SEMANTIC FRAMEWORKS FOR DOCUMENT AND ONTOLOGY CLUSTERING

A DISSERTATION
in
Computer Science
and
Computer Networking

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by
TUANJIE TONG

M. S., Western Illinois University, 2003
M. S., Western Illinois University, 2002
M. S., Beijing Institute of Technology,China, 1996
B. S., Beijing Institute of Technology, China, 1991

Kansas City, Missouri
2010

SEMANTIC FRAMEWORKS FOR DOCUMENT AND ONTOLOGY CLUSTERING

Tuanjie Tong, Candidate for the Doctor of Philosophy Degree

University of Missouri–Kansas City, 2010

ABSTRACT

The Internet has made it possible, in principle, for scientists to quickly find research papers of interest. In practice, the overwhelming volume of publications makes this a time consuming task. It is, therefore, important to develop efficient ways to identify related publications. Clustering, a technique used in many fields, is one way to facilitate this. Ontologies can also help in addressing the problem of finding related entities, including research publications. However, the development of new methods of clustering has focused mainly on the algorithm per se, with relatively less emphasis on feature selection and similarity measures. The latter can significantly impact the accuracy of clustering, as well as the runtime of clustering. Also, to fully realize the high resolution searches that ontologies can make possible, an important first step is to find automatic ways to cluster related ontologies. The major contribution of this dissertation is an innovative semantic framework for document clustering, called Citonomy, a dynamic approach that (1) exploits citation semantics of scientific documents, (2) deals with evolving datasets

of documents, and (3) addresses the interplay between algorithms, feature selections, and similarity measures in an integrated manner. This improves accuracy and runtime performance over existing clustering algorithms. As the first step in Citonomy, we propose a new approach to extract and build a model for citation semantics. Both subjective and objective evaluations prove the effectiveness of this model in extracting citation semantics. For the clustering stage, the Citonomy framework offers two approaches: (1) CS-VS: Combining Citation Semantics and VSM (Vector Space Model) Measures and (2) CS2CS: From Citation Semantics to Cluster Semantics. CS2CS is a document clustering algorithm with a 3-level feature selection process. It is an improvement over CS-VS in several aspects: i) deleting the requirement of a training step, ii) introducing an advanced feature selection mechanism, and iii) dynamic and adaptive clustering of new datasets. Compared to traditional document clustering, CS-VS and CS2CS significantly improve the accuracy of clustering by 5-15% (on average) in terms of the F-Measure. CS2CS is a linear clustering algorithm that is faster than the common document clustering algorithms K-Means and K-Medoids. In addition, it overcomes a major drawback of K-Means/Medoids algorithms in that the number of clusters can be dynamically determined by splitting and merging clusters. Fuzzy clustering with this approach has also been investigated. The related problem of ontology clustering is also addressed in this dissertation. Another semantics framework, InterOBO, has been designed for ontology clustering. A prototype to

demonstrate the potential use of this framework, has been developed. The Open Biomedical Ontologies (OBOs) are used as a case study to illustrate the clustering technique used to identify common concepts and links. Detailed experimental results on different data sets are given to show the merits of the proposed clustering algorithms.

This abstract of 452 words is approved as to form and content.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a dissertation titled “Semantic Frameworks for document and ontology clustering” presented by Tuanjie Tong, candidate for the Doctor of Philosophy degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Yugyung Lee, Ph.D., Committee Chair
Department of Computer Science Electrical Engineering

Baek-Young Choi, Ph.D.
Department of Computer Science Electrical Engineering

Deendayal Dinakarpandian, Ph.D.
Department of Computer Science Electrical Engineering

Vijay Kumar, Ph.D.
Department of Computer Science Electrical Engineering

Deep Medhi, Ph.D.
Department of Computer Science Electrical Engineering

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ACKNOWLEDGEMENTS

As my Ph.D. study is ending, I extend my gratitude to the people who have been helping me go through the most important and difficult period of my life. First, I would like to thank my advisor, Dr. Yugyung Lee, for her wonderful advising and guidance, and provision of a nice environment. I do not believe I could have accomplished what I have done without Dr. Lee's help. I would also like to thank all the members of my advisory committee, including Dr. Baek-Young Choi, Dr. Deendayal Dinakarpandian, Dr. Vijay Kumar, and Dr. Deep Medhi, for their willingness to accept my invitation, and to find time to provide important input during the process of my research and dissertation writing. I really learned a lot from each of them that will surely benefit my future career.

