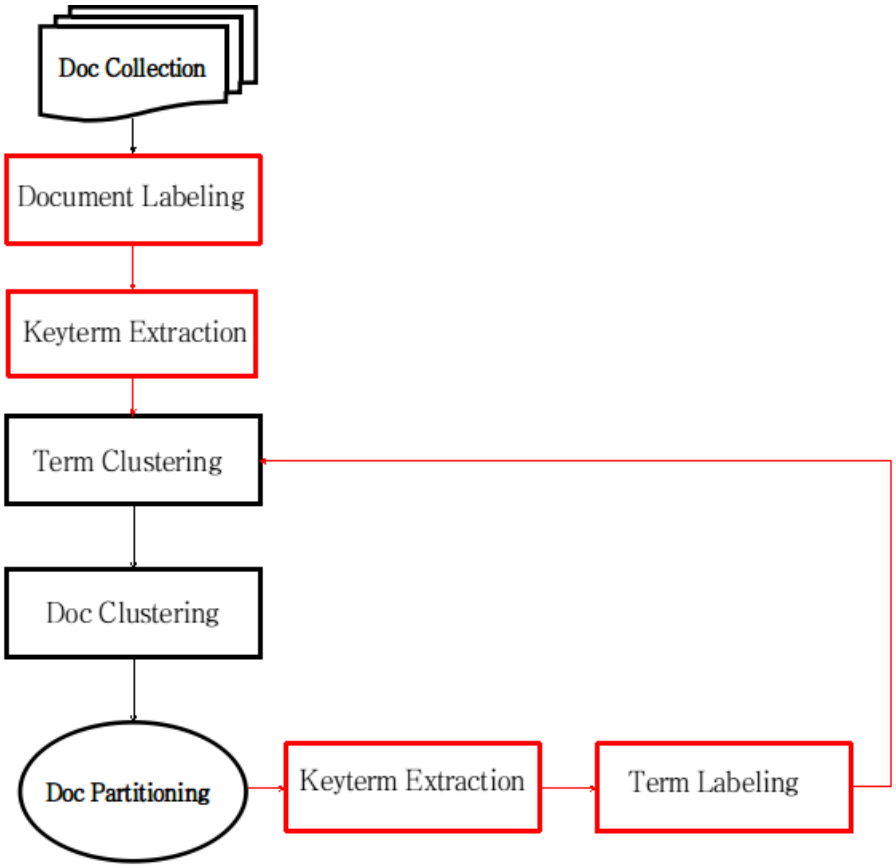

Figure 7.4: **Dual-supervised *LDC***: The structure of Dual-supervised *LDC*. The user provides some labeled documents. The respective keyterms of the labeled documents are then extracted using the χ^2 statistic. The keyterms are used to initialize the term clusterer. *LDC* generates a document partitioning. The top keyterms of the current document clusters are then extracted and displayed to the user in the form of term clouds. The user performs term labeling on the term clouds. The supervised term clouds are subsequently used to initialize the term clusterer to re-cluster the terms.

Algorithm 9 Dual-supervised *LDC*

Input: *a document-term matrix* M_{DT}, k

Output: k *document clusters* $\{W_p\}_{p=1}^{k}$

generate a training set by randomly selecting documents
ask the user to label the training documents
extract the keyterms of the labeled training documents using the χ^2 *statistic*
initialize the term clusterer, fuzzy c-means, using the extracted keyterms
obtain k document clusters using LDC, Algorithm 4
repeat
  for *each document cluster* **do**
    generate a term cloud using its f top terms based on the χ^2 *values*
  end for
  perform term labeling
  use the supervised term clouds to re-initialize LDC and obtain k document clusters
until *the maximum number of iterations is reached or the user chooses to terminate*

Table 7.1: The type of user effort needed in Term-supervised, Document-supervised, and Dual-supervised *LDC*

Algorithm	User Effort
Document-supervised *LDC*	Labeling documents: reading a few lines of each document + clustering them
Term-supervised *LDC*	Labeling terms: relocating terms among term clouds
Dual-supervised	Labeling documents + Labeling terms

documents.

7.1.4 Supervision Oracles

We used supervision oracles to simulate user interactions and evaluate our algorithms in this chapter. Although the user profiles considered in the oracles may be different from the real users' interactions, we can provide a comprehensive comparison among the algorithms with different parameter settings. Two supervision oracles are generated from class label of documents for this purpose:

1. Document Labeling oracle: This oracle knows the class label of documents.

2. Term Labeling oracle: This oracle knows the label of terms, which are computed from the class labels based on the χ^2 statistic.

Document Labeling Oracle

The input of the Document Labeling oracle is a document and the output is its class label. This is analogous to the case that the real user never makes any mistake in labeling the training documents. She put documents with the same class labels in same clusters. However, users may make mistake especially when the documents come from similar topics. To include these cases in simulated user interactions, the Document Labeling oracle has a parameter P_{exp} that indicates the degree of user's expertness. $P_{exp} = 1$ corresponds to the perfect user who never makes any mistake, and $P_{exp} = 0$ means no supervision is performed. Any value between zero and one indicates the level of user's knowledge about the dataset.

Based on this parameter, the simulated user either assigns a label to a document correctly or she says "*I do not know*". In the latter case, she might remove the document from the training set or assign it to a random cluster. The probability of removal is set to 0.5 and the probability of random assignment is set to $0.5/k$ in this case. The main steps of the Document Labeling oracle are shown in Algorithm 10.

Algorithm 10 Document Labeling Oracle

Input: *training documents*

Output: *training documents with labels*

```
for each document do
    if rand[0,1] < P_exp then
        assign the document to the true cluster
    else if rand[0,1] < 0.5 then
        remove the document from the training set
    else
        generate a random number between 1 and k
        assign the document to the random cluster
    end if
end for
```

The amount of user effort of the Document Labeling can be measured using the following formula:

$$\text{UserEffort} = \text{sizeOf(trainingSet)} * \text{time(labeling a document)} \tag{7.1}$$

Term Labeling Oracle

Term Labeling oracle knows the label of terms based on document class labels using the following process. Given the true class of documents, the χ^2 statistic is computed and each term is then assigned to the class with the largest χ^2 value. The main steps of this process are shown in Algorithm 11

Since users may make mistakes in supervision, this oracle has a parameter P_{exp} too that indicates the degree of user's expertness. $P_{exp} = 1$ corresponds to the perfect user who never makes any mistake, and $P_{exp} = 0$ means no supervision is performed. Any value between zero and one indicates the level of user's knowledge about the dataset. Based on this parameter the simulated user either assigns a term to a term cloud (cluster) correctly or she says "*I do not know*". In the latter case, she might remove the term from a cloud or assign it to a random cloud by mistake. The probability of removal is set to 0.5 and the probability of random assignment is set to $0.5/k$ in this case. The main steps of the term labeling oracle are shown in Algorithm 12.

Algorithm 11 Generating term class labels

Input: *A document collection, document true class labels*

Output: *term class labels*

1: **for** *each term* t_i **do**
2: *compute the* χ^2 *values using Eq.* (2.3)
3: **end for**
4: **for** *each term* t_i **do**
5: *assign* t_i *to the class with the largest* χ^2 *value:*
6: $\text{label}(t_i) \leftarrow \text{argmax}_c\ \chi^2(t_i, c)$
7: **end for**

The amount of user effort of Term Labeling can be measured using the following formula:

$$\text{UserEffort} = k * f * \text{time(labeling a term)} * (\text{numberOfIterations}) \tag{7.2}$$

Algorithm 12 Term Labeling Oracle

Input: *k term clouds*

Output: *k supervised term clouds*

```
1: for each term cloud do
2:    for each term in the term cloud do
3:       if rand[0,1] < P_exp then
4:          assign the term to the true cloud
5:       else if rand[0,1] < 0.5 then
6:          remove the term from the term cloud
7:       else
8:          generate a random number between 1 and k
9:          assign the term to the random term cloud
10:      end if
11:   end for
12: end for
```

where f is the number of terms in each term cloud before supervision, and k is the number of term clouds.

7.2 Experimental Results

In this section, we first show the benefits of term labeling as compared to a baseline user-supervised term selection method, which is inspired by the document clustering algorithms proposed in [53, 54]. We then perform some experiments to find out the best supervision approach for *LDC* in terms of the quality of clusters and the amount of user effort. We also compare Term-supervised *LDC* to some greedy direct clusterers, which neither use any term clustering nor any supervision in document clustering.

7.2.1 Term Labeling vs. Term Selection

To show the benefits of term labeling over term selection in document clustering, we compare two term-supervised document clustering algorithms. The main steps of the term labeling clusterer (*TLC*) and the term selection clusterer (*TSC*) are shown in Algorithm 13 and Algorithm 14, respectively.

Table 7.2: Definition of the variables used in the simulated term supervised algorithms

Variable	Definition
k	the number of document clusters the number of term clusters the number of term clouds the number of classes
f	the number of keyterms in each term cloud before user supervision
m	the size of feature set used for document clustering
g	the coefficient used for re-weighting terms in FeatureSet
P_{exp}	the degree of user expertness
B	oracle budget ($k*f$)

Term labeling clusterer, Algorithm 13, calls the Term Labeling oracle, Algorithm 12, in order to simulate user interactions. This oracle knows the true label of terms as mentioned above.

The baseline term selection clusterer, Algorithm 14, calls the Term Selection oracle, Algorithm 15, in order to simulate user interactions. In this oracle, the simulated user accepts a term as discriminative if the term was among the top m terms of the dataset [53, 54]. A reference list of the top m terms is generated a priori using the true classes of documents. The average of χ^2 values of each term in all document classes is computed for this purpose. The top m terms corresponding to the top average values form the reference list. The value of m is user-defined.

The user may make mistakes in specifying discriminative terms. She may accept a term while the term is not discriminative and vice versa. To simulate these mistakes in Algorithm 15, the Term Selection oracle randomly picks a term from the bottom half of T, which is considered as a noisy term, and adds it to the current *FeatureSet*.

It is worth mentioning that the number of queries submitted to both Term Labeling and Selection oracles are the same in our simulations. The definitions of all variables used in the simulations are shown in Table 7.2.

All terms are used for document clustering in the Term-supervised *LDC*, Fig. 7.3. However, only the best m terms are used in *TLC*, Algorithm 13. We considered this change to make *TLC* fairly comparable to *TSC*, Algorithm 14.

Algorithm 13 Term Labeling Clusterer (TLC)

Input: *a document-term matrix* M_{DT}*,* k*,* m

Output: k *document clusters* $\{W_p\}_{p=1}^{k}$

obtain initial k *document clusters using LDC, Algorithm 4 with the* m *best terms extracted by using the mean-TFIDF term Selection method, Eq.* (2.4)
repeat
  compute χ^2 *values for all terms using the current document clusters*
  sort all terms according to their average χ^2 *values in document clusters and obtain the ordered list* T
  for *each document cluster* **do**
    generate a term cloud using its f *top terms based on the* χ^2 *values*
  end for
  perform term labeling by using Algorithm 12
  use the term clouds to re-initialize LDC and obtain k *document clusters with the terms included in the term clouds and* m *- numberOfTerms(termClouds) best terms of* T
until *the maximum number of iterations is reached or the user chooses to terminate*

Algorithm 14 Term Selection Clusterer (TSC)

Input: *a document-term matrix* M_{DT}*,* k*,* m

Output: k *document clusters* $\{W_p\}_{p=1}^{k}$

obtain initial k *document clusters using* k*-means with* m *best terms extracted by using the mean-TFIDF term selection method*
repeat
  compute χ^2 *values for all terms using the current document clusters*
  sort all terms according to their average χ^2 *values in document clusters and obtain the ordered list* T
  perform term selection using Algorithm 15 and the ordered list $T(1\colon B)$
  perform term re-weighting using Algorithm 16
  obtain k *document clusters using* k*-means with the terms of the FeatureSet and* m *- sizeOf(FeatureSet) best terms of* T
until *the maximum number of iterations is reached or the user chooses to terminate*

Algorithm 15 Term Selection Oracle

Input: *an ordered list T*

Output: *FeatureSet*

FeatureSet = []
for *each term in T* **do**
    present the term to the user and get reply
    if *rand[0,1]* $< P_{exp}$ **then**
        if *the user accepts the term* **then**
            add the term to the current FeatureSet
        end if
    else if *rand[0,1]* < 0.5 **then**
        randomly pick a term from the bottom half of T and add it to the current FeatureSet
    end if
end for

Algorithm 16 Term Re-weighting

Input: *a document-term matrix, FeatureSet, g*

Output: *a re-weighted document-term matrix*

for *each term in the FeatureSet* **do**
    multiply its corresponding term vector by g
end for
normalize all the document vectors using L2 norm

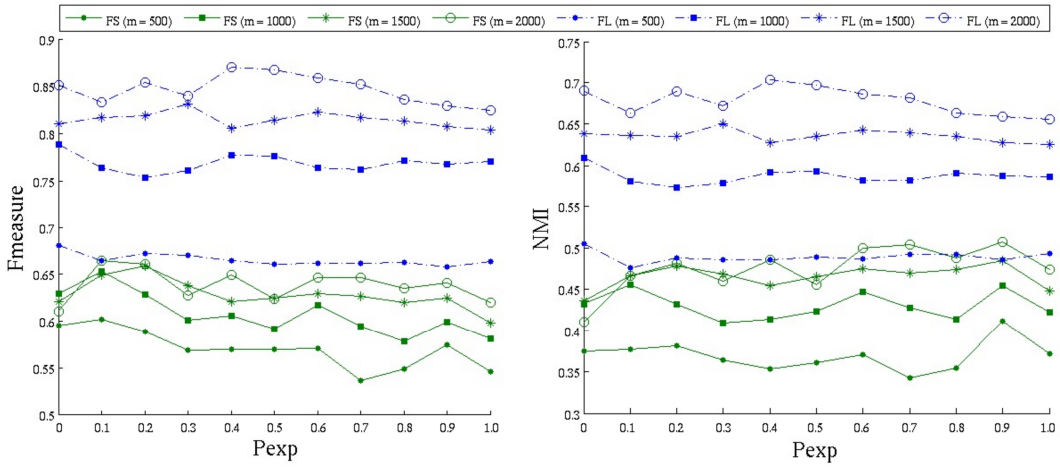

Figure 7.5: The quality of clusters based on feature (term) labeling (*FL*) and feature (term) selection (*FS*) on *Classic4*. Neither term labeling nor term selection could improve the quality of clusters significantly. This is because classes of *Classic4* are well separated [56] and term selection and labeling are not able to improve the quality of clusterings further.

Results and Discussion

We evaluated the performance of two term-supervised clusterers, *TLC* and *TSC*, on six datasets using two evaluation measures. For each dataset, we ran the algorithms in the following way:

- Each term-supervised algorithm is run 50 times for each degree of expertness $P_{exp} = \{0, 0.1, 0.2, ..., 1.0\}$ and each size of feature set $m = \{500, 1000, 1500, 2000\}$.

- The average *Fmeasure*s and *NMI*s of these 50 runs are depicted in Fig. 7.5 to Fig. 7.8 and in Fig. C.1 to Fig. C.2. Standard deviations of the evaluation measures are not shown to avoid clutter.

- The number of terms in each term cloud before supervision, f, is set to 20. Only two iterations of term supervision are considered in our experiments. The term re-weighting coefficient in Algorithm 16, g, is set to 10.

The experimental results show that our term labeling outperformed the baseline term selection method in most cases. This is more evident when the topics of document clusters are similar, specifically in *News-sim3* and *News-rel3*.

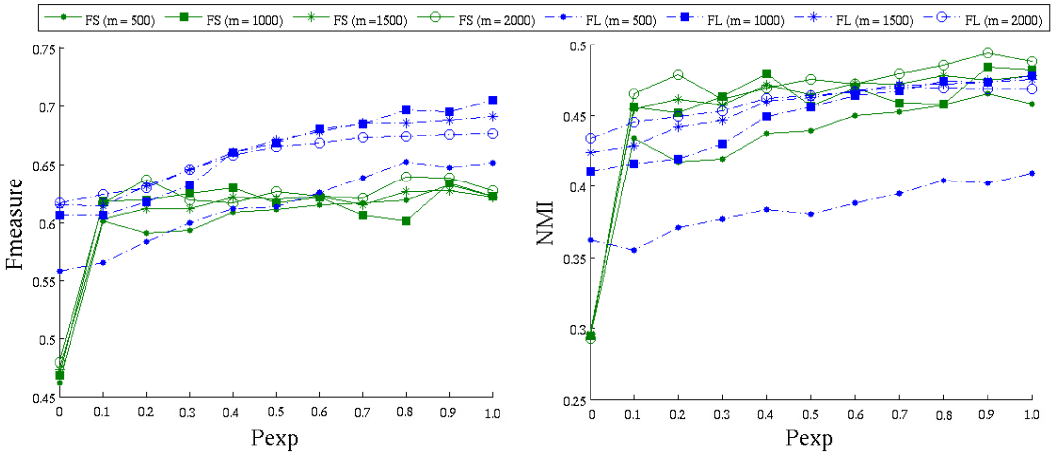

Figure 7.6: The quality of clusters based on term labeling (*FL*) and term selection (*FS*) on *LA Times*. Both term labeling and term selection methods improved the quality of clusters. The quality of the clusterings obtained by the algorithms is similar based on *NMI*.

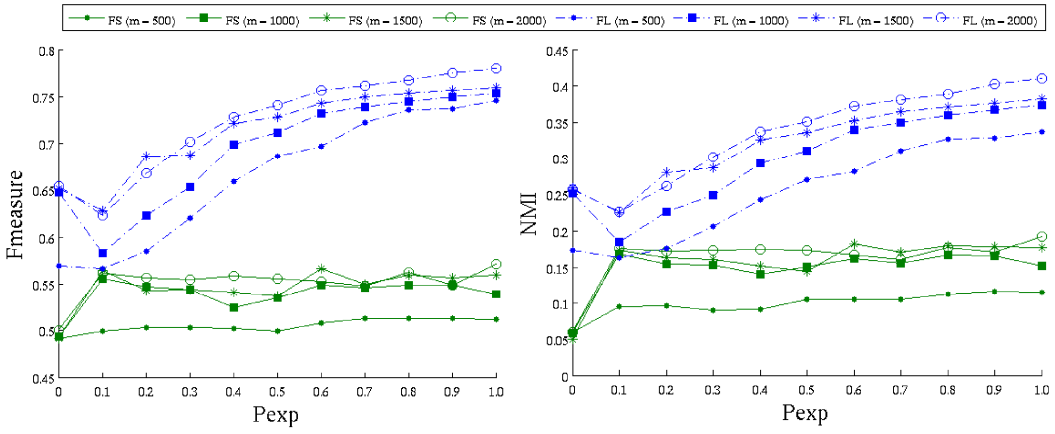

Figure 7.7: The quality of clusters based on term labeling (*FL*) and term selection (*FS*) on *News-sim3*. Term labeling algorithm significantly outperformed the term selection algorithm. As the degree of expertness increases, the quality of clusters obtained by term labeling clusterer improves more.

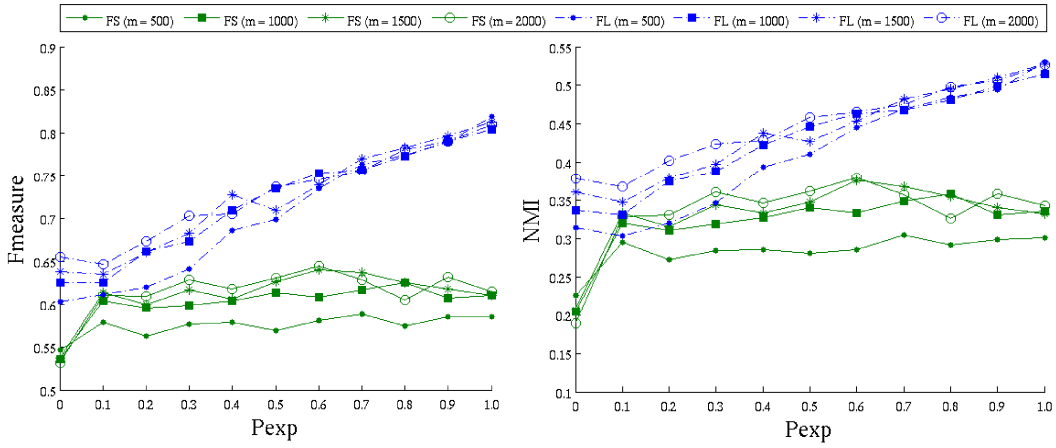

Figure 7.8: The quality of clusters based on term labeling (*FL*) and term selection (*FS*) on *News-rel3*. Feature labeling is much more effective than the term selection method in improving the quality of clusterings. Unlike term labeling, there is not much improvement in the results of term selection.

Except for *Classic4* where neither term supervision methods could improve the quality of clusters, our term labeling method generates much better clusters as the user expertness increases.

The experiments also reveal that the *LDC* algorithm outperformed k-means in all cases. This fact can be observed in Fig. 7.5 to Fig. 7.8 when no user supervision is involved ($P_{exp} = 0$).

As the size of term set, m, increases from 500 to 2000, the quality of clusters mostly increases for both algorithms, regardless of the degree of user expertness. This observation indicates that it is more useful to focus on term labeling than term selection and use all the terms that exist in a dataset in our experiments.

Another observation in the plots of Fig. 7.5 to Fig. 7.8 is that term labelling sometimes reduces the quality of the clusters, compared to the unsupervised mode, when P_{exp} is as low as 0.1. This is because making almost random replacements (P_{exp} near zero) may tend to deteriorate the quality of term clusters by forming a bad initialization for fuzzy c-means.

7.2.2 Term-Supervised LDC vs. Document-Supervised LDC

In this experiment, we compare Term-supervised *LDC*, Algorithm 8, to Document-supervised *LDC*, Algorithm 7. The goal of this experiment is to find out which type of

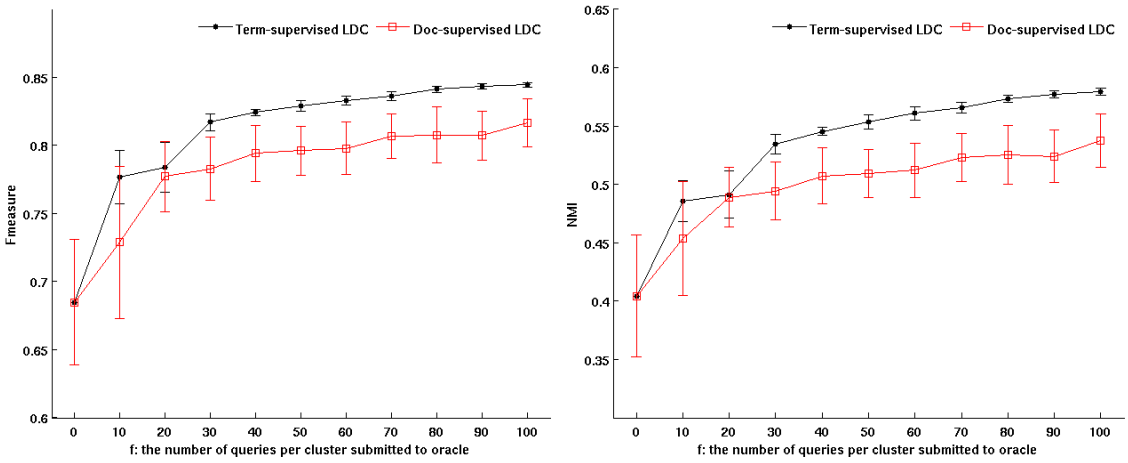

Figure 7.9: The quality of clusters obtained from Term-supervised *LDC* and Document-supervised *LDC* on *News-rel3*. Term-supervised *LDC* significantly outperforms Document-supervised *LDC* when f is 30 or more.

supervision improves the quality of clusters better, document labeling or term labeling.