Lastly, but never the least, I would like to thank my wife, Xuemei Lu, for her wholehearted support and patience in waiting for me to finish my Ph.D. study, my kids, Kevin and Hannah, for the inspiration and comfort I obtained from them, and my parents for their encouragement and support during my Ph.D. study.

CHAPTER 1

INTRODUCTION

Recently, researchers in scientific communities have witnessed the tremendous growth of publications. Even though search engines on the Internet provide the efficient way for researchers to find publications of interests, the overwhelming amount of information still makes it a time-consuming task. Clustering, an important technique used in many fields such as knowledge discovery and information retrieval, can help researchers find related information more quickly and thus, keep them updated with new findings in their fields.

Clustering is the process of grouping/dividing a set of objects into subsets (called clusters) so that the objects are similar to one another within the cluster and are dissimilar to objects in other clusters regarding some selected features of these objects. In other words, an object is closer to at least one object in the same cluster than any objects in other clusters in terms of the predefined distance or similarity measure. Document/Text clustering is a specific clustering technique where objects to be clustered are documents/texts.

Considering features used in document/text clustering, the document/text clustering algorithms can be divided into two classes – those that use vector space and those that use frequent terms. The vector space clustering creates a vector for each document where each dimension represents a term in that document; the value of each dimension or the

weight of each term is usually calculated with TF-IDF (Term Frequency-Inverse Document Frequency). Then the clustering algorithms compute the distances of two vectors to determine clustering. The frequent terms clustering algorithm first finds frequent term sets using association rule mining, then uses the mutual overlap of the frequent term sets with respect to their sets of supporting documents to determine clustering. It is intended to solve the high dimensionality problem of vector space clustering.

An ontology is an explicit specification of a conceptualization in a particular domain. Its importance in knowledge management, knowledge sharing and information retrieval has been realized by researchers, especially in biological and biomedical domains, where new discoveries and knowledge emerge at a fast pace. Many different ontologies have been developed in recent years. Whereas each ontology is useful for a particular domain or subdomain, the interoperability between these ontologies has yet to be built up.

1.1 Problem Definition

First, in both classes of document clustering algorithms mentioned above, all words or terms in the document are treated equally. In other words, the context or semantics or words are not taken into consideration in clustering, even in the case of scientific documents. By doing this, the significance of some words or terms in a scientific document, such as references, titles, and keywords, were ignored. That results in a lower accuracy of clusters. Some surveys on document clustering algorithms have shown that these algorithms can hardly achieve higher than 73% (on average) regarding the accuracy

of resulted clusters.

Secondly, due to the hight dimensionality of the vectors in vector space model which is used in most document clustering algorithms, the process of clustering is usually slow. Even though the approach of using frequent terms reduces the dimensionality, the step of finding frequent terms is computationally costly and hence, the entire process of this approach is not fast either.

Thirdly, the traditional document clustering algorithms tend to focus on the process of clustering, and pay less attention to the feature selection and similarity measure process. However, both of them can significantly affect the quality and runtime of a clustering algorithm.

To solve these problems, we propose a semantic framework, called Citonomy. In this framework, we consider the semantic information such as citations, titles, and keywords, in document clustering. They are like gold buried in sand. We assume that, if this hidden gold is explored in designing a document clustering algorithm, it will produce clusters with higher accuracy. Two approaches of Citonomy are fully discussed in this dissertation. The first approach, CS-VS, combining citation semantics and vector space measures, utilizes this information by calculating and combining two similarities between two documents. In CS-VS, we pay much attention to the issue of similarity measure. We also use the evolution strategy to train the system. The limitation of CS-VS is that its runtime complexity is high. The second approach, CS2CS, citation semantics to cluster semantics, utilizes the semantic information by considering it in constructing document feature vectors. In CS2CS, we use a 3-level feature selection process with a 2-dimensional

normalization to extract significant features of documents and clusters. Not only does CS2CS solve overcome the runtime problem, but it also produces clusters with higher quality. In addition, domain knowledge was also utilized in the process of document clustering with a domain ontology.

In terms of ontologies, many domain ontologies have been developed in recent years. To use them effectively, we first need to know the relation or mapping between them. The current ontology mapping approaches have not covered every aspects of mapping. For example, to our knowledge, no one has done clustering over ontologies to explore their relations. In this dissertation, we propose a semantic framework with a clustering technique to find the relations between ontologies. Also, to keep up with the growth of a domain knowledge, the ontology of that domain needs to be updated frequently. In this dissertation, we demonstrated that ontology and our document clustering algorithms benefit each other. On one hand, we utilize ontology to improve the document clustering results. On the other hand, the feature vectors of resulted clusters can help update ontology.