To simulate user interactions, Term-supervised *LDC* calls the Term Labeling oracle, Algorithm 12, and Document-supervised *LDC* calls the Document Labeling oracle, Algorithm 10. We also assume that the user of this experiment has comprehensive knowledge about the datasets, $P_{exp} = 1.0$.

Results and Discussion

Five datasets are used in this experiment. For each dataset, we ran the user-supervised algorithms in the following way:

1. Each algorithm is run 50 times for each value of $f = \{0, 10, 20, ..., 100\}$ and the average *Fmeasure*s and *NMI*s are depicted in Fig. 7.10 to Fig. 7.13. The run with $f = 0$ corresponds to the case when *LDC* algorithm is run with no supervision. The size of training set is $k * f$ in Document-Supervised *LDC*.

2. The standard deviations are computed for each value of f in 50 runs and are shown as error bars in the plots.

3. Only one round of supervision is used for Term-supervised *LDC* to perform a fair comparison.

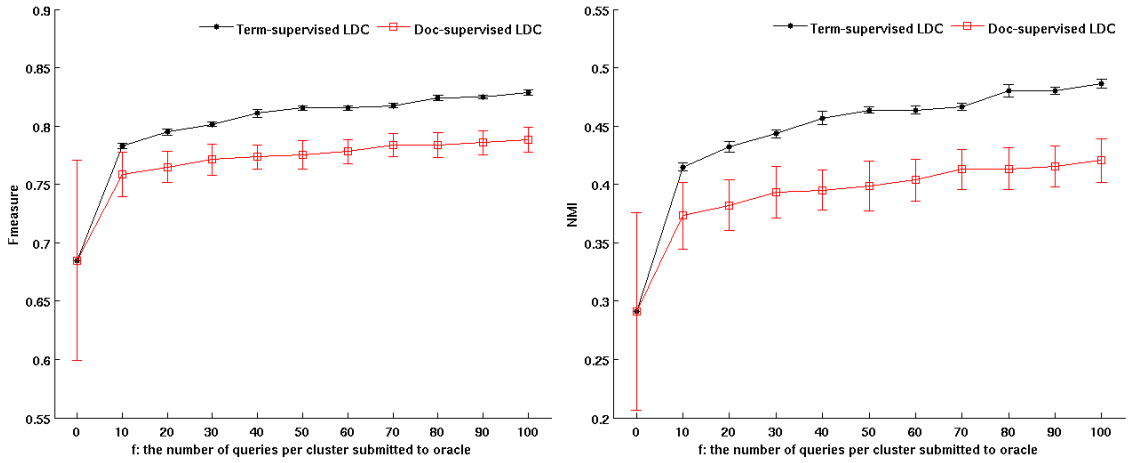

Figure 7.10: The quality of clusters obtained from Term-supervised *LDC* and Document-supervised *LDC* on *News-sim3*. Term-supervised *LDC* significantly outperforms Document-supervised *LDC*.

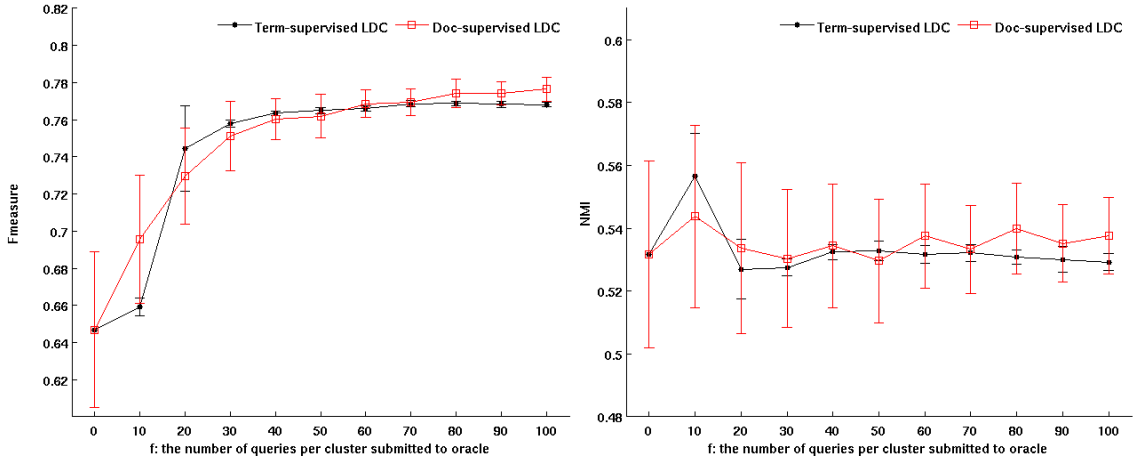

Figure 7.11: The quality of clusters obtained from Term-supervised *LDC* and Document-supervised *LDC* on *Reuters8-whole*. The quality of clusters are similar and neither of algorithms could outperform significantly the other one. The error bars of Term-supervised *LDC* are smaller and it shows that this algorithm is more reliable.

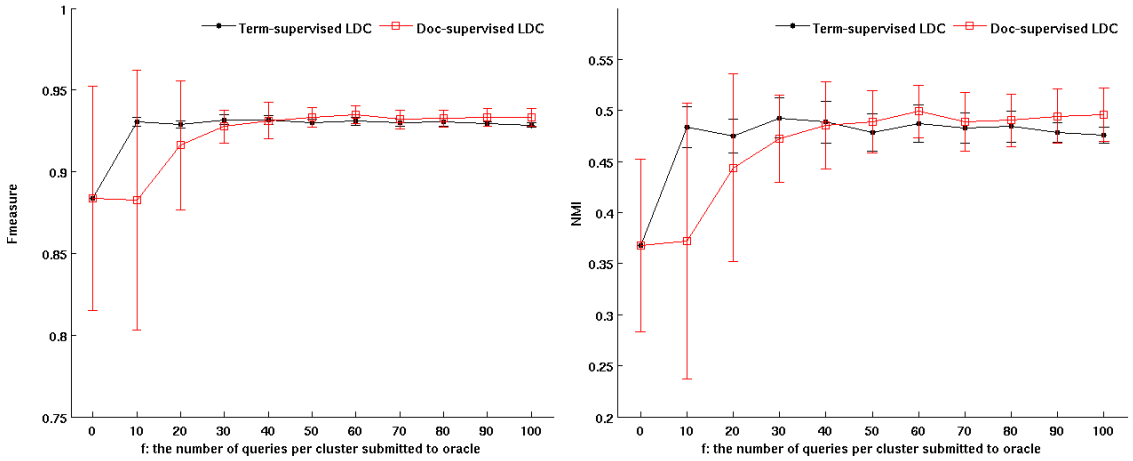

Figure 7.12: The quality of clusters obtained from Term-supervised *LDC* and Document-supervised *LDC* on *SMS*. Neither of algorithms could outperform significantly the other one. Term-supervised *LDC* is more reliable.

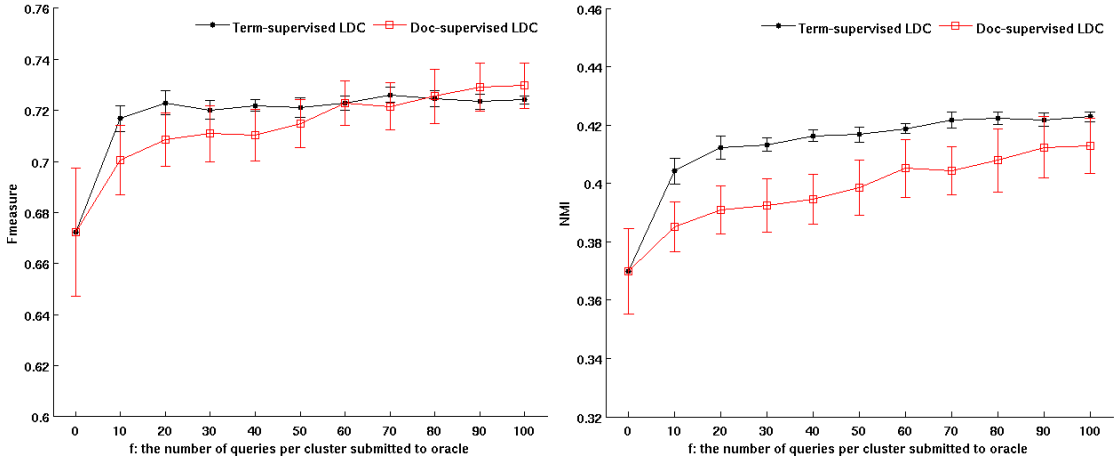

Figure 7.13: The quality of clusters obtained from Term-supervised *LDC* and Document-supervised *LDC* on *WebKB*. Term-supervised *LDC* could outperform significantly based on *NMI* measure for some values of f.

The main observation in this experiment is that Document-supervised *LDC* could never significantly outperform Term-supervised *LDC*. However, Term-supervised *LDC* significantly outperformed Document-supervised *LDC* in most cases. On the other hand, Term-supervised *LDC* was more reliable in this experiment by having smaller standard deviation in quality of clusters. Moreover, labeling a document basically needs more time and user effort than labeling a term. We can thus conclude that the term supervision is a better interaction approach than the document supervision for *LDC*.

7.2.3 Term-Supervised LDC vs. Dual-Supervised LDC

In this experiment, we compare Term-supervised *LDC*, Algorithm 8, to Dual-supervised *LDC*, Algorithm 9. The goal of this experiment is to find out whether adding document supervision to Term-supervised *LDC* would improve the quality of clusters further.

To simulate user interactions, Dual-supervised *LDC* calls the Document Labeling and Term Labeling oracles, Algorithm 10 and Algorithm 12, and Term-supervised *LDC* calls the Term Labeling oracle. We also assume that the user of this experiment has comprehensive knowledge about the datasets, $P_{exp} = 1.0$.

Results and Discussion

Five datasets are used in this experiment. For each dataset, we ran the user-supervised algorithms in the following way:

1. Each algorithm is run 50 times for each value of $f = \{0, 10, 20, ..., 100\}$ and the average *Fmeasure*s and *NMI*s are depicted in Fig. 7.14 to Fig. 7.16 and in Fig. D.1 to Fig. D.2. The run with $f = 0$ corresponds to the case when *LDC* algorithm is run with no supervision.

2. The standard deviations are computed for each value of f in 50 runs and are shown as error bars in the plots.

3. Only one round of supervision is used for term supervision.

The main observation of this experiment is that no significant improvement is obtained after adding document supervision to Term-supervised *LDC*.

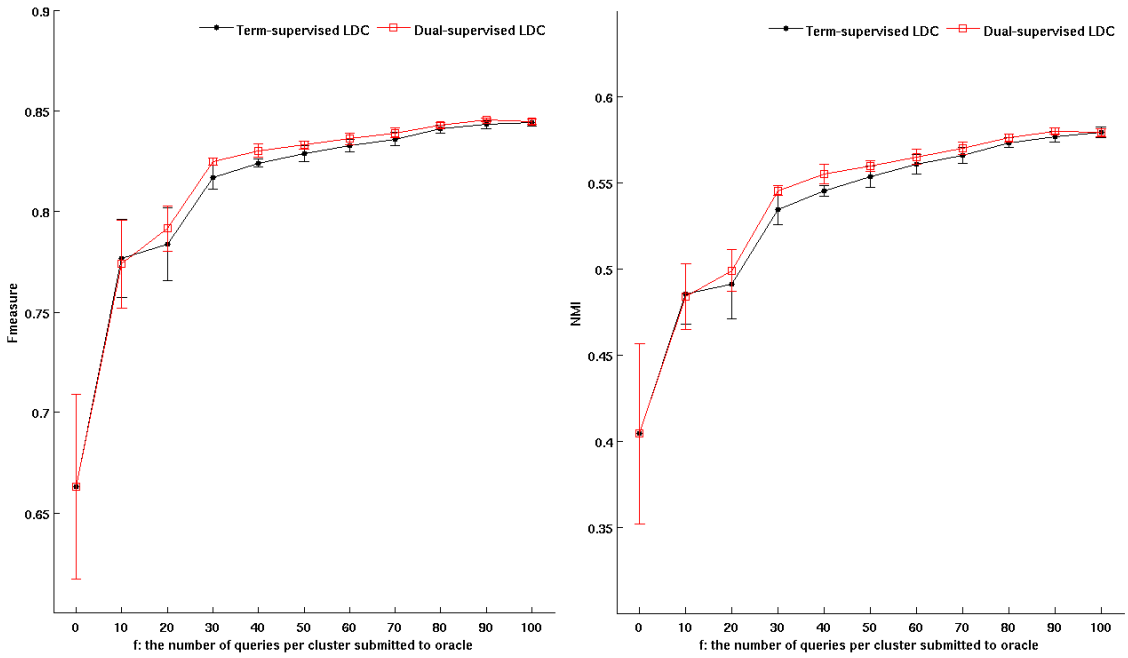

Figure 7.14: The quality of clusters obtained from Term-supervised *LDC* and Dual-supervised *LDC* on *News-rel3*. No significant improvement is obtained after adding document supervision to Term-supervised *LDC*.

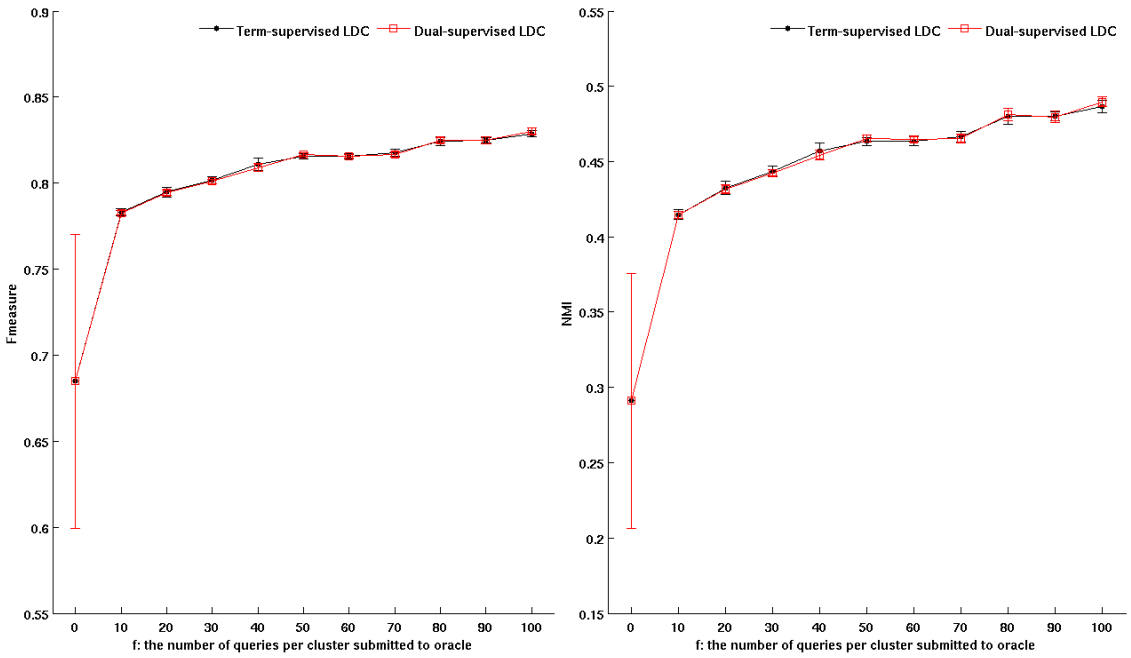

Figure 7.15: The quality of clusters obtained from Term-supervised *LDC* and Dual-supervised *LDC* on *News-sim3*. Dual-supervised *LDC* never outperformed Term-supervised *LDC*.

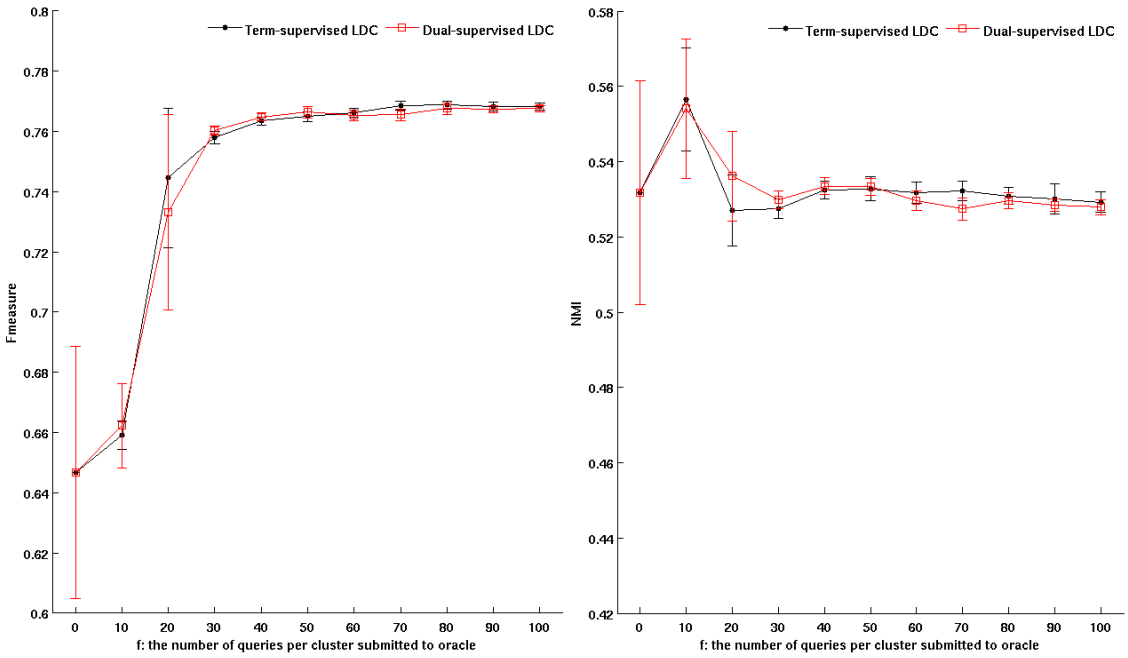

Figure 7.16: The quality of clusters obtained from Term-supervised *LDC* and Dual-supervised *LDC* on *Reuters8-whole*. The quality of clusters are almost similar and neither of algorithms could outperform significantly the other one.

On the other hand, dual supervision clearly needs more time and effort than term supervision. We can thus conclude that term supervision is sufficient for *LDC* and no significant improvement can be obtained after adding document supervision.

7.2.4 Noisy-Supervised LDC

The goal of this experiment is to show the benefit of different user supervision approaches when the simulated user makes mistakes. The underlying clustering algorithm is *LDC* and we compare document-supervised, term-supervised, and dual-supervised versions of this algorithm in this experiment.

To simulate user interactions, Dual-supervised *LDC*, Algorithm 9, calls the Document Labeling and Term Labeling oracles, Algorithm 10 and Algorithm 12, Term-supervised *LDC*, Algorithm 8, calls the Term Labeling oracle, and Document-supervised *LDC*, Algorithm 7, calls the Document Labeling oracle. We assume that the number of queries per cluster submitted to oracles, f is fixed to 20.

Results and Discussion

Five datasets are used in this experiment. For each dataset, we ran the user-supervised algorithms in the following way:

1. Each algorithm is run 50 times for each value of $P_{exp} = \{0, 0.1, 0.2, ..., 1.0\}$ and the average *Fmeasure*s and *NMI*s are depicted in Fig. 7.17 to Fig. 7.19 and in Fig. E.1 to Fig. E.2. The run with $P_{exp} = 0$ corresponds to the case when *LDC* algorithm is run without supervision.

2. Only one round of supervision is used in this experiment.

The first observation of this experiment is that Term-supervised *LDC* and Dual-supervised *LDC* generate similar results independent of the degree of user expertness. It means that document labeling could not improve the performance of term labeling for *LDC* in this experiment.

The second observation is that the quality of clusterings, obtained from Document-supervised *LDC*, is either similar to Term-supervised *LDC* and Dual-supervised *LDC* or it is inferior to theirs. We believe that Document-supervised *LDC* cannot improve

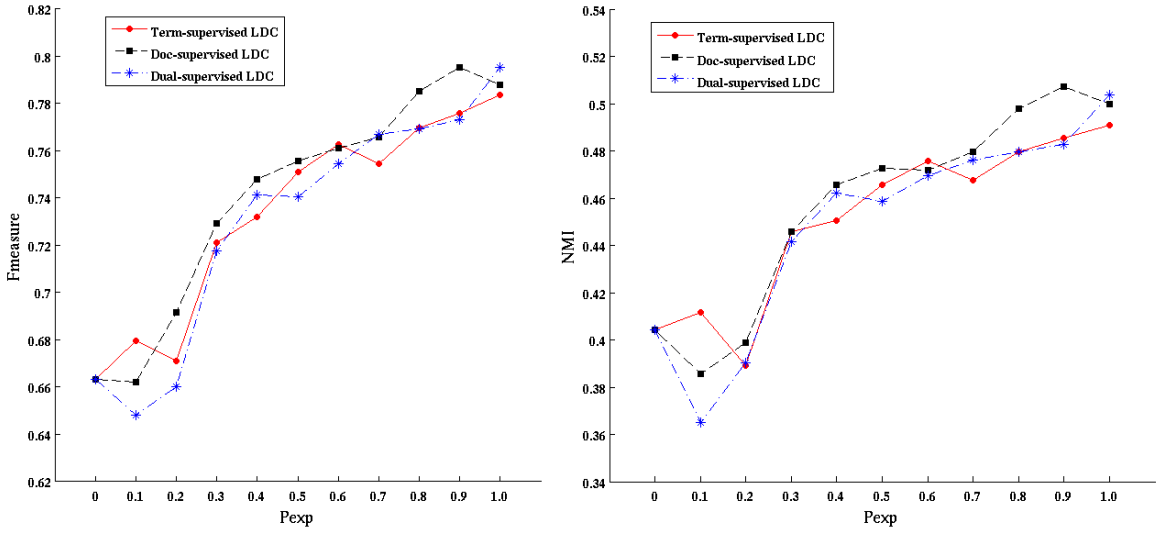

Figure 7.17: The quality of clusters obtained from the user-supervised *LDC* algorithms on *News-rel3* when the user's degree of expertness is variable. User-supervised algorithms generate similar results.

the quality of clusterings in case of user's noisy feedback as Term-supervised *LDC* can do.