1.2 Contributions of this Dissertation

The major contributions of this dissertation are as follows:

1. It is the first time that citation semantics is utilized in document clustering.
2. A semantic framework, Citonomy, is proposed. it includes a citation semantics extraction model and two approaches.

3. A model, CSE, Citation Semantics Extraction, for reference clustering and labeling, together with formulas for similarity measure between reference clusters are proposed.
4. CS-VS, combining citation semantics and vector space similarity measure for document clustering is designed. It offers a significant improvement over traditional document clustering. In CS-VS,
   (a) The similarity issue between documents is thoroughly explored.
   (b) A system training model utilizing an evolution strategy is designed to find the optimal similarity weights.
5. CS2CS, citation semantics to cluster semantics, is designed to utilize the citation semantics by considering them in forming feature vectors. It involves a 3-level feature selection model with a 2-dimensional normalization process.
   (a) CS2CS can do realtime clustering over evolving datasets of documents.
   (b) CS2CS can determine the number of clusters dynamically by cluster splitting and merging.
   (c) CS2CS is not limited to scientific documents. It also outperformed traditional document clustering algorithms without using the semantics of the documents.
   (d) CS2CS based fuzzy clustering algorithm is also proposed and the results are promising too.
   (e) Methods of using ontology in document clustering and updating ontology with document clustering results are proposed.
6. A semantic framework, InterOBO, is proposed for ontology mapping and clustering

1.3 Outline of this Dissertation

The rest of this dissertation is organized as follows: Chapter 2 covers the review over the related literature. Chapter 3 presents Citonomy, which is the overall framework of utilizing citation semantics in document clustering. It is followed by discussions on two approaches of Citonomy – CS-VS and CS2CS that are in Chapter 4 and Chapter 5, respectively. Chapter 6 shows InterOBO that is the framework of knowledge sharing between ontologies. The detailed experimental results of CS-VS, CS2CS, and InterOBO are displayed and discussed in Chapter 7. Finally, the summary and discussion on future work are included in Chapter 8.

CHAPTER 2

REVIEW OF LITERATURE

In this dissertation, algorithms in document clustering and ontology clustering are discussed. Before unfolding these discussions, we do a review on clustering, document clustering, feature selection, and ontologies. Since one of the major contributions of this dissertation is the use of citation in clustering, we also review the use of citation and existing research topics on citation.

2.1 Clustering

Clustering is the process of grouping/dividing a set of objects into subsets (called clusters) so that the objects are similar to one another within the cluster and are dissimilar to objects in other clusters regarding some selected features of these objects. Clustering is a method of unsupervised classification. It is a common technique of statistical data analysis used in many fields and applications such as biology, geology, medicine, market research, educational research, social network analysis, image segmentation, data mining, and so on.

The process of clustering typically involves the following steps [63]: (1) object representation (optionally feature extraction and /or selection), (2) definition of distance/similarity measure, (3) clustering or grouping, and (4) data abstraction or labeling (optional).

Object representation is the step of selecting features to represent objects to be clustered. Feature selection and/or feature extraction are usually used in this step. Feature selection is the process of identifying the most effective subset of the original features to be used in clustering. Feature extraction is the process of using linear or non-linear transformations on original features to generate projected features to be used in clustering. Both could reduce the dimensionality of features.

Definition of distance/similarity measure is the step of defining a proper distance/similarity measure to characterize the conceptual distance/similarity between objects. Different distance/similarity measures are used in different situations. For example, to cluster points in a two- or three-dimensional space, the Euclidean distance is usually used, while in document clustering with the vector space model, the cosine coefficient similarity is commonly adopted.

Clustering or grouping is the step of assigning the objects to different clusters (or subsets, or groups). It is the major step of the entire clustering process. Different clustering algorithms usually differ at this step. In terms of the relation of objects and resulting clusters, clustering algorithms could be categorized as hard (an object belongs to only one cluster) and fuzzy (an object belongs to multiple clusters each with a degree of membership). In terms of the structure of resulting clusters, clustering algorithms could be hierarchical or partitional. A hierarchical clustering algorithm produces a nested series of partitions based on a criterion of merging or splitting clusters with a given distance/similarity measure. A partitional algorithm partitions the objects into groups at the

same level with a clustering criterion optimized (usually locally). Other clustering algorithms include Model-based such as SOM (Self-organizing Map, [66]) that is based on an artificial neural network [60] and graph-based such as [102] and [47].