Overall, we conclude that Term labeling is more effective than document labeling in our experiments if we consider user efforts as well. Document labeling is most effective when the user's feedback is noiseless.

7.2.5 Document-Supervised LDC vs. Seeded K-means

The goal of this experiment is to show the benefit of Document-supervised *LDC* over a semi-supervised clustering algorithm. The closest semi-supervised algorithm to the Document-supervised *LDC* is Seeded k-means [9], reviewed in Section 2.5.1. We compare this algorithm to *LDC* based on document labeling approach.

The user supervision is based on document labeling in both algorithms. A subset of documents is first randomly selected and the user is asked to label them. The labeled training documents are then used to generate initial centroids of k-means in Seeded k-means. After the initialization, the user has no interaction with the clustering process like in Document-supervised *LDC*. We also assume that the user

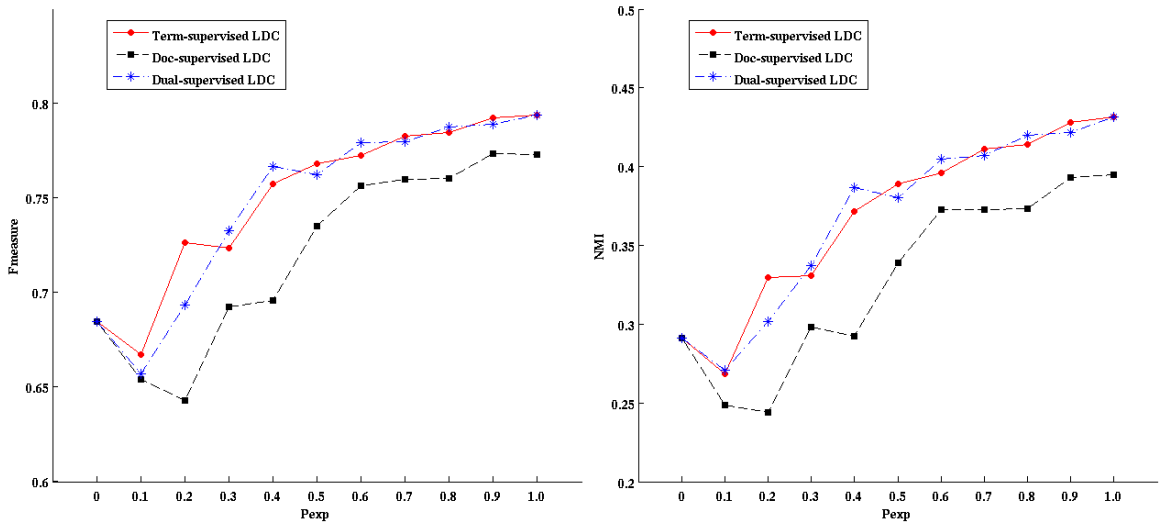

Figure 7.18: The quality of clusters obtained from the user-supervised *LDC* algorithms on *News-sim3* when the user's degree of expertness is variable. Term-supervised and Dual-supervised *LDC* outperform Document-supervised *LDC* regardless of the degree of user expertness.

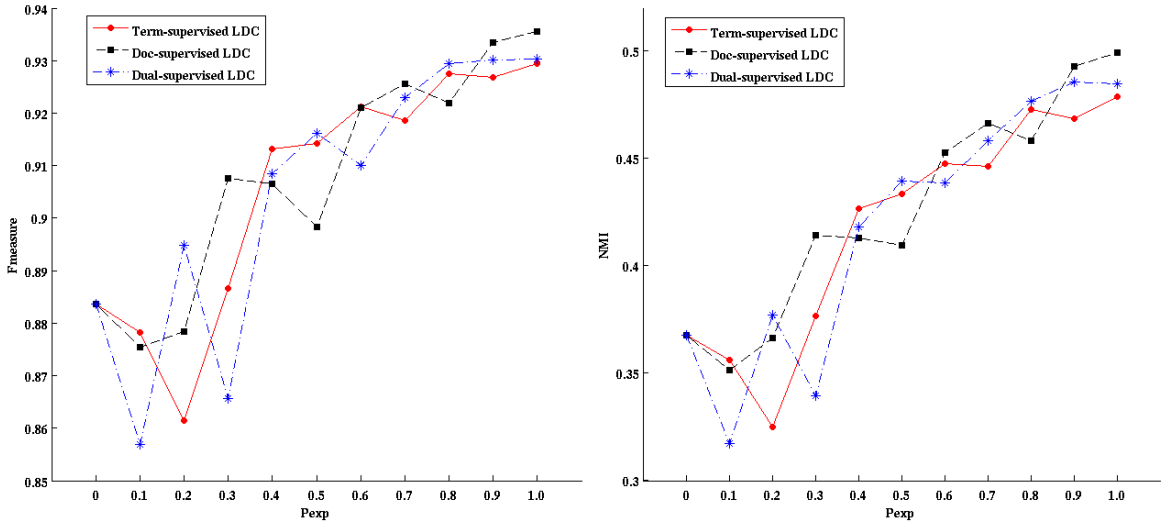

Figure 7.19: The quality of clusters obtained from the user-supervised *LDC* algorithms on *SMS* when the user's degree of expertness is variable. User-supervised algorithms generate similar results.

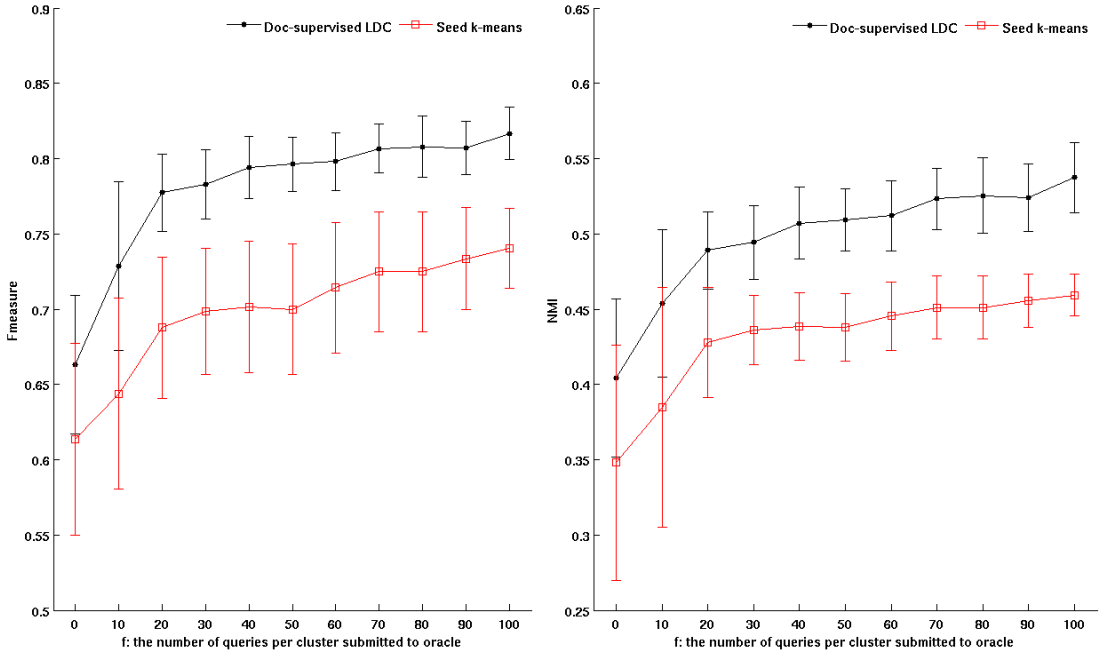

Figure 7.20: The quality of clusters obtained from Seeded k-means and Document-supervised *LDC* on *News-rel3*. Document-supervised *LDC* significantly outperforms Seeded k-means when f is 20 or more. In case of no supervision, the average quality of clusters obtained by *LDC* is better than k-means.

of this experiment has comprehensive knowledge about the datasets, $P_{exp} = 1.0$. In other words, the simulate user never makes mistake in labeling documents.

Results and Discussion

Five datasets are used in this experiment. For each dataset, we ran the algorithms in the following way:

1. Algorithm are run 50 times for each value of $f = \{0, 10, 20, ..., 100\}$ and the average *Fmeaure*s and *NMI*s are depicted in Fig. 7.20 to Fig. 7.24. The run with $f = 0$ corresponds to the case when no supervision is used.

2. The standard deviations are computed for each value of f in 50 runs and are shown as error bars in the plots.

The main observation of this experiment is that Document-supervised *LDC* outperformed Seeded k-means on all datasets. *LDC* has time complexity of $O(NMK^2I)$

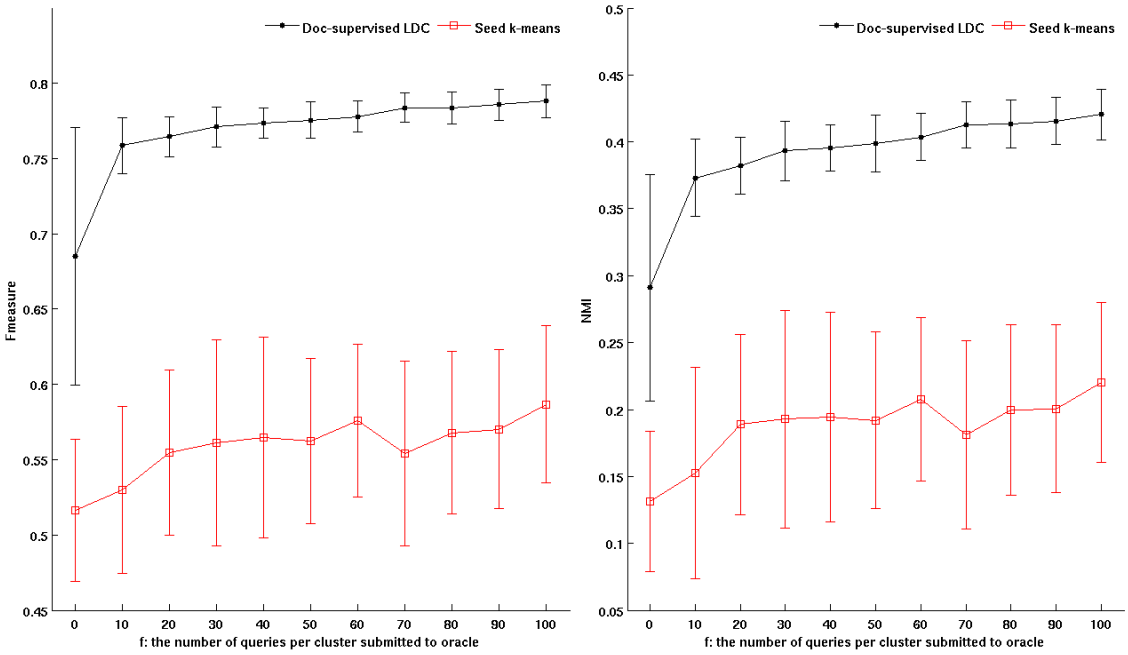

Figure 7.21: The quality of clusters obtained from Seeded k-means and Document-supervised *LDC* on *News-sim3*. Document-supervised *LDC* outperforms Seeded k-means significantly. *LDC* significantly outperforms k-means as well.

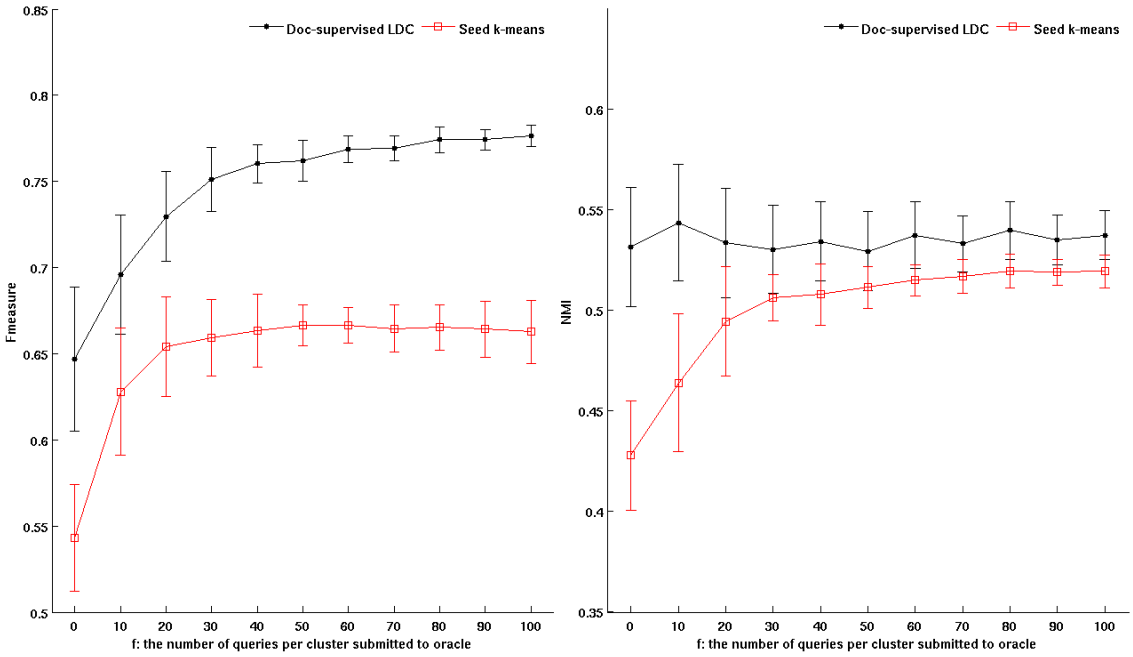

Figure 7.22: The quality of clusters obtained from Seeded k-means and Document-supervised *LDC* on *Reuters8-whole*. Document-supervised *LDC* is significantly better than Seeded k-means in most cases.

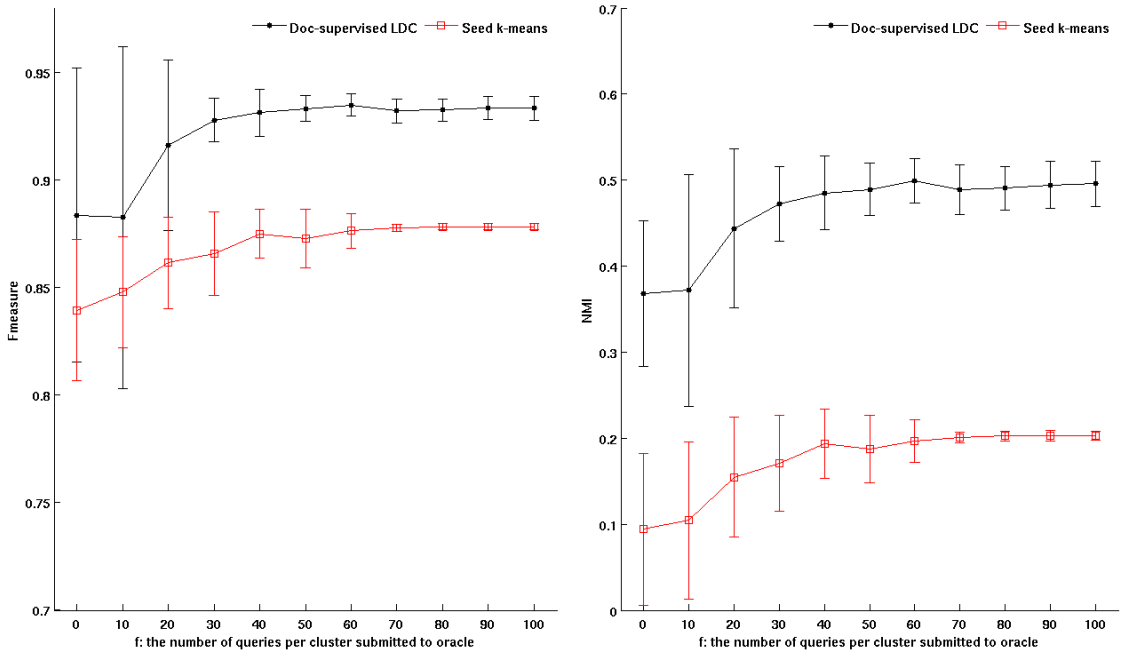

Figure 7.23: The quality of clusters obtained from Seeded k-means and Document-supervised *LDC* on *SMS*. Based on *NMI*, Document-supervised *LDC* outperforms Seeded k-means significantly. Based on *Fmeasure*, Document-supervised *LDC* outperforms Seeded k-means significantly after $f = 20$.

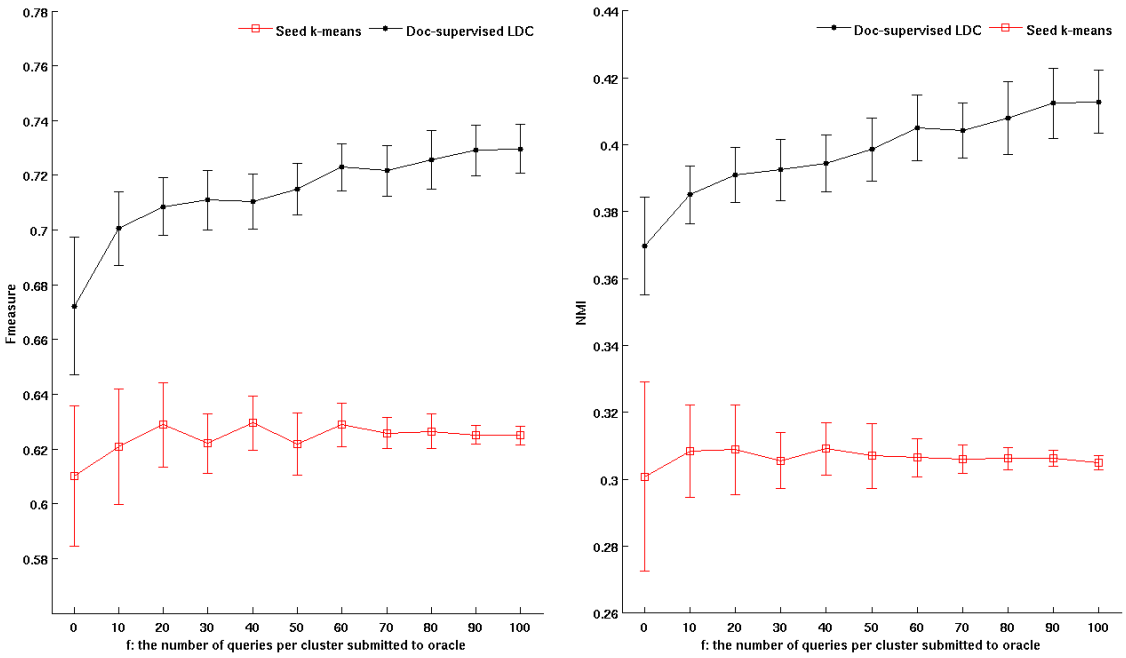

Figure 7.24: The quality of clusters obtained from Seeded k-means and Document-supervised *LDC* on *WebKB*. *LDC* and its document-supervised version significantly outperform k-means and Seeded k-means, respectively.

and k-means has the time complexity of $O(NMKI)$. Since the number of clusters, K, is much smaller than N and M, and the number of iterations I is a constant in experiments, we can conclude that Document-supervised *LDC* is a better semi-supervised algorithm than Seeded k-means in this experiment.

In the final experiment of this chapter, we compare Term-supervised *LDC* to some direct unsupervised clustering algorithms. No supervision is used in the direct clusterers and document clustering is performed without term clustering.

7.2.6 Term-Supervised LDC vs. Unsupervised Direct Clustering

In this section we compare Term-supervised *LDC*, Algorithm 8, to four unsupervised direct clustering algorithms. Term labeling is simulated using the Term Labeling oracle, Algorithm 12. We used four unsupervised clustering algorithms from [118] in this experiment, including Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*). The objective functions of these algorithms *I2, H2* are introduced in Section 2.3.2.

Results and Discussion

Five datasets are used in this experiment. For each dataset, we ran the algorithms in the following way:

1. Each unsupervised algorithm is run 50 times and the average *Fmeasure*s are shown as horizontal lines in Fig. 7.25 to Fig. 7.32.

2. Term-supervised *LDC* is also run 50 times for each value of $P_{exp} = \{0, 0.1, ..., 1.0\}$ and the average *Fmeasure*s are depicted in the plots.

3. The standard deviations are also shown as error bars. The standard deviation of Term-supervised *LDC* is computed for each value of P_{exp} in 50 runs.

4. Only two rounds of supervision used in this experiment.

The main observations of this experiment include the following:

1. *LDC* significantly outperformed the direct clusterers even without term labeling on datasets *Classic4*, *SMS*, and *Reuters8-whole*.

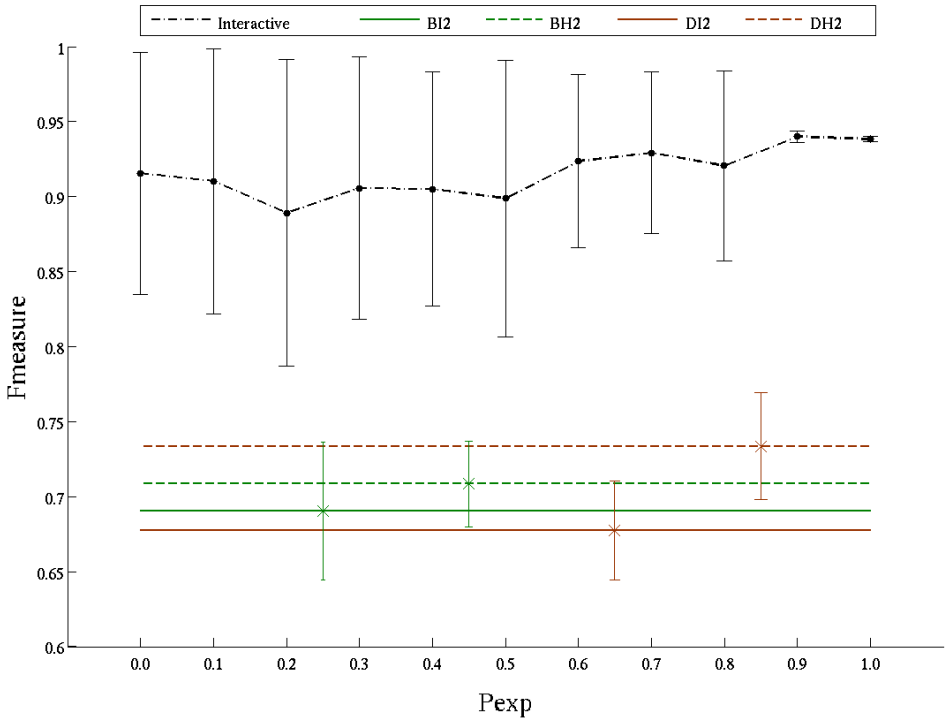

Figure 7.25: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *Classic4*. *LDC* significantly outperforms the direct clusterers in unsupervised mode ($P_{exp} = 0$) and also with term labeling. Term labeling cannot improve the quality of clusters since classes of *Classic4* are well separated. Only 10 terms are used for supervision ($f = 10$).

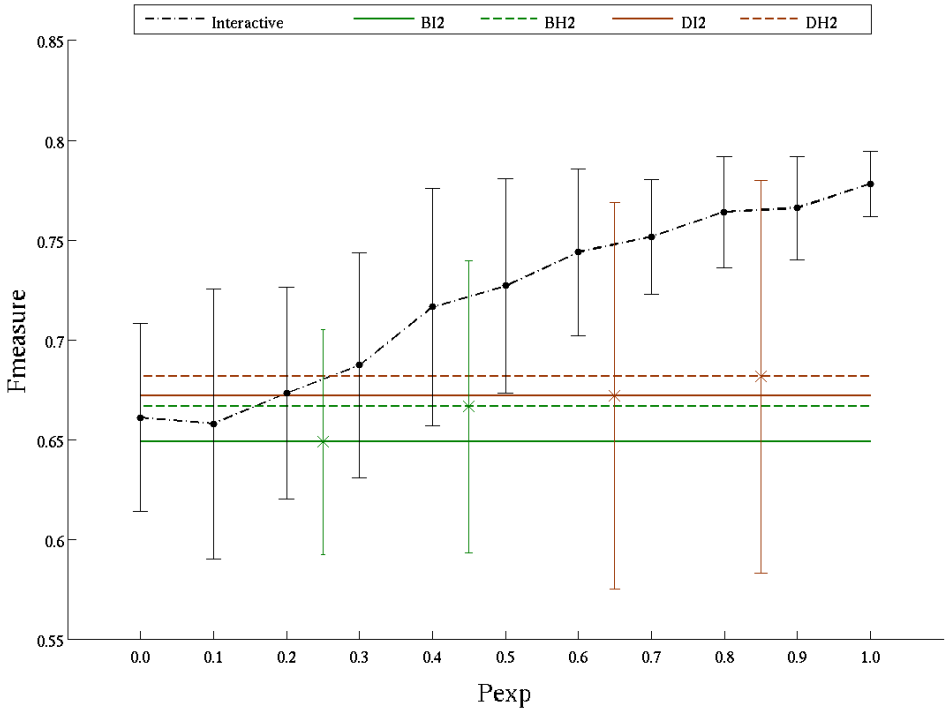

Figure 7.26: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *News-rel3*. *LDC* generates similar results to the direct clusterers in unsupervised mode ($P_{exp} = 0$). Once the level of user's expertness reaches 0.3, *LDC* outperforms the direct clusterers. The standard deviations of Term-supervised *LDC* decreases as the level of user's expertness increases. Only 10 terms are used for supervision ($f = 10$).

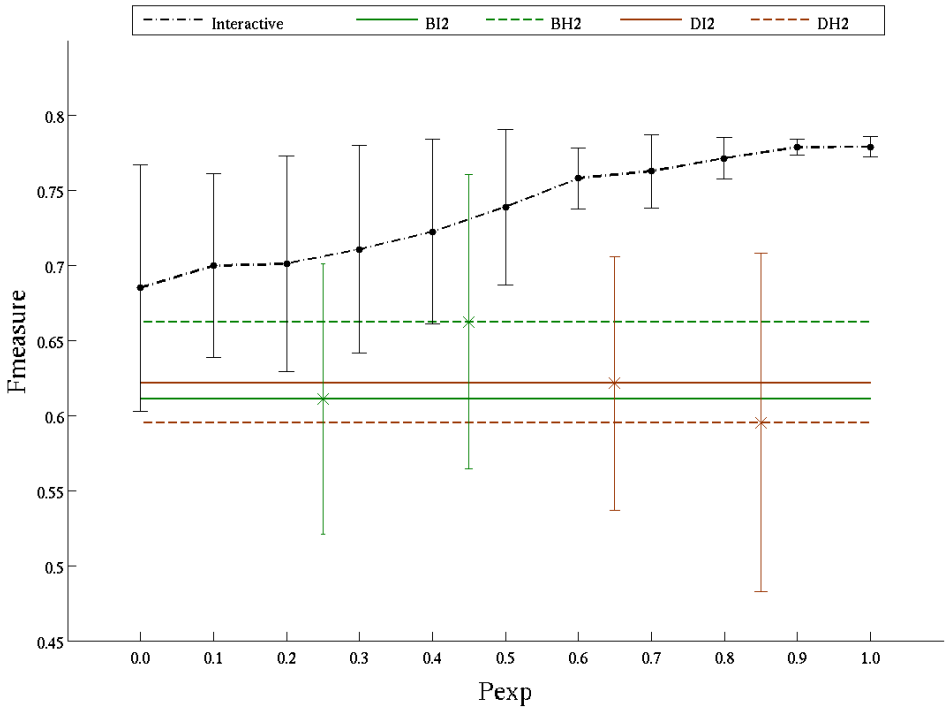

Figure 7.27: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *News-sim3*. *LDC* outperforms the direct clusterers in unsupervised mode ($P_{exp} = 0$) but not significantly. Term-supervised *LDC* has the smallest standard deviations. This is more evident when the level of user's expertness increases from 0.6 to 1.0. Only 10 terms are used for supervision ($f = 10$).

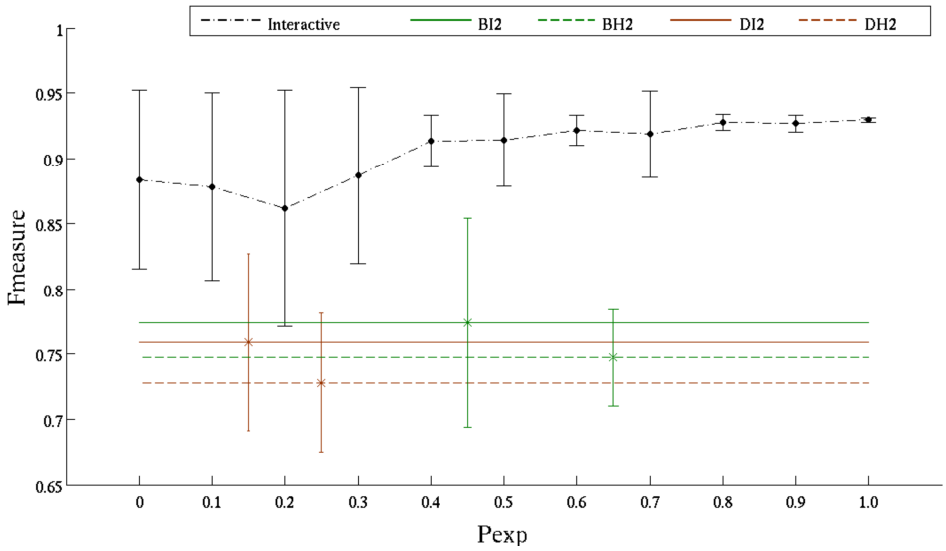

Figure 7.28: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *SMS*. *LDC* significantly outperforms *BH2*, *DI2*, and *DH2* in unsupervised mode ($P_{exp} = 0$). Once the level of user's expertness reaches 0.4, *LDC* significantly outperforms the direct clusterers. The number of terms for term labeling is 20 ($f = 20$).

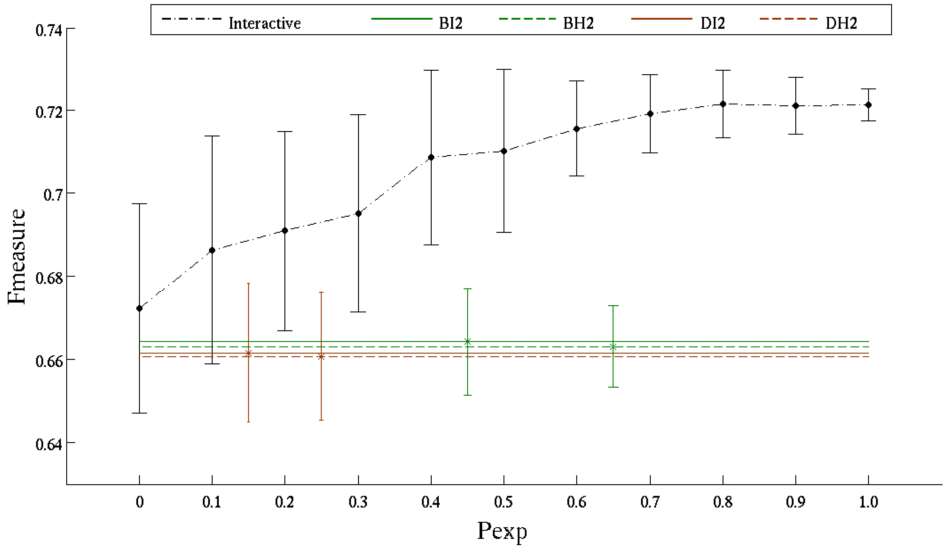

Figure 7.29: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *WebKB*. *LDC* outperforms the direct clusterers in unsupervised mode ($P_{exp} = 0$) but not significantly. Once the level of user's expertness reaches 0.4, *LDC* significantly outperforms the direct clusterers. The number of terms for term labeling is 20 ($f = 20$).

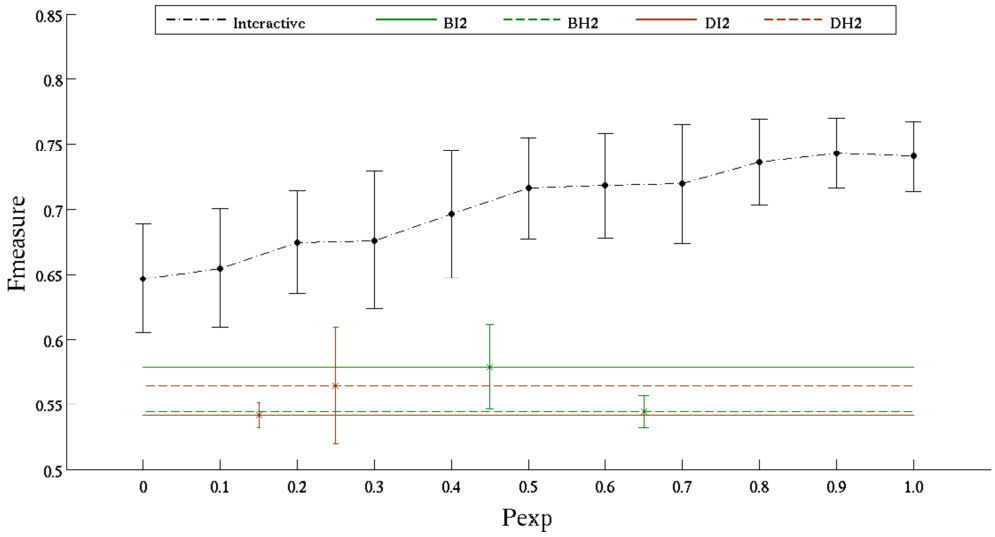

Figure 7.30: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *Reuters8-whole*. *LDC* significantly outperforms the direct clusterers in unsupervised mode ($P_{exp} = 0$) and also with term labeling. The number of terms for term labeling is 20 ($f = 20$).

2. *LDC* in unsupervised mode generated similar results to the direct clusterers on datasets *News-sim3*, *News-rel3*, and *WebKB*. Once *LDC* is enhanced by term labeling, it significantly outperformed the competitors.

3. On two datasets *LA-Times* and *News-multi7*, the direct clusterers outperformed unsupervised *LDC*. *BH2* significantly outperformed unsupervised *LDC* on *LA Times* and *BI2* significantly outperformed it on *News-multi7*. For these two datasets, we needed more term labeling to generate similar results to the direct clusterers.

The conclusion of this section is that *LDC* can either outperform the direct clusterers even without supervision or its term-supervised version can be used to generate similar results. However, more term labeling might be needed on some datasets in order to generate comparable results to the direct clusterers. For instance, fuzzy c-means puts keyterms of two classes *comp.sys.mac.hardware* and *misc.forsale* of *News-multi7*, and two classes *Foreign* and *National* of *LA Times* in one cluster. In order to separate the subspaces of these classes and extract seed documents, we need more term labeling. One benefit of *LDC* over the direct clusterers is that the user

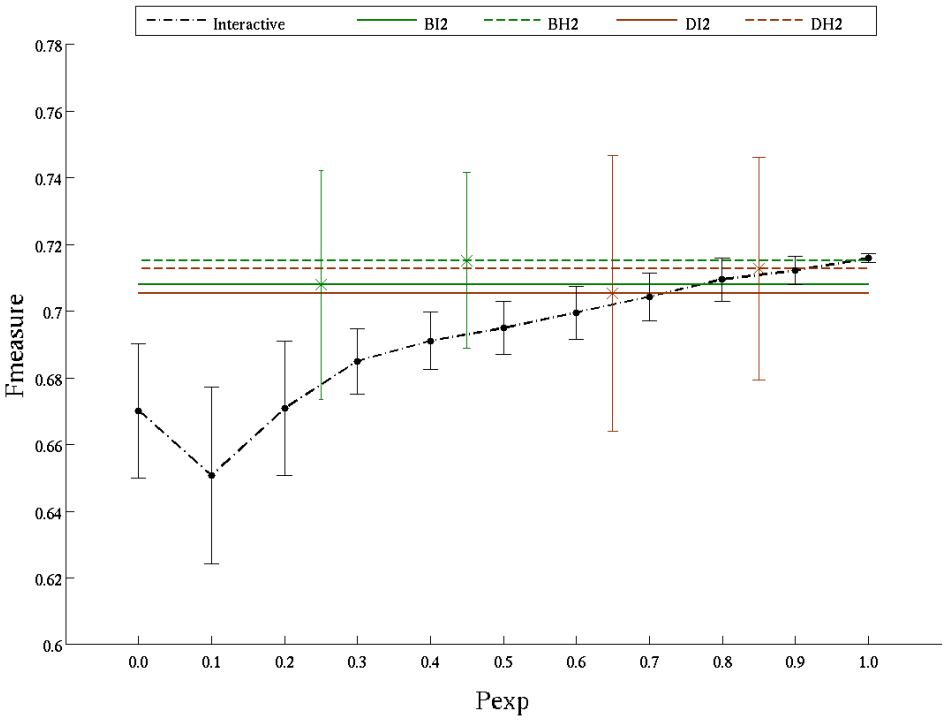

Figure 7.31: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *LA Times*. The direct clusterers outperform *LDC* in unsupervised mode ($P_{exp} = 0$) and it is significant for *BH2*. Once the level of user's expertness reaches 0.7, *LDC* generates similar results to the direct clusterers. However, we need to label more terms for this dataset ($f = 50$).

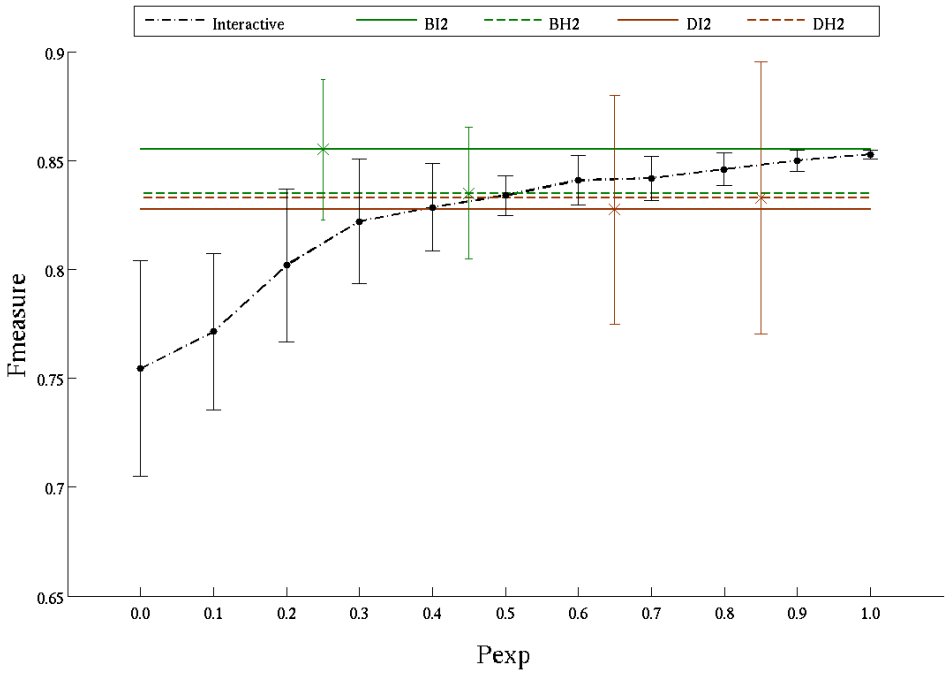

Figure 7.32: The quality of clusters obtained from the unsupervised clusterers, Bisecting I2 (*BI2*), Bisecting H2 (*BH2*), Direct I2 (*DI2*), and Direct H2 (*DH2*)), and Term-supervised *LDC* (*Interactive*) on *News-multi7*. The direct clusterers outperform *LDC* in unsupervised mode ($P_{exp} = 0$) and it is significant for *BI2*. Once the level of user's expertness reaches 0.4, *LDC* generates similar results to the direct clusterers. However, we need to label more terms for this dataset ($f = 40$).

can be involved in the clustering process with different types of user-supervision. In our graphical interface, the user can search for keyterms and make subspaces in these cases provided that she has background knowledge about the collections.

7.3 Conclusion

We proposed three user-supervised versions of *LDC*, Document-supervised, Term-supervised, and Dual supervised. The document-supervised algorithm is similar to semi-supervised clustering algorithms. Labeling a set of training documents is first performed by the user. The labeled documents are then used to initialize *LDC*.

Term-supervised *LDC* is an interactive clustering algorithm. After document clustering is done, the top keyterms associated with each document cluster will be displayed to the user in the form of a term cloud. The user reorganizes the terms among the term clouds in order to determine her preferred topics of document clusters. She can also increase or decrease the number of clouds. In this way, the user adjusts her desired number of document clusters interactively. We did not examine this option in our simulations. This is because the true number of classes in benchmark datasets is used for evaluation in this work. This feature is available in the visual interface of Chapter 8.

Dual-supervised *LDC* has a combination of the document and term supervisions. Labeling document is performed once in advance and term labeling can be performed interactively till the end of *LDC*.

Based on the experiments reported in this chapter we can get the following conclusions:

1. We demonstrated that with a comparable amount of simulated user interactions, the proposed term labeling approach is more effective than the baseline term selection approach. This observation suggests that term labeling is a more effective interactive method in text document clustering.

2. Based on the experiments performed in this chapter, term-supervised LDC could not improve the quality of clusterings for *Classic4*, compared to the quality of clusterings obtained from *LDC* in the unsupervised mode as shown in Fig. 7.5 and Fig. 7.25. This is mainly because the classes of this dataset are well separated [56]

and term labeling cannot improve the quality of clusterings further. The main advantage of term labeling is in clustering document collections which have similar topics like *News-sim3* and *News-rel3*. Term labeling helps to separate the subspaces of similar topics, which in turn improves our proposed seed documents extraction approach and document clustering.

3. We demonstrated that term supervision is better than document supervision for *LDC*. No significant improvement is obtained after adding document supervision to Term-supervised *LDC*.

4. We compared Document-supervised *LDC* to Seeded k-means. The experiment revealed that semi-supervision based on *LDC* is significantly better than semi-supervision based on k-means in this case.

5. We also compared Term-supervised *LDC* to some unsupervised direct clusterers. *LDC* can either outperform the direct clusterers even without supervision or its term-supervised version can be used to generate similar results. However, more term labeling might be needed on some datasets in order to generate comparable results to the direct clusterers. One benefit of *LDC* over the direct clusterers is that the user can be involved in the clustering process with different types of user-supervision.

Chapter 8

Personalized Document Clustering by Term Clouds

The best clustering is the one which reflects the user's point of view. Personalized document clustering aims to elicit the user's preferences with minimum effort and generate clusterings matching her point of view. It has broad applications in practice and anyone who has a document collection confronts this problem at least once. For instance, students writing a thesis, professors writing a proposal or planning a reading course, or a conference chair organizing sessions of a conference. It can help in organizing files into folders on a laptop or PC, or grouping personal emails into multiple inboxes.

We propose a visual interface for interactive clustering of text document collections in this chapter. Interaction is in the form of term labeling and the underlying clustering method is the Term-supervised *LDC* algorithm proposed in Chapter 7. The interactive clustering is performed in the following way. The keyterms of document clusters are displayed as term clouds in the interface. The user is then asked to label keyterms by relocating them among the clouds. Making new clouds, removing them, splitting or merging them are the other available operations in the interface. The supervised term clouds are then used to initialize Term-supervised *LDC* to re-cluster documents.

We have conducted a user study to evaluate the interface and its underlying algorithm. Analysis of the data gathered through the user study reveals that different users have different points of view in clustering the same collection. The effectiveness of using term clouds in clustering text corpora is also evaluated through users' comments. The comments confirm that presenting the topics of document clusters in the form of term clouds is effective in exploring text corpora. Participants also agreed that they could generate their desired number of clusters by using the interface.

The remainder of this chapter is organized as follows. Section 8.1 explains the user interface implemented to support Term-supervised *LDC*. Section 8.2 explains how users interact with the interface. The user study performed to evaluate the interface

is reviewed in Section 8.3. Section 8.4 reviews the comments and data gathered in user study. Section 8.5 presents conclusions and future work.

8.1 User Interface

The interface is implemented as a client-server web application and the interaction is through a web browser. The interface consists of three main panels as shown in Fig. 8.1:

- **Insight Panel**: This panel includes utility buttons, a drop-down menu, and some miniature term clouds. Each miniature term cloud shows four to five important keywords of each document cluster. The number of clouds is the same as the number of document clusters and this number is initially asked from the user. Miniature term clouds give the user a broad insight into the topics of document clusters. The following utility buttons are considered in this panel:
  - button +: It creates a new term cluster. The cluster has no terms or related documents initially and is considered empty. The user can assign terms to the cluster and create its term cloud.
  - button ↻: It redraws the miniature term clouds. The position of terms in the clouds is selected randomly.
  - button C: It shows the fixed term clouds of the current session. Each run of the interface is a session. After term labeling, the participant should fix the clouds.
  - button U: It opens the upload page. The users can upload their *PDF* documents from that page.
  - button S: It saves all the fixed term clouds of the current session into the server. The user can fix a term cloud in the current session, but fixed clouds are not saved permanently in the system unless the user saves them with this button. The sessions are saved based on the system date and time and will be kept in the user's profile. The drop-down menu of the panel shows the list of saved sessions. By selecting a session, its corresponding fixed term clouds are shown in a pop-up window as depicted in Fig. 8.2. The

term clouds can be loaded from the pop-up window into the Interaction Panel. If the user does not fix a cloud, it means that she does not like the respective cluster and it should be removed.

- button ?: It opens the user-guide page. The main steps of the interactive document clustering along with some videos and pictures are provided in this page.

- **Interaction Panel**: Term labeling is performed in this panel. The user selects two clusters every time. For each selected cluster, a document view and a keyterm tree view are displayed. The document view lets the user see the content of the documents. She also can download the respective *PDF* files of documents through the links provided. The number of documents in the selected cluster is also shown on top of document view.

  The tree view includes top 150 keyterms of the selected cluster computed based on the current partitioning by using the χ^2 Statistic. There is a white panel between two clusters. User-selected terms of the left cluster are in blue and those of the right cluster are in green. The terms in red belong to neither of the selected clusters. The main duty of the user is to select the important keyterms of each cluster and put them into the white panel. The following operations are provided for this white panel:

  1. Adding terms to the panel by clicking on terms in tree views.
  2. Adding terms to the panel by clicking on terms in document content view.
  3. Removing terms from this panel by double-click.
  4. Drag and dropping terms to left or right of the white panel to relocate them between the clusters.
  5. Drag and dropping terms to change their vertical positions. The vertical position of the terms show their relevance to the respective clusters. The term at the top of the panel is the most relevant one.

  The buttons in the Interaction Panel have the following functionalities:

- button 'Cloud': It re-create the term cloud based on the top 20 to 25 terms of the respective tree view. The keyterms in a tree view are descendingly sorted based on their relevance to the respective cluster.
- button 'My cloud': It creates a cloud based on the blue or green terms. This is the cloud generated based on the user-selected terms.
- button 'Clear Cloud': It removes the terms from the cloud.
- button 'Clear Terms': It removes the blue or green terms from the white panel.
- button 'Fix': It fixes the cloud generated from blue or green terms. The fixed clouds are saved in the current session.
- button 'Send': It sends the fixed term clouds of the current session to the Term-supervised *LDC* algorithm on the server so as to re-cluster documents based on the user's term labeling. The number of fixed term clouds specifies the number of document clusters. The terms in the term clouds are used to initialize the term clusterer of Term-supervised *LDC*. The most relevant term will get a value of 1 in the membership matrix of fuzzy c-means. The membership value of the least relevant term is 0.5 and the other terms have a value between 0.5 and 1 based on the following formula:

$$\text{Membership}(t_i) = 1 - \frac{0.5}{M_i - 1}(i - 1) \tag{8.1}$$

  where M_i is the number of terms selected for a cluster by user, and i is the position of term t_i in the white panel. The position of the most relevant term is 1.

There are four drop-down menus in the Interaction Panel. Two menus are used to select the clusters and the other two are used to explore the content of documents. There is also a filter option in each document view. This option lets the users search for the documents based on the terms they select from tree views.

- **Cloud Panel**: This panel consists of two sub panels to show term clouds. Each cloud shows at most 25 keyterms of each cluster. The font size and transparency

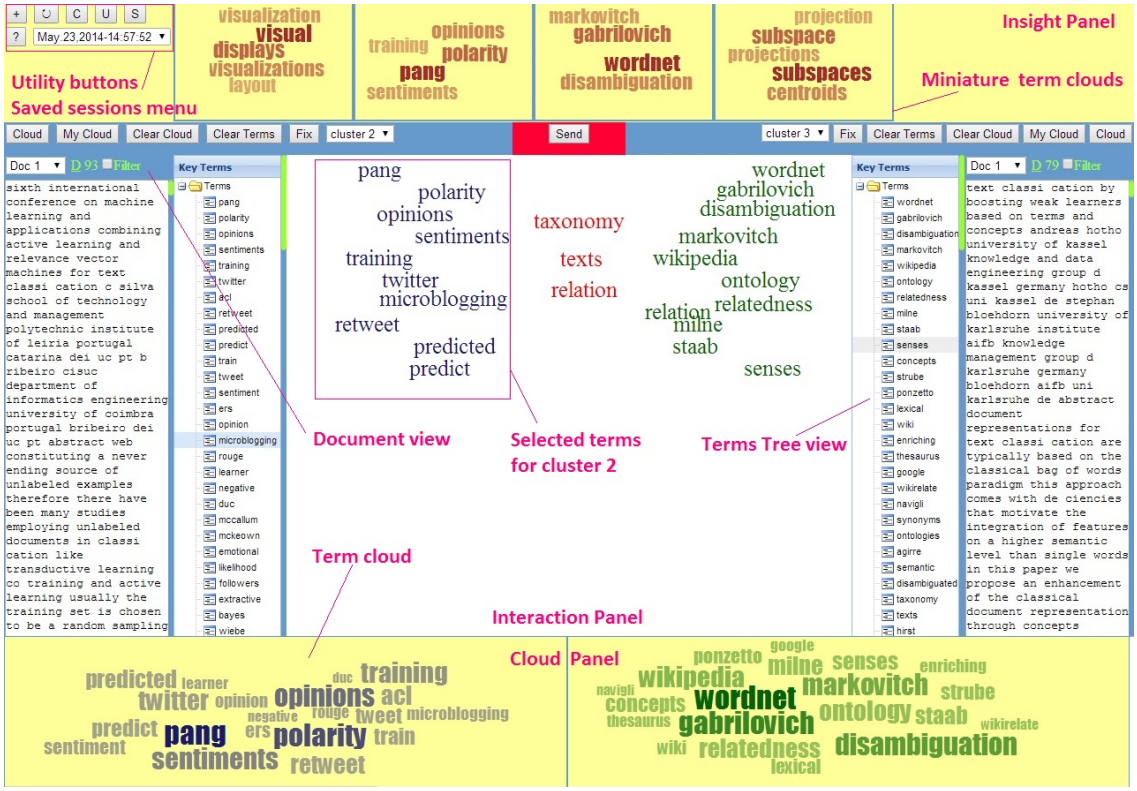

Figure 8.1: The visual interface implemented to support Term-supervised *LDC*. Insight Panel provides a broad insight into the topics of document clusters. The term labeling is performed through the Interaction Panel. Could Panel shows the term clouds the system generated or the term clouds generated based on the user-selected terms.

of terms in the clouds indicate their relevance in the respective cluster. The vertical position of the blue or green terms in the white panel are used for this purpose.

The Term-supervised *LDC* algorithm is implemented in *Python*[1]. Uploading *PDF* documents into server is performed using *Uploadify* module[2] . The content of *PDF* documents are extracted using *XPDF* module[3]. All the pre-processing steps are implemented in *Perl*[4]. The main interface is implemented using *HTML* and *JavaScript* libraries like *JQuery*[5], *D3*[6], and *ExtJs*[7].

[1] https://www.python.org/
[2] http://www.uploadify.com/
[3] http://www.foolabs.com/xpdf/Theco.html
[4] http://www.perl.org/
[5] http://jquery.com/
[6] http://d3js.org/
[7] http://www.sencha.com/products/extjs/

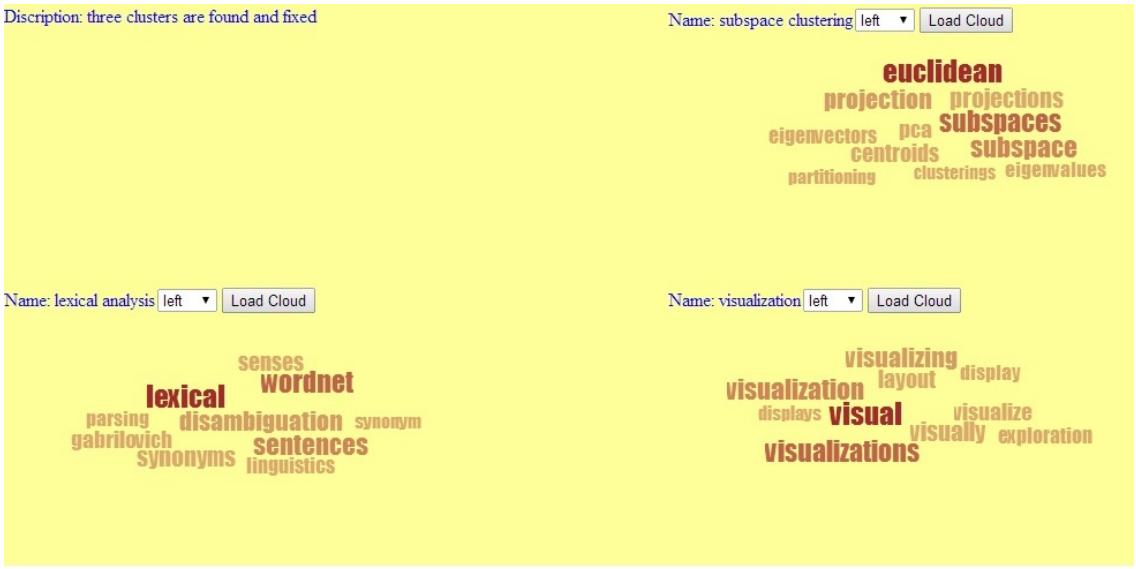

Figure 8.2: A sample of a saved session. Three term clouds are fixed in this session. The user can load the save clouds from this page into the white panel of the Interaction Panel by clicking on "Load Cloud". The drop-down menus next to the buttons let the user select in which side of the white panel the terms should be loaded.

8.2 User Interaction

In this section, we present the main steps of the user interaction. We also present the best scenario to generate user-desired clusters by using our interface.

The whole user interaction consists of the following three main steps:

1. The user uploads her documents in *PDF* format. The contents of the documents are pre-processed and then a document-term matrix is generated.

2. The user interacts with the interface so as to generate her preferred partitioning.

3. The underlying clustering algorithm generates a clustering based on the user's preferences. The generated clustering is provided in *Zip* format and she can download the clusters from the links provided in the interface.

To generate a desired document clustering, the participant needs to send her fixed term clouds to the server. The term clouds are used to guide Term-supervised *LDC*. Based on our experiences, the best scenario to perform this task includes two phases:

1. **Exploration Phase**: The user runs the clusterer multiple times with different

numbers of document clusters. Each run corresponds to a session. In each session, the user performs term labeling in the following way:

(a) The user explores the automatically generated document clusters.

(b) If she likes a cluster:

- She should create a term cloud by selecting the best keyterms representing the respective topic. Keyterms of tree view or document contents can be used for this purpose.
- She also decides on the relevance of keyterms by tuning their vertical positions.
- She fixes the respective cloud.

(c) After fixing keyterms of interesting clusters, she saves the fixed clouds of the session.

2. **Closure Phase**: In this phase, the user has explored the collection and decided on the number of document clusters she likes to generate. The user should now send the fixed term clouds to Term-supervised *LDC* on the server. She just needs to have all the fixed term clouds in a session and click on 'Send' button. For this purpose, she can use the fixed term clouds of the current session, or load the clouds of the saved sessions. She either loads a saved cloud into a new empty cluster or into an existing cluster whose terms are not fixed. After loading all desired term clouds in the current session, the user sends them to the server to generate a new clustering. She can also go back to the Exploration Phase to modify her term clouds.

8.3 User Study

A group of 30 computer science students have participated in the user study. There was no restriction on the amount of time they spent on the study and they could do the study anywhere at any time. We have considered the following assumptions about a participant:

1. She is familiar with the concept of document clustering in general.

2. She has read scientific papers and knows the concept of keyterms.

3. She has enough background knowledge to cluster her document collection. In other words, she should be able to organize her documents in folders on her computer.

The user study is performed in two modes. In the first mode, a participant clusters her own document collection. In the second mode, a document collection is shared among multiple participants, but each participant must cluster the collection individually. The only difference is that a collection is shared in the second mode. More information about these two modes is provided below:

1. **Individual participation**: Each participant has her own document collection. She uploads her documents into the system and performs the interactive clustering. After the user study, she fills out a questionnaire so as to rate her satisfaction with the interface. 21 individual participants have interacted with the interface in this mode.

2. **Group participation**: In this mode, we have a group of individual participants who are asked to cluster a document collection. Each participant must generate a partitioning individually. After document clustering, each participant fills out a questionnaire and provides some comments. The purpose of this participation is to find out whether the participants have different points of view in clustering the same collection. A group of 9 participants with research interest in text mining has clustered a collection of 300 scientific papers in this mode. The participants are the members of a study group who meet weekly to discuss scientific papers. The collection includes all the papers they have discussed already.

We create operation logs to find out the most frequent operations used during the interactions with the interface.

8.4 Results and Discussion

In this section, we analyze the data obtained from the user study. The analysis is based on the questionnaires the participants filled out and the system logs generated during

Table 8.1: The participants' ratings in the questionnaire

Rate	Label
1	"Strongly Disagree"
2	"Somewhat Disagree"
3	"Neutral"
4	"Somewhat Agree"
5	"Strongly Agree"

their interactions with the interface. There are 20 statements in the questionnaire and the participants have rated them by a value between one and five. The label of the rates are shown in Table 8.1.

Different Points of View

The aim of this analysis is to find out whether the participants of the group participation mode have different points of view in clustering the collection. We created the collection for these participants. Each participant then clustered the collection without collaboration with the other participants of the group.

The maximum number of clusters generated is 7 and the minimum number is 3 as shown in Fig. 8.3. The top five keyterms of clusters generated by participants are shown in Table. 8.2.

Users 1 and 3 have generated three clusters with almost similar topics: Visualization, Subspace Projection, and Semantic Analysis. Users 5, 6 and 7 have generated five clusters. Four clusters are common among them including Sentiment analysis, Visualization, Subspace Projection, and Semantic analysis. Users 8 and 9 have generated four clusters with similar topics: Sentiment Analysis, Visualization, Subspace Projection, and Semantic Analysis. User 4 has generated six clusters with topics: Sentiment Analysis, Visualization, Subspace Projection , Semantic Analysis, Supervised Learning, Document Summarization. User 2 has generated seven clusters. He has generated a cluster about Topic Modeling (*LDA*) which has not been seen in the others' clusterings.

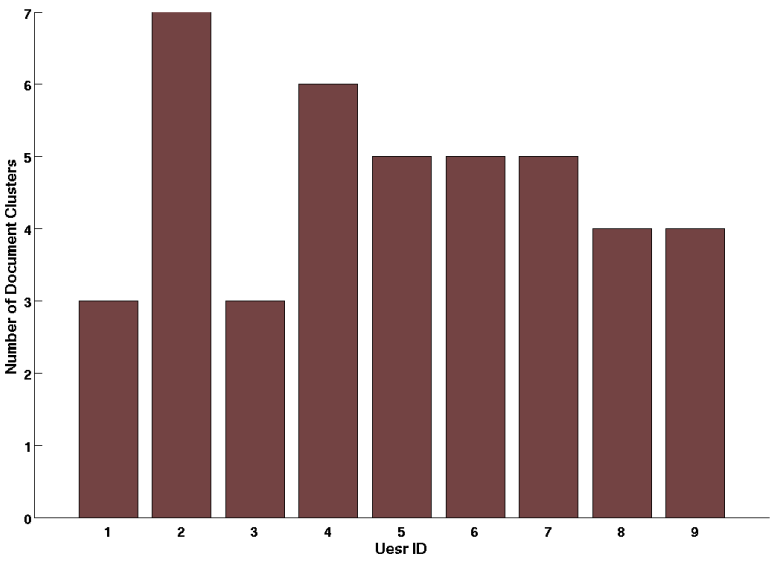

Figure 8.3: Different numbers of document clusters are generated in a group of 9 participants. The average number of clusters is 4.66 and its standard deviation is 1.32. The collection includes 300 scientific papers in the area of text mining.

The number of generated clusters and their topics in this case confirm that different users may generate different partitionings of the same collection even in a small group of participants in this user study.

We also analyzed the amount of time spent by the participants. The analysis shows that there is no correlation between the number of obtained clusters and the interaction time, based on Spearman[8] and Kendall[9] correlation tests. For instance, users 3 and 6 have both interacted between one and two hours, while user 3 has generated three but user 6 has generated six clusters. Or, users 5, 6, 7 have spent different times but generated the same number of clusters.

User-supervised clustering

We analyze the participants' opinions about being able to change the number of clusters and their topics in this section. Three statements are considered in the questionnaire for this purpose:

1. The automatically generated clusters are far from what you desired.

[8] `http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient`
[9] `http://en.wikipedia.org/wiki/Kendall's_tau`

Table 8.2: The top five keyterms of the clusters generated in the group participation. Different numbers of clusters with different topics are generated.

User ID	Top keyterms
1	visual,displays,visualizations,visualization,visually subspaces,subspace,projected,projections,principal gabrilovich,lexical,disambiguation,wordnet,markovitch
2	gabrilovich,markovitch,strube,ponzetto,staab duc,multidocument,summarizer,marcu,radev visual,infovis,displays,graphics,visually,displayed subspaces,projections,partitioning,variance,gaussian tweet,twitter,tweets,friends,celebrities icio,lda,dirichlet,blei,erences polarity,pang,subjectivity,neutral,wiebe
3	visual,visualization,displays,visualizations,visually gaussian,dimensional,variance,subspaces,criterion lexical,wordnet,acl,linguistic,sentence
4	sentiment,polarity,opinion,pang,sentiments visual,visualization,visualizations,visually,displays gabrilovich,markovitch,wikipedia,disambiguation,staab multidocument,duc,summarizer,marcu,rouge subspaces,subspace,principal,projected,orthogonal mccallum,icml,cohn,neural,learner
5	sentiment,polarity,opinion,pang,twitter scientometrics,citations,chairs,citespace,fronts visual,displays,visualization,displayed,exploration subspaces,gaussian,subspace,variance,projections lexical,wordnet,gabrilovich,senses,ontology
6	principal,subspaces,subspace,projections,pca mccallum,generative,duc,cohn,supervised polarity,sentiment,opinion,pang,sentiments gabrilovich,markovitch,disambiguation,wordnet,ontology visual,visualization,visualizations,visualizing,interfaces
7	gaussian,variance,subspaces,subspace,projections betweenness,velardi,citations,paths,cited visual,visualization,displays,displayed,infovis polarity,pang,opinion,sentiment,sentiments gabrilovich,markovitch,disambiguation,milne,relatedness
8	subspaces,centroids,subspace,projections,projections pang,polarity,opinions,sentiments,twitter gabrilovich,disambiguation,wordnet,markovitch,ontology visual,visualizations,visualization,displays,layout
9	opinion,polarity,sentiment,opinions,pang visual,displays,visualizations,visualization,visually lexical,wordnet,gabrilovich,senses,disambiguation subspaces,variance,gaussian,dimensional,subspace

2. It is necessary to be able to change the number of clusters.

3. It is necessary to be able to change the topics of document clusters generated automatically.

Dependency tests based on Spearman and Kendall correlations reveal that:

- Statements 2 and 3 are independent of statement 1. This means that the participants like to be involved in the clustering process no matter the quality of the clusters generated automatically. Regarding the first statement, four participants selected "Somewhat Agree", eight selected "Neutral", fifteen selected "Somewhat Disagree", and three participants selected "Strongly Disagree". Therefore, 13% of participants did not like the automatically generated clusters and for the other 87%, the clusters generated by *LDC* are close to what they expected.

- Statements 2 and 3 are dependent on each other. This means that if a participant likes to change the number of clusters, she also likes to change the topics of clusters and vice versa. Hence, tuning the clustering algorithms with different numbers of clusters is not sufficient and the participants like to be able to change the topics of document clusters as well. In the 30 questionnaires, only two participants disagree with the second statement and three participants disagree with the third one.

- The participants' ratings to these three statements are independent of the amount of time they spent on clustering.

The main observation of this analysis is that even though *LDC* can generate clusterings which are close to what participants like to generate, they still like to interact with the clustering process in order to generate their desired clusters.

In the following two analyses, we try to find out whether the visual interface was successful in generating user-desired clusters. We intend to evaluate the role of term clouds in exploring a collection and generating document clusters in user-desired topics. We also evaluate the interface in generating desired number of document clusters.

Topics of document clusters

The aim of this analysis is to evaluate the presentation of document clusters topics in the form of term clouds. We considered the following three statements in the questionnaire for this purpose:

1. Term cloud visualization is a useful way in exploring the topics of a collection.
2. Term cloud visualization and term labeling is a useful way in generating desired cluster topics.
3. It is easy to identify the topic of a document cluster from the keyterms in the corresponding term cloud.

Our analysis reveals the following facts about the statements:

- These statements are independent of the amount of interaction time.
- Only one participant somewhat disagreed, two participants were neutral and the remaining 90% of participants agreed with the first statement.
- Regarding the second statement, only one participant somewhat disagreed, one participant was neutral, and the remaining 93% of participants agreed.
- Regarding the third statement, two participants somewhat disagreed and three were neutral. Around 83% of participants agreed that it is easy to understand the topics of clusters from the keyterms included in the term clouds.

The participants' feedback confirms that term clouds visualization is an effective way of presenting topics of document clusters. It is also effective in generating clusters in desired topics. This observation also confirms that Term-supervised *LDC* was successful in generating clusters matching users' preferences. In other words, initializing fuzzy c-means based on the supervised term clouds is an effective way of involving users' feedback in the clustering process.

Number of document clusters

In order to evaluate the role of the interface in generating the user-desired number of clusters, we considered two statements in the questionnaire:

1. Term cloud visualization is a useful way to find a desired number of clusters.

2. It is easy to determine a desired number of clusters.

Based on the analysis performed, we found the following results:

- Dependency tests based on Spearman and Kendall correlations reveal that the first statement is independent of the interaction time, but the second one has an inverse correlation as shown in Fig. 8.4.

- Only 6.6% of participants disagreed with the first statement and 20 participants agreed that the visualization is useful to find a desired number of clusters.

- Regarding the second statement, only two participants disagreed as shown in Fig. 8.4.

  - Participant A also believes that the term cloud visualization is not a useful way to find desired number of clusters, the first statement of this analysis. She also somewhat disagreed with the statement "the interface is fast enough as an interactive system." However, she is either neutral or agreed with the other statements of the questionnaire. For instance, she somewhat disagreed with the statement "The automatically generated clusters are far from what you desired."

  - Participant B never agreed with any statement of the questionnaire. Her opinion about the statement "The automatically generated clusters are far from what you desired" is also neutral. She also believes that the interface is not fast enough to be used interactively.

  The dependency tests show that there is no correlation among these two participants' ratings in the questionnaire. Besides the statements of this analysis, the other common point between these two participants is that the system is not fast enough to be used interactively. We thus evaluate the users' rating to the statement "The user interface is fast enough as an interactive system" for this purpose.

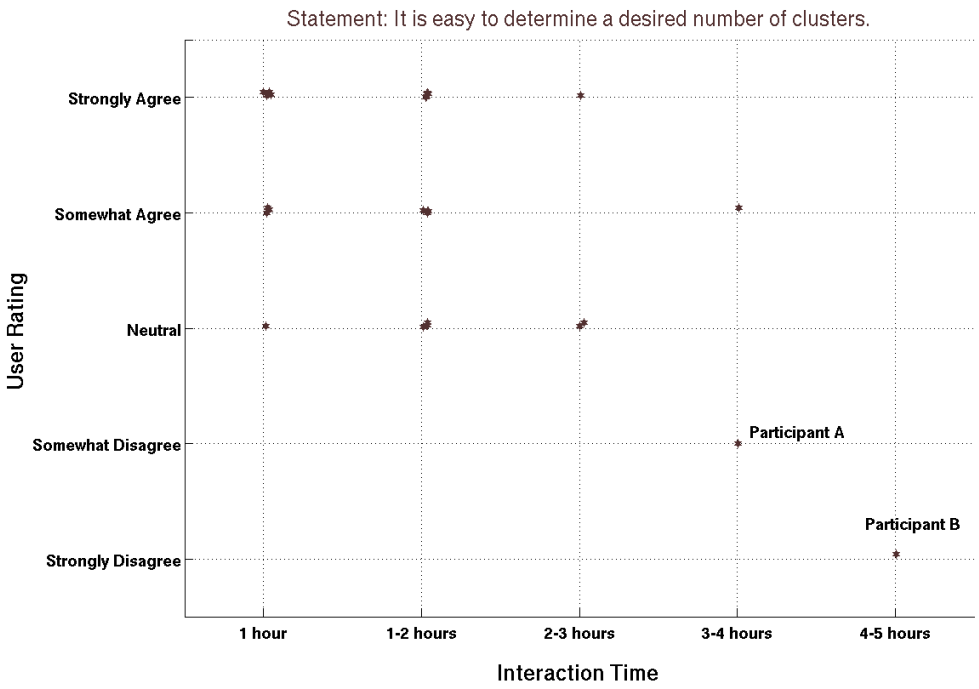

Figure 8.4: The participants' ratings to the statement "It is easy to determine a desired number of clusters." The ratings have an inverse correlation with the interaction time the participants spent in the study.

The statement is independent of the interaction time. Out of 30 participants, 27 participants agreed that the system is fast and only 3 participants disagreed. Hence, the running time of the interface was acceptable to 90% of participants.

One possibility is that our server was busy during the user study of participants A and B since participant A gave us the following comment:
"If it runs on a local machine, I imagine it'd be faster, but running the clustering algorithm on a server seemed to be slow more often than I wanted."

Overall, we could not find out why participant B had difficulties in clustering her collection. Her only comment in the questionnaire is that "From the point of the view of aesthetics, the user interface is not very pretty. Maybe a new design can be conducted to the layout and the colors combination."

Document Content View

A document content view is considered for each document cluster in the interface. This view helps the user in exploring the content of documents and getting better insight into the clusters. They can also add a term to a term cloud from the content of documents. In this analysis, we try to evaluate the role of this view in document clustering. Two statements are considered in the questionnaire for this purpose:

1. It is useful to have document content views in the user interface.
2. It is easy to select a keyterm and add it to a cluster from document content views.

The following facts are obtained in this analysis:

- The participants' ratings to these two statements are independent of the interaction time.
- Only three participants did not find the document view useful and seven participants are neutral. Around 67% of participants mentioned that this view was useful.
- Three participants mentioned that it was not easy to add a term from content view and two participants are neutral in this case. Therefore, 83% of participants did not have difficulties in adding terms from this views

These results confirm that the participants may like to take a look at the contents even though the underlying clustering algorithm, Term-supervised *LDC*, is based on term labeling. Besides, a participant gave a comment that "Visualization of document clusters in addition to terms" should be added to the visualization. Based on the operation logs, we will later show that the participants rather explore the document clusters and make sure that they have generated a good cluster than selecting terms from documents' contents.

Document Clustering Tool

In this experiment we evaluate the participants' opinions about our visual interface in general. We put two statements in the questionnaire for this purpose:

1. The user interface is a useful tool for document clustering in general.

2. With proper documentation, I would like to use the interface in the future.

We get the following facts from the ratings:

- The participants' opinions respective to these statements are independent of the interaction time.

- One participant somewhat disagreed with the first statement and two participants were neutral. Hence, 90% of participants believe that the interface is a useful tool in document clustering.

- Regarding the second statement, one participant somewhat disagreed and six participants were neutral. Around 77% of participants would like to use the interface in the future provided proper documentation.

These results confirm that the visual interface was successful in involving user interaction in the clustering process and generating user-desired clusters in this study.

Frequency of Interface Operations

We have created a log from the participants' interactions with the interface. During the user study, 15834 operations are recorded in the log. The list of operations along with their frequency percentage is shown in Table 8.3. The log of the interface reveals the following facts about the interactions:

- The most frequent operation is "Relocating a term" which includes assigning a term from one cluster into another, or changing the relevance of a term in a cluster.

- The four most frequent operations are related to term labeling including: Relocate a term, Remove all terms, Add a term from terms tree view, and Remove a term.

- Even though our analysis confirms the usefulness of document content views, the system log shows that the participants prefer to add a term from tree views than content views. The frequency of adding a term from tree views is nine times

Table 8.3: The participants' operations in the user study

Operation	**Frequency Percentage**
Add a new cluster	0.36%
Fix the selected terms	4.7%
Remove a term	7.19%
Remove all terms	12.69%
Add a term from document contents	1%
Add a term from terms tree view	9.10%
Relocate a term	46.34%
Filter documents by terms	0.49%
Save a session	0.71%
Open a saved session	1.48%
Load a cloud from a saved session	1.66%
Send the fixed clouds to server (re-clustering)	0.57%

larger than adding from document contents. It reveals that the participants browse the document contents to get more insight about the clusters than finding an informative term.

- The frequency of re-clustering is 91 and its percentage is 0.57% in the system log. Since we had 30 participants in the user study, three rounds of term labeling is performed on average by each participant.

- The frequency of removing all terms is larger than the frequencies of adding terms. This is simply because all the terms in the white panel should be removed when participant switches to the other clusters or refreshes the interface. Therefore, we cannot make any conclusion based on this frequency count.

Participants' Comments

We present a few comments the participants mentioned in the questionnaires:

- "There are some terms like name of authors in term cloud which I found disturbing. They can easily be identified and removed in a pre-processing step."
  We believe that sometimes a name of an author is discriminative enough to be used as a keyterm of a cluster. For instance, when "Gabrilovich" and "Markovitch" are keyterms of a cluster, the cluster is probably related to semantic analysis. However, we agree that the *Pdf2Text* module sometimes outputs some words which are not in any dictionary. In the pre-processing step, we should have removed this kind of terms.

- "Some sort of stemming or lemmatization would reduce the number of very similar keyterms (e.g. disambiguate and disambiguation)."
  We first believed that it is easier for the participants to see the terms without any lemmatization. However, we observed in the user study that there are some clusters whose top keyterms are very similar, e.g. visual, visualized, visualization, and visually. It is thus much better to perform stemming or lemmatization.

- "I really liked being able to control the importance of a term in a cluster, and being able to place a term into multiple clusters. The workflow was well explained with good videos and documentation."
  This comment shows the importance of soft term clustering for this participant.

- "It might be useful to see the editable term clouds for all your clusters at once. This would give a good overview, and allow relabeling terms with far fewer clicks."
  "I used left side more than two together."
  We totally agree that presenting two clusters every time is not the best way for presenting term clouds.

- A participant mentioned that "Highlighting terms in documents" should be added to the interface. We totally agree with this comment.

- "Searching an arbitrary keyword (not existing in the extracted keyterms) and see if it exists in the corpus or not e.g. I wanted to search for Neural Networks in the set of documents. Being able to search or filter multi-term keywords could

be useful too."

We agree with this comment that a global search in the corpus is useful too.

- "The final results of clustering is very good in terms of categorization quality but the keywords that it finds (and put in the final clouds) are far away from the keywords that I defined (and I think they should be). I don't know if this a big issue or not."

  This is simply because the keyterms in the supervised term clouds are fed into the term clusterer, but the keyterms we display in the interface are extracted from the document clusters. These keyterms are not necessarily identical for some clusters since a keyterm might be very discriminative but the participant did not put it in the clouds.

- "The system is designed very well, The cloud visualization of words is interesting and it will really help some company for big data. The interface is also nice."

  We will later mention in the Future Work how our Term-supervised *LDC* algorithm can be extended to collaborative clustering of big data.

- "It is a great interactive user interface which I have worked with. It has lot of options and buttons to help users in terms of interaction and also it is neat using cloud to the clusters' topic and keyterms. To be honest, I guess the coloring in main interface isn't much desirable, but using the tree view of keyterms is an excellent idea and would help users to interact with system well."

  "The system works good and the operation was acceptable for me, but the user interface was not user friendly. Adding more animation or color in cluster part would be useful."

  These comments also indicate that we still need to work on the aesthetic aspects of the interface.

- "In my opinion, the system is well designed and works perfectly with words. i think it is better to have capabilities related to phrase processing or combination of two or three words."

  " A lot of the keyterms for my clusters have multiple words, so it would be nice to extend the algorithm to detect multi-word terms."

We totally agree with the comments since presenting multi-word terms provide more insight into a document collection. We also mention later in the Future Work how the semantic of keyterms can be presented in the interface.

- "I liked too much the intelligence behind the proposed system. Linking topics of research and generating documents quickly. Processing the changes in clusters too quickly. Mainly the cloud, very useful. Anyway, in the end the possibility to download the articles generated from the topics of research makes it upon super interesting and useful for the user researcher."
  "The term cloud visualization is a very useful and intuitive way to cluster documents. The interface of the system provides great user experience. Besides, some functions are really helpful, such as adding a new cluster, and can compare with another cluster at the same time."
  "The idea of using keyterm clouds for document clustering is very interesting and useful."
  These participants liked the idea of using term clouds and found it useful in the interface.

Reviewing the participants' comments reveal that most participants are satisfied with the idea of using term clouds in presenting topics of document clusters. They also did not complain about the underlying clustering algorithm, *LDC*. However, they mentioned that the user interface should have been implemented in a more user friendly way and its aesthetic aspects should be improved.

8.5 Conclusion

We proposed a visual interface to support our Term-supervised *LDC* algorithm. The interface is based on presenting document clusters by their keyterms using term clouds. Each term cloud includes the keyterms of a document cluster in this interface. The operations of the interface let the user perform term labeling in the form of relocating terms among clouds or making a new one, or merging the existing ones.

A user study is then conducted to evaluate the interface and its underlying clustering algorithm. By analyzing the questionnaires filled out by participants and reviewing their comments, we conclude that:

1. Regardless of the quality of the clusters generated automatically, the participants like to be involved in the clustering process. 87% of participants mentioned that it is necessary to be able to change the number of document clusters and only 10% of participants did not agree that it is necessary to change their topics.

2. Different users have different points of view in clustering text documents.

3. Term cloud based visualization is an effective way in presenting topics of document clusters and most participants liked this idea.

4. The participants did browsing into the content of documents of a cluster mostly to get a deeper insight into the cluster. They preferred to use the keyterms we have extracted from the document clusters to create their own term clouds. This is because it needs less effort.

5. Around 84% of participants found the interface a useful tool in document clustering and would like to use it in the future. This fact confirms that Term-supervised *LDC* was successful in generating user-desired clusters. However, some improvements are needed in order to make the interface more user-friendly.

6. Except one participant, the other participants mentioned that it was easy to perform term labeling in general.

Chapter 9

Conclusion and Future Work

The research focus of this thesis is interactive personalized text document clustering. The main challenge of the research is to involve the user in the clustering process, in order to generate her desired clusters, with minimum effort. To address this challenge, we first proposed a novel unsupervised text clustering algorithm. One advantage of the proposed algorithm is that it can be easily adapted for interactive clustering, thanks to its double clustering approach. It can also generate soft document clusterings. We then proposed three user-supervised versions of the algorithm based on term supervision, document supervision, and dual supervision. The user-supervised algorithms are first evaluated by supervision oracles. Oracles are generated based on the document class labels so as to simulate user interactions. We then proposed a user interface and conducted a user study to evaluate the proposed algorithm in interaction with human users. Document clusters are represented by term clouds in this interface. Participants of the user study are asked to generate personalized document clusters and evaluate the proposed interface and its underlying clustering algorithm. In detail, the contributions of this thesis include the following.

First, we proposed an unsupervised evolutionary algorithm to cluster text document corpora. There are two novelties in the algorithm: (1) We used an evolutionary module to distill non-discriminative terms from term clusters. (2) We proposed a heuristic approach to extract lexical seed documents from distilled term clusters. We demonstrated the benefits of the algorithm over double and co-clustering algorithms in our experiments.

Second, we proposed an unsupervised non-evolutionary algorithm to cluster text documents. The Lexical Double Clustering (*LDC*) algorithm uses a feature selection method locally to remove non-discriminative terms from term clusters. This is the main novelty of the proposed algorithm. We conducted comprehensive experiments to evaluate the performance of *LDC* against state-of-the-art clustering algorithms. The

experimental results demonstrated that *LDC* can generate comparable clusterings to the *LDA* model. The main advantage of *LDC* is that it can be easily extended to involve users in the clustering process. It has a computation time complexity of $O(NMK^2I)$, where N is the number of documents, M is the number of terms, K is the number of document clusters, and I is the maximum number of iterations of fuzzy c-means.

Third, we proposed *LDC* based on Google NGram similarity instead of cosine similarity. We then performed an experiment to evaluate *LDC* using Google NGram similarity metric. The results showed that *TFIDF* based cosine similarity is better than Google NGram based similarity in our experiments.

Fourth, we extended *LDC* into an ensemble clustering algorithm to incorporate Wikipedia concepts in the document representation. The novelty of the ensemble algorithm is its consensus module, which aggregates the clusterings generated lexically and semantically. The main component of the module is a Naive Bayes classifier. The classifier is trained by the documents which sit in the same clusters generated lexically and semantically. Our experimental results demonstrated that the proposed ensemble algorithm can improve the quality of document clusters even if the clusters obtained from the document-concept representation alone are inferior to those obtained from the document-term representation. In our experiments, the quality of clusters obtained from Wikipedia concept representation is always inferior to those obtained from term representation.

Fifth, we proposed three user-supervised versions of *LDC*: Document-supervised, Term-supervised, and Dual supervised. Term supervision is in the form of term labeling. Document supervision is similar to the document labeling approach used in semi-supervised text clustering algorithms. Dual supervision includes both term labeling and document labeling. We performed several experiments to evaluate the user-supervised algorithms by using supervision oracles. We first demonstrated that term labeling is more effective than document labeling in terms of improving quality of clusterings compared to unsupervised mode. This is more evident when the user feedback is noisy. We then demonstrated that no significant improvement can be obtained after adding document supervision to Term-supervised *LDC* in our experiments. We also demonstrated that with a comparable amount of simulated user

interactions, the proposed term labeling approach is more effective than a baseline term selection approach. We can conclude that term labeling is more effective than both document labeling and term selection based on the *LDC* algorithm. We finally demonstrated that Document-supervised *LDC* can outperform seeded k-means, which is a semi-supervised algorithm based on document labeling.

Finally, we proposed a graphical interface to support our term supervised clustering algorithm in interaction with human users. The graphical interface is based on term cloud visualization. We conducted a user study to evaluate the interface and its underlying clustering algorithm, Term-supervised *LDC*. Analyzing the data gathered from the questionnaires and comments revealed that the proposed interface is useful in text document clustering. Most participants mentioned that they would like to use the interface in the future. The comments also demonstrated that the participants liked the idea of representing document clusters by term clouds.

Future Work

In this section, we provide the following future work based on the clustering algorithms proposed in this thesis:

1. Clustering big data is one of the main challenges in text mining. The size of a collection in big data is too large for a user to be able to cluster documents individually. On the other hand, clustering big data with the fastest clustering algorithm still takes too long time to be performed interactively. One solution is to divide the collection into smaller corpora and ask multiple users to cluster them. Each user clusters a corpus and an aggregation module combines the obtained clusters finally. As a future work, our idea of using term cloud visualization can be used as a collaborative big data clustering in the following way:

    - Each user clusters a subset of collection interactively and her desired term clouds are saved in the system.
    - A visualization displays all the term clouds obtained from the collection.
    - The users can then discuss and make an agreement about their desired term clouds.

- Given the desired aggregated term clouds, we can extract lexical seed documents as proposed in Section 5.1.2.
- The seed documents can then be used to cluster the collection.

The other possible scenario is based on ensemble clustering approach:

- Each user clusters a subset of collection interactively and her desired term clouds are saved in the system.
- A visualization displays all the term clouds generated from the collection.
- The users can then discuss and make an agreement about their desired term clouds.
- Ensemble clustering approaches can be used to find a consensus document partitioning, e.g. by voting, provided that the subsets are overlapped.

2. Multiple users may also help with clustering smaller collections where consensus is required, e.g. when a program committee is clustering the accepted papers of a conference. Term-supervised *LDC* based on term cloud visualization can be used in this case.

3. A possible future work is to use Wikipedia concepts in the interface. Instead of representing topics by terms, their related Wikipedia concepts can be displayed as concept clouds. Given the keyterms of each cluster, it is possible to wikify their top related topics from Wikipedia. The user then interacts with the concept clouds instead of term clouds. A user study can then clarify which representation the participants prefer to use; term clouds or concept clouds.

4. Another future work is to use multi-word terms in the clustering process and the visualization. A text processing module extracts multi-word terms from documents. An experiment will then show whether single terms, multi-word terms, or their combination result in better clusters in terms of quality. A user study can also clarify which presentation the participants prefer to use: single terms, multi-word terms, or their combination. Multi-word terms are also related to concepts, and they can be considered in conjunction with concepts extracted from Wikipedia.

5. The other future work is to extend the Term-supervised *LDC* in order to generate hierarchical clusterings. Each document cluster can be treated as a document collection to generate more specific clusters. Representing document clusters by term clouds helps the user to figure whether a document cluster includes other specific clusters or not. The final hierarchical clusters can be used to create a visualization like a mind map.

6. One active research area in data clustering is multi-view clustering[1]. As future work, *LDC* can be extended to a multi-view clustering algorithm [71, 22, 14]. Text documents can be represented based on Wikipedia concepts, document terms, or multi-word terms. The clustering obtained from one feature space can be used to bootstrap the clustering process in the other spaces. Even though we have proposed an ensemble algorithm in Chapter 6 to aggregate the clusterings obtained from *BOW* and *BOC* models, further analysis is required to compare multi-view clustering algorithms to our proposed ensemble algorithm.

Limitations

This section enumerates cases in which further evaluations on the proposed clustering algorithms of this thesis are required:

1. Further analysis is required in order to evaluate the performance of *FSDC* in case of multi-labeled document clustering.

2. Some experiments on significantly larger datasets should also be performed in order to evaluate the performance of *LDC*. Further evaluations on document similarity metrics are also required, specifically on semantic text similarity.

3. We need to compare *LDC* to density-based clustering algorithms [12, 5, 98]. The key idea behind these algorithms is that neighboring data points which form a dense region in the feature space should be grouped into one cluster. We believe that density-based algorithms like *DBSCAN* [37] cannot be directly applied on text document collections. Document-term matrices of text collections are sparse and algorithms like *DBSCAN* fail in clustering sparse datasets [98]. This

[1] `http://dl.acm.org/citation.cfm?id=2501006`

is mainly because proximity fails to register if neighborhood is computed in high-dimensional spaces. A solution to this problem is proposed in [12]. A feature transformation technique is used to map documents into a lower dimensional space and then neighboring data points are found in this space. This algorithm is evaluated on synthetic datasets and further evaluations on real text document collections are needed.

4. We need to compare *FSDC* and *LDC* to probabilistic model based co-clustering algorithms [99, 111]. Even though *LDC* is based on soft term clustering and it can generate a soft clustering of documents, further analysis is required to evaluate its performance against these model based co-clusterers.

5. Further analysis is required to evaluate the performance of *FSDC* against the Information Bottleneck co-clustering algorithm proposed in [112]. The co-clusterer is based on soft partitioning of documents and terms, while hard partitioning is assumed in *FSDC*.

6. We should compare our proposed ensemble algorithm (*ELSDC*) to Nonparametric Bayesian clustering ensemble [113]. The main advantage of this model-based ensemble algorithm is in discovering the number of clusters in the consensus clustering [113]. Even though the number of consensus clusters is fixed in *ELSDC* to integrate concept and term representations, further analysis is required to evaluate the performance of *ELSDC* against this ensemble algorithm.

Bibliography

[1] C. Aggarwal, C. Procopiuc, and P. Yu. Finding localized associations in market basket data. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):51–62, Jan 2002.

[2] C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer US, 2012.

[3] R. Aroraand and B. Ravindran. Latent Dirichlet allocation based multi-document summarization. In *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*, AND '08, pages 91–97. ACM, New York, NY, USA, 2008.

[4] J. Attenberg, P. Melville, and F. Provost. A unified approach to active dual supervision for labeling features and examples. In *Machine Learning and Knowledge Discovery in Databases*, volume 6321 of *Lecture Notes in Computer Science*, pages 40–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[5] K. Kummamuru B. Mandhani, S. Joshi. A matrix density based algorithm to hierarchically co-cluster documents and words. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 511–518. ACM, New York, NY, USA, 2003.

[6] M. Balcan and A. Blum. Clustering with interactive feedback. In *Algorithmic Learning Theory*, volume 5254 of *Lecture Notes in Computer Science*, pages 316–328. Springer Berlin Heidelberg, 2008.

[7] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, 2007.

[8] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using Wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 787–788. ACM, New York, NY, USA, 2007.

[9] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[10] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the SIAM International Conference on Data Mining*, pages 333–344, 2004.

[11] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 59–68. ACM, New York, NY, USA, 2004.

[12] J. R. Bellegarda. Unsupervised document clustering using multi-resolution latent semantic density analysis. In *Proceedings of 2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 361–366, Aug 2010.

[13] J. Bezdek. *Pattern recognition with fuzzy objective functions*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[14] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the IEEE International Conference on Data Mining*, volume 4, pages 19–26, 2004.

[15] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first International Conference on Machine Learning*, ICML '04, pages 11–18. ACM, New York, NY, USA, 2004.

[16] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 245–250. ACM, New York, NY, USA, 2001.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[18] A. Cardoso-Cachopo. Improving Methods for Single-label Text Categorization. PhD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa, 2007.

[19] D. Carmel, H. Roitman, and N. Zwerdling. Enhancing cluster labeling using Wikipedia. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 139–146. ACM, New York, NY, USA, 2009.

[20] H. C. Chang and C. C. Hsu. Using topic keyword clusters for automatic document clustering. *IEICE Transactions on Information and Systems*, 88(8):1852–1860, 2005.

[21] Y. Chang and J. Chien. Latent Dirichlet learning for document summarization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, pages 1689–1692, April 2009.

[22] K. Chaudhuri, Sh. M. Kakade, K. Livescu, and K. Sridharan. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 129–136. ACM, 2009.

[23] H. Cho, I.S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. pages 114–125, 2004.

[24] J. Chuang, C. D. Manning, and J. Heer. Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 74–77, New York, NY, USA, 2012. ACM.

[25] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 4(1):17–25, 2003.

[26] V. Cormack, J. Hidalgo, and E. Sánz. Feature engineering for mobile SMS spam filtering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 871–872. ACM, New York, NY, USA, 2007.

[27] S. P. Crain, K. Zhou, S. Yang, and H. Zha. Dimensionality reduction and topic modeling: From latent semantic indexing to latent Dirichlet allocation and beyond. In *Mining Text Data*, pages 129–161. Springer US, 2012.

[28] S. Dasgupta and V. Ng. Towards subjectifying text clustering. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 483–490. ACM, New York, NY, USA, 2010.

[29] M. Dash and H. Liu. Feature selection for clustering. In Terano, Takao, Liu, Huan, Chen, and ArbeeL.P., editors, *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 110–121. Springer Berlin Heidelberg, 2000.

[30] I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the fc-means algorithm. In *Proceedings of the fifth SIAM International Conference on Data Mining*, pages 138–149, 2005.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[32] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[33] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977.

[34] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 89–98. ACM, New York, NY, USA, 2003.

[35] W. Dou, X. Wang, R. Chang, and W. Ribarsky. Paralleltopics: A probabilistic approach to exploring document collections. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 231–240, October 2011.

[36] G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 595–602. ACM, New York, NY, USA, 2008.

[37] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 226–231, 1996.

[38] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. *Software: Practice and Experience*, 38(2):189–225, 2008.

[39] L. Galavotti, F. Sebastiani, and M. Simi. Experiments on the use of feature selection and negative evidence in automated text categorization. In *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, pages 59–68. Springer-Verlag, London, UK, 2000.

[40] M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger. Pulse: Mining customer opinions from free text. In *Advances in Intelligent Data Analysis VI*, volume 3646 of *Lecture Notes in Computer Science*, pages 121–132. Springer Berlin Heidelberg, 2005.

[41] M. J. Gardner, J. Lutes, J. Lund, J. Hansen, D. Walker, E. Ringger, and K. Seppi. The topic browser: An interactive tool for browsing topic models. In *NIPS Workshop on Challenges of Data Visualization*, 2010.

[42] S. Ghosh and S. K. Dubey. Comparative analysis of k-means and fuzzy c-means algorithms. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4:35–38, 2013.

[43] C. Gorg, L. Zhicheng, J. Kihm, J. Choo, H. Park, and J. Stasko. Combining computational analyses and interactive visualization for document exploration and sensemaking in jigsaw. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1646–1663, October 2013.

[44] H. Lee, J. Kihm, J. Choo, J. Stasko and H. Park. iVisClustering: An interactive visual document clustering via topic modeling. *Computer Graphics Forum*, 31(3pt3):1155–1164, 2012.

[45] J. Hansohm. Two-mode clustering with genetic algorithms. In *Classification, Automation, and New Media*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 87–93. Springer Berlin Heidelberg, 2002.

[46] F. Heimerl, S. Lohmann, S. Lange, and T. Ertl. Word cloud explorer: Text analytics based on word clouds. In *47th Hawaii International Conference on System Sciences (HICSS 2014)*, pages 1833–1842, Jan 2014.

[47] B. Hewitson and R. Crane. Self-organizing maps: applications to synoptic climatology. *Climate Research*, 22(1):13–26, 2002.

[48] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57. ACM, New York, NY, USA, 1999.

[49] A. Hotho, S. Staab, and G. Stumme. Ontologies improve text document clustering. In *Third IEEE International Conference on Data Mining (ICDM 2003)*, pages 541–544, Nov 2003.

[50] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39(2):133–155, March 2009.

[51] J. Hu, L. Fang, Y. Cao, H. Zeng, H. Li, Q. Yang, and Z. Chen. Enhancing text clustering by leveraging Wikipedia semantics. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 179–186. ACM, New York, NY, USA, 2008.

[52] X. Hu, X. Zhang, C. Lu, E. K. Park, and X. Zhou. Exploiting Wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 389–396. ACM, New York, NY, USA, 2009.

[53] Y. Hu, E. E. Milios, and J. Blustein. Interactive feature selection for document clustering. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1143–1150. ACM, New York, NY, USA, 2011.

[54] Y. Hu, E. E. Milios, and J. Blustein. Enhancing semi-supervised document clustering with feature supervision. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 929–936. ACM, New York, NY, USA, 2012.

[55] Y. Hu, E. E. Milios, J. Blustein, and S. Liu. Personalized document clustering with dual supervision. In *Proceedings of the 2012 ACM symposium on Document engineering*, DocEng '12, pages 161–170. ACM, New York, NY, USA, 2012.

[56] A. Huang. Similarity measures for text document clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, pages 49–56, 2008.

[57] A. Huang, D. Milne, E. Frank, and I. H. Witten. Clustering documents with active learning using Wikipedia. In *Eighth IEEE International Conference on Data Mining (ICDM '08)*, pages 839–844, Dec 2008.

[58] A. Huang, D. Milne, E. Frank, and I. H. Witten. Clustering documents using a Wikipedia-based concept representation. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '09, pages 628–636. Springer-Verlag, Berlin, Heidelberg, 2009.

[59] Y. Huang and T. Mitchell. Text clustering with extended user feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 413–420. ACM, New York, NY, USA, 2006.

[60] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[61] I. A. Islam, E. Milios, and V. Kešelj. Text similarity using Google tri-grams. In *Advances in Artificial Intelligence*, volume 7310 of *Lecture Notes in Computer Science*, pages 312–317. Springer, Berlin, Heidelberg, 2012.

[62] M. Iwayama and T. Tokunaga. A probabilistic model for text categorization: Based on a single random variable with multiple values. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, ANLC '94, pages 162–167, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[63] A. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, June 2010.

[64] A. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233 – 1244, 1996.

[65] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 307–314, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[66] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.

[67] J. Kogan, C. Nicholas, and V. Volkovich. Text mining with information-theoretic clustering. *Computing in Science and Engineering*, 5(6):52–59, November 2003.

[68] T. Kolda and D. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transaction on Information Systems*, 16(4):322–346, October 1998.

[69] H. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transaction on Knowledge Discovery from Data (TKDD)*, 3(1):1–58, 2009.

[70] R. Krishnapuram, A. Joshi, and L. Yi. A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering. In *Proceedings of the IEEE International Fuzzy Systems Conference*, volume 3, pages 1281–1286, 1999.

[71] A. Kumar and H. Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 393–400, 2011.

[72] B. Larsen and Ch. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 16–22. ACM, New York, NY, USA, 1999.

[73] J. Lee, J. Han and K. Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 593–604. ACM, New York, NY, USA, 2007.

[74] S. Liu, M. X. Zhou, S. Pan, Y. Song, W. Qian, W. Cai, and X. Lian. Tiara: Interactive, topic-based visual text summarization and analysis. *ACM Transactions on Intelligent Systems and Technology*, 3(2):25:1–25:28, February 2012.

[75] T. Liu, S. Liu, Z. Chen, and W.Y. Ma. An evaluation on feature selection for text clustering. In *Proceedings of the In Proc. 20th International Conference on Machine Learning (ICML'03)*, pages 488–495, 2003.

[76] S. Luke and L. Panait. A comparison of bloat control methods for genetic programming. *Evolutionary Computation*, 14(3):309–344, September 2006.

[77] J. MacGlashan M. desJardins and J. Ferraioli. Interactive visual clustering. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, pages 361–364. ACM, New York, NY, USA, 2007.

[78] H. Mahmoodi and E. Mansoori. Document clustering based on semi-supervised term clustering. *International Journal of Artificial Intelligence & Applications (IJAIA)*, 3(3):69–82, 2012.

[79] C. D. Manning, P. Raghavan, and H. Schütze. Flat clustering. In *Introduction to Information Retrieval*, pages 253–287. Cambridge University Press, New York, NY, USA, 2008.

[80] R.M. Marcacini, G.N. Correa, and S.O. Rezende. An active learning approach to frequent itemset-based text clustering. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 3529–3532, 2012.

[81] C. Mathwick. Understanding the online consumer: A typology of online relational norms and behavior. *Journal of Interactive Marketing*, 16(1):40–55, 2002.

[82] I. V. Mechelen, H. H. Bock, and P. De Boeck. Two-mode clustering methods: astructuredoverview. *Statistical Methods in Medical Research*, 13(5):363–394, 2004.

[83] B. L. Miller and D. E Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995.

[84] D. Milne and I. H. Witten. An open-source toolkit for mining Wikipedia. *Artificial Intelligence*, 194(0):222 – 239, 2013. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[85] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. *Proceedings of VLDB Endowment*, 2(1):1270–1281, 2009.

[86] J. M. Neuhaus and J. D. Kalbfleisch. Between- and within-cluster covariate effects in the analysis of clustered data. *Biometrics*, 54(2):638–645, 1998.

[87] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering analysis and an algorithm. *Proceedings of Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press*, 14:849–856, 2001.

[88] S. N. Nourashrafeddin, D. Arnold, and E. Milios. An evolutionary subspace clustering algorithm for high-dimensional data. In *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, GECCO Companion '12, pages 1497–1498. ACM, New York, NY, USA, 2012.

[89] S. N. Nourashrafeddin, E. Milios, and D. V. Arnold. An evolutionary algorithm for feature selective double clustering of text documents. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC'13)*, pages 446–453, Cancun, Mexico, 2013.

[90] S. N. Nourashrafeddin, E. Milios, and D. V. Arnold. Interactive text document clustering using feature labeling. In *Proceedings of the 2013 ACM Symposium on Document Engineering*, DocEng '13, pages 61–70. ACM, New York, NY, USA, 2013.

[91] S. N. Nourashrafeddin, E. Milios, and D. V. Arnold. An ensemble approach for text document clustering using Wikipedia concepts. In *Proceedings of the 2014 ACM Symposium on Document Engineering*, DocEng '14, pages 107–116. ACM, New York, NY, USA, 2014. Best Student Paper.

[92] S. Osinski and D. Weiss. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54, May 2005.

[93] M. F Porter. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3):130–137, 1980.

[94] R. Krestel, P. Fankhauser, and W. Nejdl. Latent Dirichlet allocation for tag recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 61–68. ACM, New York, NY, USA, 2009.

[95] H. Raghavan, O. Madani, and R. Jones. Interactive feature selection. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 841–846. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[96] P. Schonhofen. Identifying document topics using the Wikipedia category network. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, pages 456–462, Washington, DC, USA, 2006. IEEE Computer Society.

[97] M. M. Shafiei and E. E. Milios. Latent Dirichlet co-clustering. In *Sixth International Conference on Data Mining, 2006. ICDM '06.*, pages 542–551, 2006.

[98] G.H. Shah. An improved DBSCAN, a density based clustering algorithm with parameter selection for high dimensional data sets. In *Proceedings of 2012 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–6, Dec 2012.

[99] H. Shan and A. Banerjee. Bayesian co-clustering. In *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM '08).*, pages 530–539, Dec 2008.

[100] Q. Sheng, Y. Moreau, and B. De Moor. Biclustering microarray data by Gibbs sampling. *Bioinformatics*, 19(suppl 2):ii196–ii205, 2003.

[101] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[102] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 208–215. ACM, New York, NY, USA, 2000.

[103] G. Spanakis, G. Siolas, and A. Stafylopatis. Exploiting Wikipedia knowledge for conceptual hierarchical clustering of documents. *Computer Journal*, 55(3):299–312, March 2012.

[104] J. Sun, Z. Chen, H. Zeng, Y. Lu, C. Shi, and W. Ma. Supervised latent semantic indexing for document categorization. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04)*, pages 535–538, Nov 2004.

[105] B. Tang, M. Shepherd, E. Milios, and M. I. Heywood. Comparing and combining dimension reduction techniques for efficient text clustering. In *Proceedings of the Workshop on Feature Selection for Data Mining, in conjunction with SIAM International Conference on Data Mining*, pages 17–26, 2005.

[106] W. Tang, H. Xiong, S. Zhong, and J. Wu. Enhancing semi-supervised clustering: a feature projection perspective. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 707–716, New York, NY, USA, 2007. ACM.

[107] Y. W. Teh, D. Newman, and M. Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1353–1360, 2006.

[108] A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G. Tseng. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22(19):2405–2412, 2006.

[109] A. Vahdat, M. Heywood, and N. Zincir-Heywood. Bottom-up evolutionary subspace clustering. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2010.

[110] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[111] P. Wang, C. Domeniconi, and K. B. Laskey. Latent dirichlet bayesian co-clustering. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD '09): Part II*, ECML PKDD '09, pages 522–537. Springer-Verlag, Berlin, Heidelberg, 2009.

[112] P. Wang, C. Domeniconi, and K. B. Laskey. Information bottleneck co-clustering. In *Proceedings of the Text Mining Workshop, SIAM International Conference on Data Mining, Ohio*, 2010.

[113] P. Wang, C. Domeniconi, and K. B. Laskey. Nonparametric bayesian clustering ensembles. In *Proceedings of the 2010 European Conference on Machine Learning*

and Knowledge Discovery in Databases: Part III, ECML PKDD'10, pages 435–450. Springer-Verlag, Berlin, Heidelberg, 2010.

[114] Y. Wang, H. Bai, M. Stanton, W. Y. Chen, and E. Y. Chang. PLDA: Parallel latent Dirichlet allocation for large-scale applications. In *Algorithmic Aspects in Information and Management*, pages 301–314. Springer, 2009.

[115] X. Wei, and W.B. Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 178–185. ACM, New York, NY, USA, 2006.

[116] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[117] M. Yazdani and A. Popescu-Belis. Using a Wikipedia-based semantic relatedness measure for document clustering. In *Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing*, TextGraphs-6, pages 29–36, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[118] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.

[119] Y. Zhao and G. Karypis. Comparison of agglomerative and partitional document clustering algorithms. Technical Report 02-014, University of Minnesota, 2002.

Appendices

Appendix A

Google Ngram vs. TFIDF Cosine Plots

To improve the readability of the thesis, some of the plots of Section 5.2.3 are shown here.

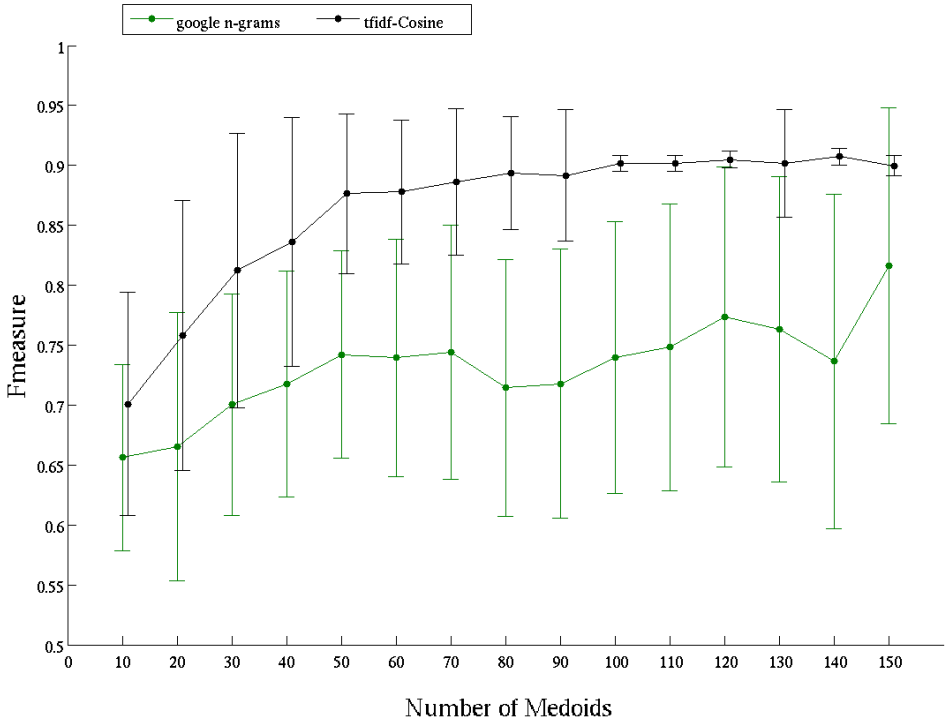

Figure A.1: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *Classic4*. Not only Cosine similarity results in better clusters, its standard deviations decrease as the number of medoids increases.

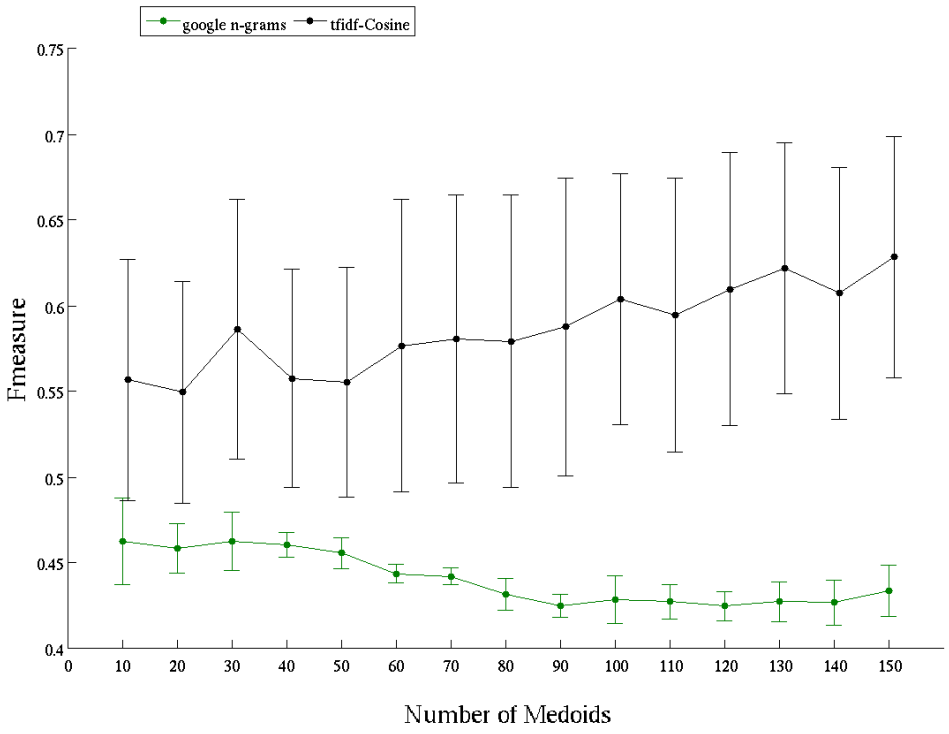

Figure A.2: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *News-sim3*. Cosine similarity generates better clusters, but the standard deviations are much greater than those of Google Ngrams based distances.

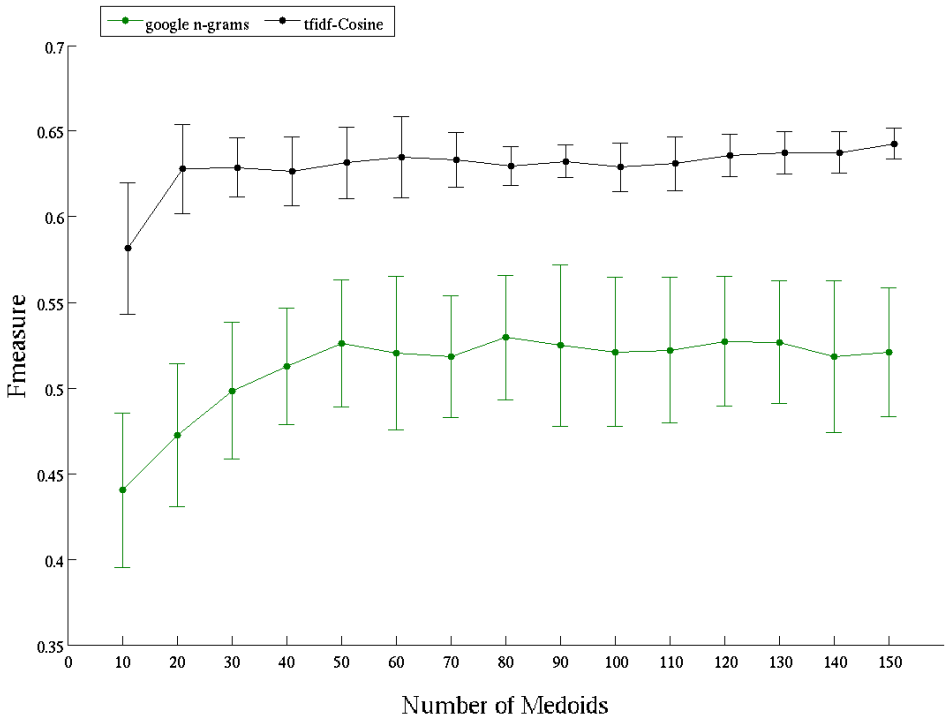

Figure A.3: The quality of clusters obtained from *LDC* using fuzzy *c*-medoids on *LA Times*. Cosine similarity generates better clusters and its standard deviations are smaller.

Appendix B

Fuzzy c-means vs. Fuzzy c-medoids Plots

To improve the readability of the thesis, some of the plots of Section 5.2.4 are shown here.

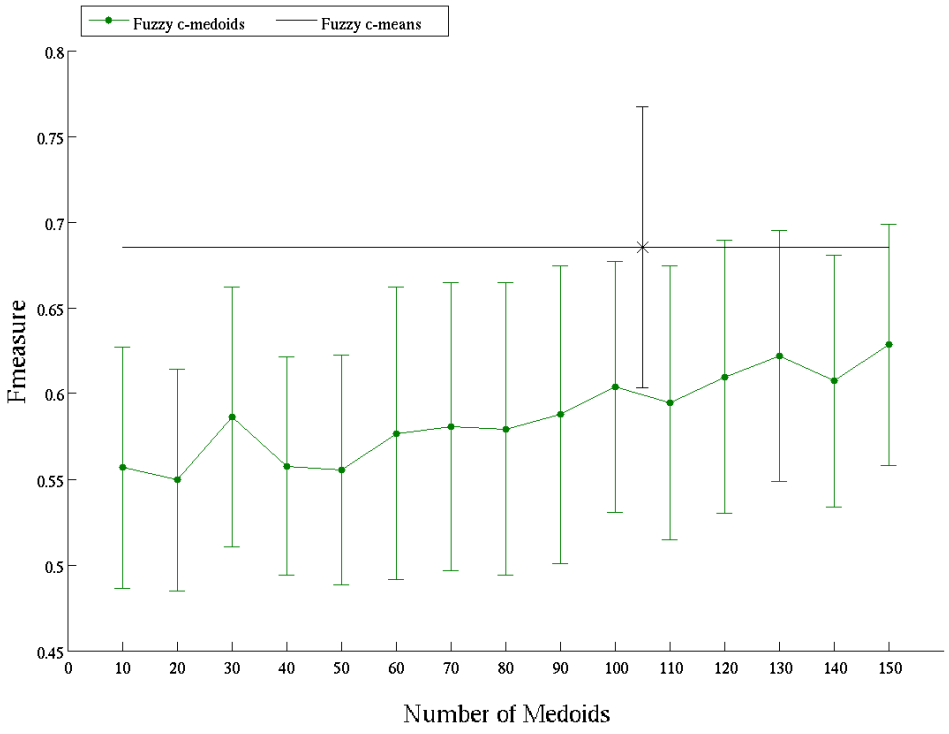

Figure B.1: The quality of clusters obtained from *LDC* using fuzzy *c*-means and fuzzy *c*-medoids on *News-sim3*. The fuzzy *c*-means clusterer outperforms fuzzy *c*-medoids. The standard deviations are mostly similar for different number of medoids.

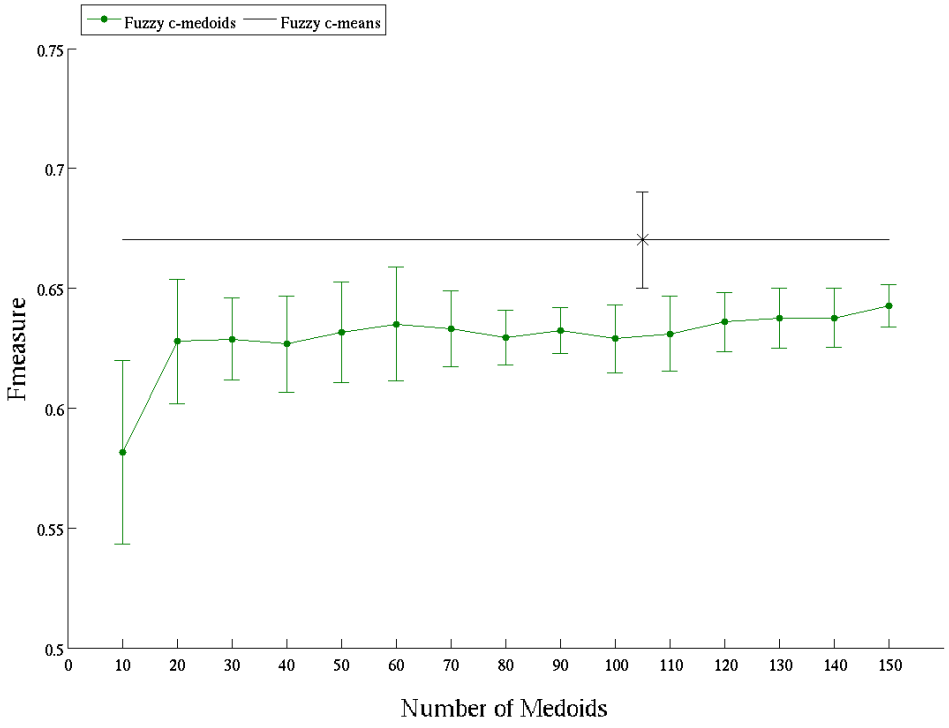

Figure B.2: The quality of clusters obtained from *LDC* using fuzzy *c*-means and fuzzy *c*-medoids on *LA Times*. The result of fuzzy *c*-means is slightly better and its standard deviations are small.

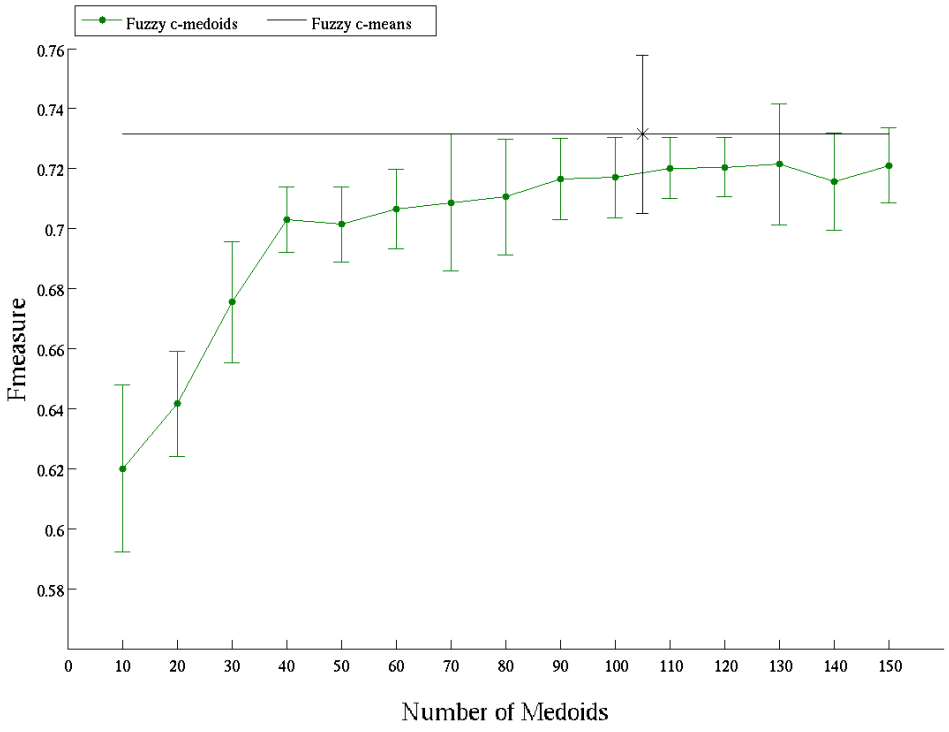

Figure B.3: The quality of clusters obtained from *LDC* using fuzzy c-means and fuzzy c-medoids on *News-multi10*. As the number of medoids increases to 100, the quality of clusterings obtained by using either term clusterer is similar.

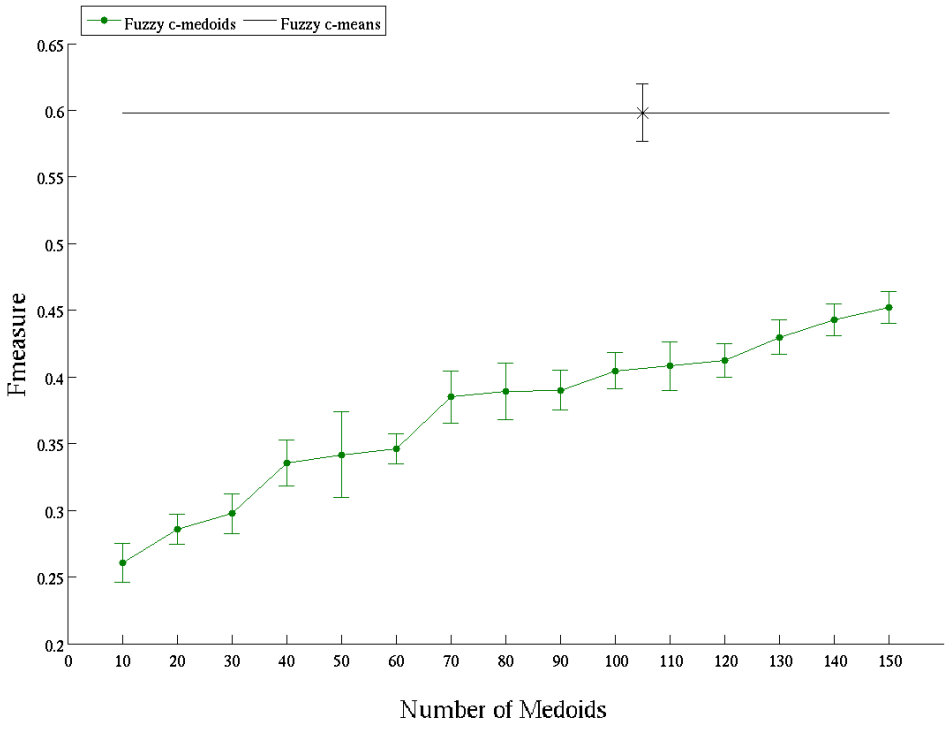

Figure B.4: The quality of clusters obtained from *LDC* using fuzzy *c*-means and fuzzy *c*-medoids on *Newsgroups20*. Fuzzy *c*-means outperforms fuzzy *c*-medoids and the differences among the quality of clusterings are significant.

Appendix C

Term Labeling vs. Term Selection Plots

To improve the readability of the thesis, some of the plots of Section 7.2.1 are shown here.

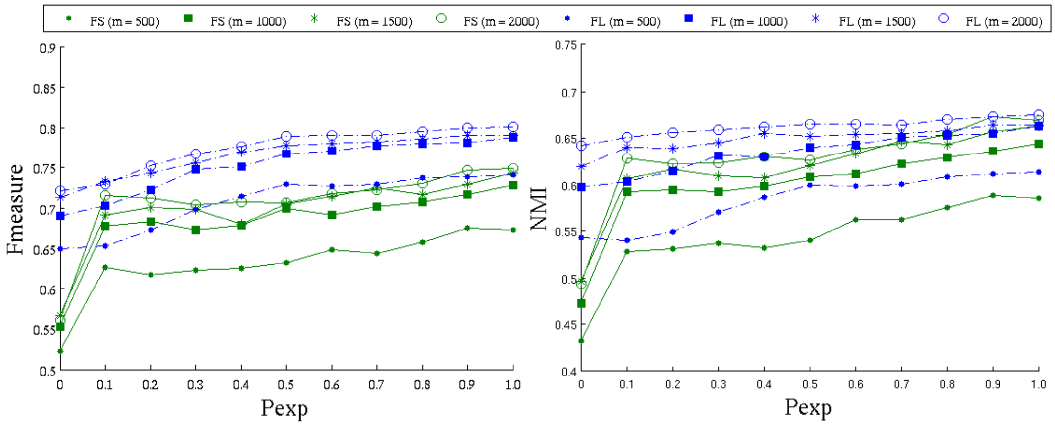

Figure C.1: The quality of clusters based on term labeling (*FL*) and term selection (*FS*) on *News-multi7*. Both term labeling and term selection methods improved the quality of clusters. Feature labeling generates better results compared to the term selection.

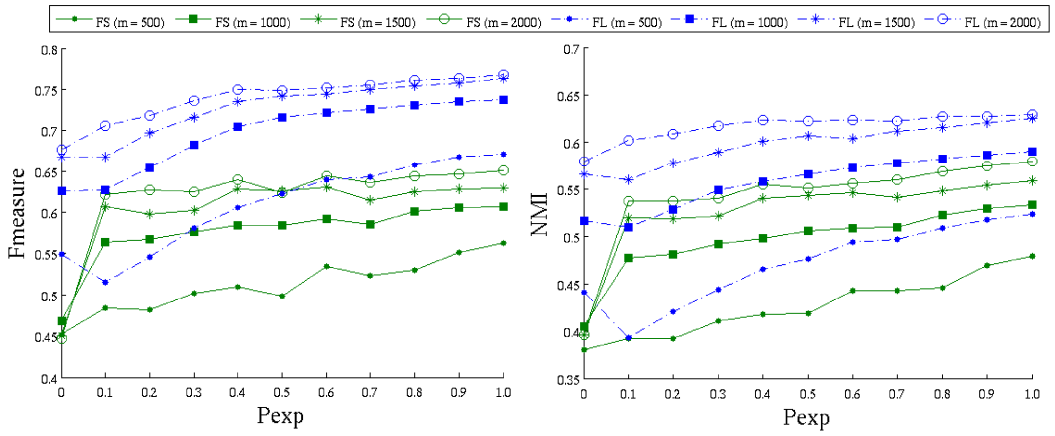

Figure C.2: The quality of clusters based on term labeling (*FL*) and term selection (*FS*) on *News-multi10*. Both term labeling and term selection methods improved the quality of clusters. Feature labeling outperformed the term selection.

Appendix D

Term- vs. Dual-Supervised LDC Plots

To improve the readability of the thesis, some of the plots of Section 7.2.3 are shown here.

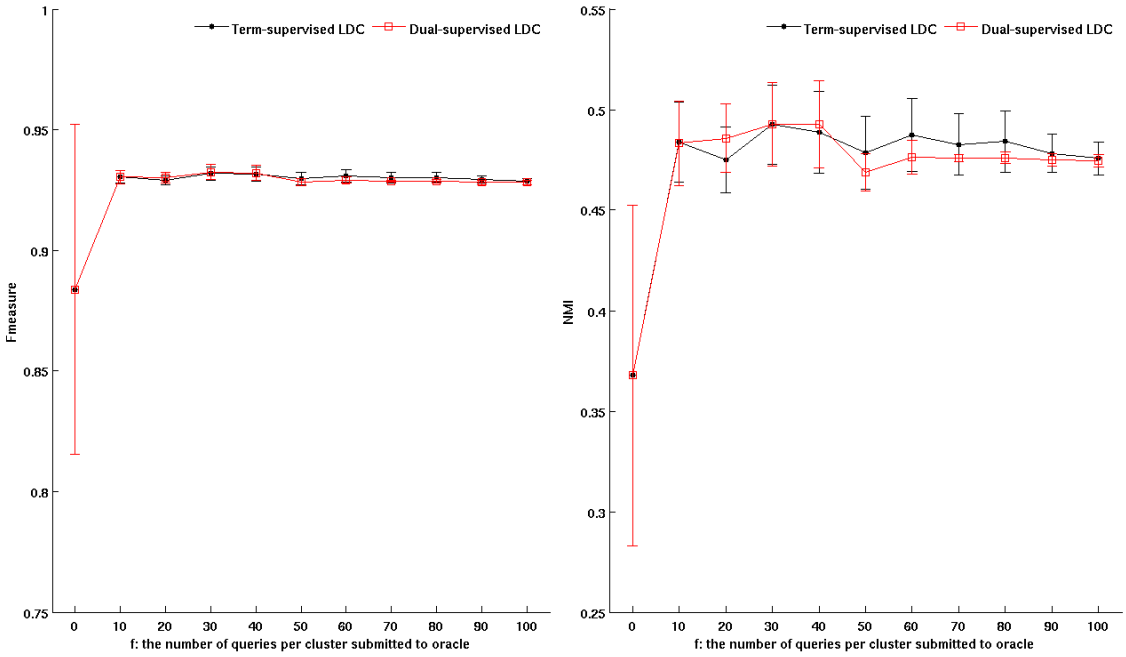

Figure D.1: The quality of clusters obtained from Term-supervised *LDC* and Dual-supervised *LDC* on *SMS*. Neither of algorithms could outperform significantly the other one.

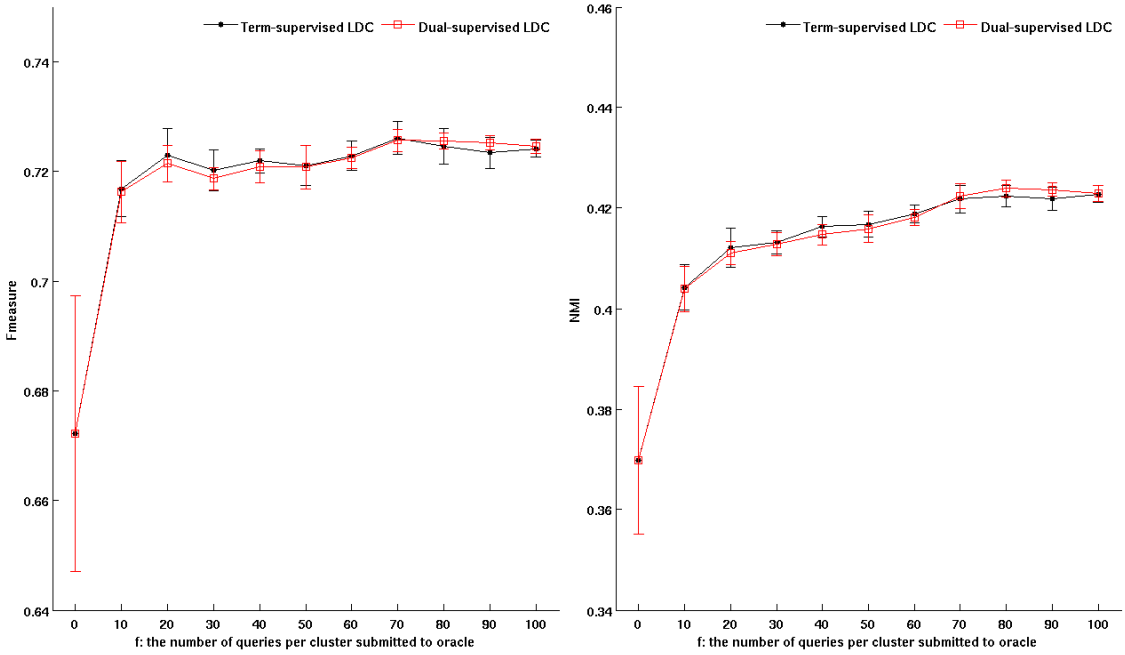

Figure D.2: The quality of clusters obtained from Term-supervised *LDC* and Dual-supervised *LDC* on *WebKB*. Neither of algorithms could outperform significantly the other one.

Appendix E

Noisy-Supervised LDC Plots

To improve the readability of the thesis, some of the plots of Section 7.2.4 are shown here.

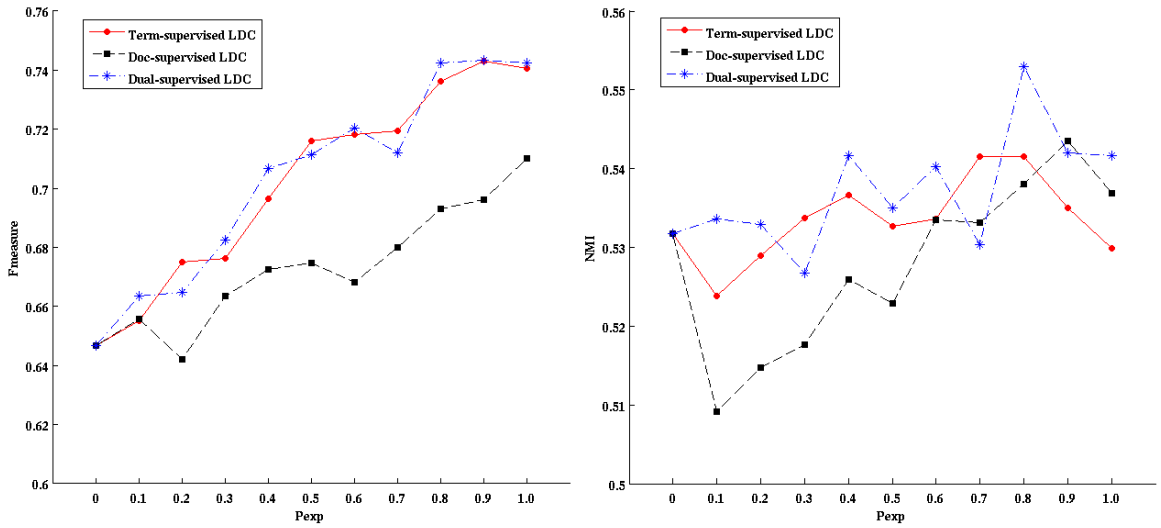

Figure E.1: The quality of clusters obtained from the user-supervised *LDC* algorithms on *Reuters8-whole* when the user's degree of expertness is variable. Document-supervised *LDC* could not generate comparable results to the other user-supervised algorithms in most cases.

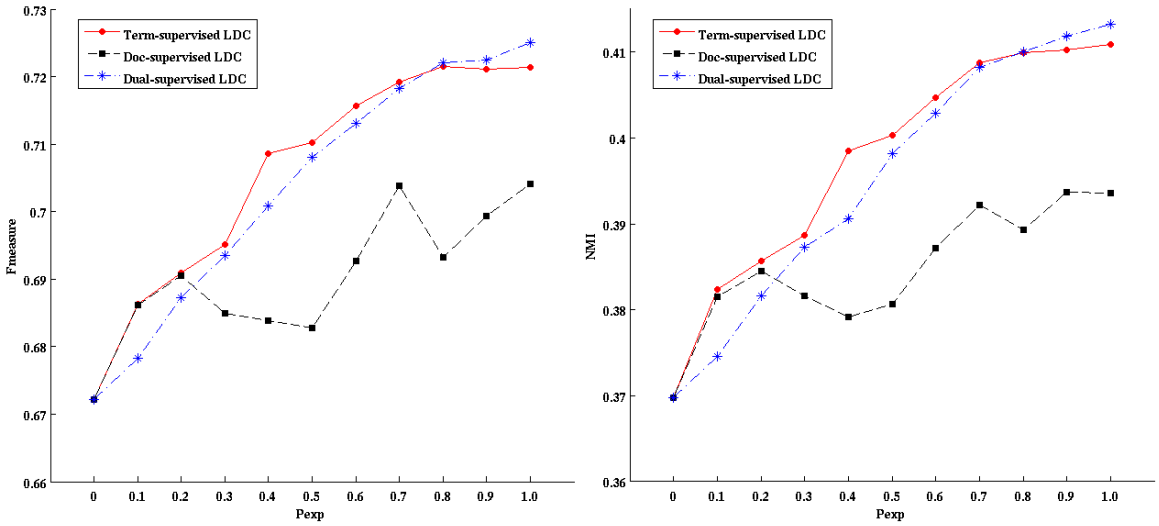

Figure E.2: The quality of clusters obtained from the user-supervised *LDC* algorithms on *WebKB* when the user's degree of expertness is variable. Document-supervised *LDC* is outperformed by the other user-supervised algorithms when the degree of user expertness is more than 0.3.