Data abstraction or labeling is the step to extract brief representations for resulted clusters. They are compact descriptions or a summary of clusters.

Whereas clustering could be used in many fields, we will focus on its use in document management, namely, document/text clustering. The following section is thus dedicated to the review on document/text clustering.

2.2 Document/Text Clustering

Document clustering is the process of grouping a set of documents into clusters so that the documents within each cluster are similar to each other, in other words, they belong to the same topic or subtopic, while documents in different clusters belong to different topics or subtopics. A document clustering algorithm is typically dependent on the use of a pair-wise distance measure between the individual documents to be clustered. The vector space model (VSM) [90] is commonly used for the distance measure in document clustering. Each document is represented by a vector of frequencies of terms after removing stop words and word stemming (reducing a word to its canonical form). In practice, the term frequency is usually the weighted frequency, e.g., TF-IDF (term frequency-inverse document frequency). That is, in the VSM model, the documents in a collection are converted into vectors in vector space:

$$D = \{d_1, d_2, ..., d_n\} \rightarrow M = \{\vec{v_1}, \vec{v_2}, ..., \vec{v_n}\}. \tag{2.1}$$

Where n the number of documents, $\vec{v_j}, j = 1, ..., n$ is defined as the following equation:

$$\vec{v_j} = (TF - IDF_{1,j}, TF - IDF_{2,j}, ...TF - IDF_{m,j}) \tag{2.2}$$

Where m is the number of unique terms in the set of documents to be clustered, and $TF - IDF_{i,j}$ is calculated through the following three equations:

$$TF - IDF_{i,j} = tf_{i,j} \times idf_i \tag{2.3}$$

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2.4}$$

$$idf_i = log \frac{|D|}{|\{d : t_i \in d\}|} \tag{2.5}$$

Where $n_{i,j}$ is the number of occurrences of the considered term t_i in document d_j, and the denominator is the sum of the number of occurrences of all terms in document d_j. This formula is used instead of a simple term count to prevent a bias towards longer documents. $|D|$ is the total number of documents in the corpus, and $|\{d : t_i \in d\}|$ is the number of documents where the term t_i appears.

The idea of combining IDF with TF is that if a term is highly frequent across different documents, then it would have little discriminating power, and vice versa [89].

To compute the similarity between two documents, the corresponding vector representations are used with measures like the inner product, dice coefficient, or cosine coefficient.

All the general purpose clustering algorithms can be applied to document/text clustering. Some algorithms have been developed solely for document/text clustering. All these algorithms can be classified into partitional, hierarchical, and others such as probabilistic, graph-based, and frequent term-based.

Partitional clustering attempts to break the given data set into k disjoint classes such that the data objects in a class are nearer to one another than the data objects in other classes. The most well-known and commonly used partitional clustering algorithm is K-Means([59]), as well as its variances Bisecting K-Means ([49]) and K-Medoids ([64]).

Hierarchical clustering proceeds successively by building a tree of clusters. There are two types of hierarchical clustering methods: agglomerative and divisive. Agglomerative hierarchical clustering is a bottom-up strategy that starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until a user-defined criterion is met. Divisive hierarchical clustering is a top-down strategy that starts with all objects in one cluster. It divides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until certain termination conditions are satisfied. In terms of the distance/similarity measure, a hierarchical clustering could use minimum distance (single-link) [95], maximum distance (complete-link) [65], mean distance, or average distance.

Model-based clustering algorithms try to optimize the fit between the given data and some mathematical model under the assumption that the data are generated by a mixture of underlying probability distributions. SOM [66] is one of the most popular model-based algorithms that uses neural network methods for clustering. It represents all points in a high-dimensional space by points in a low-dimensional (2-D or 3-D) target space, such that the distance and proximity relationship are preserved as much as possible. It assumes that there is some topology or ordering among input objects and that the points will eventually take on this structure in the target space.

Graph-based clustering algorithms apply graph theories to clustering. A well-known graph-based divisive clustering algorithm [102] is based on the construction of the minimal spanning tree (MST) of the data, and then deleting the MST edges with the largest lengths to generate clusters. Another popular graph-based clustering algorithm is MCL (Markov Cluster algorithm [47]). It will be discussed with more details later in this section.

Whereas there are many document/text clustering algorithms available, we only have interests in some of them in the context of this dissertation. Some surveys and comparison studies such as [96] and [101] over document/text clustering algorithms suggest that K-Means and Bisecting K-Means algorithms perform better than other clustering algorithms in document/text clustering. Therefore, in this dissertation, we compare the performance of our algorithms to that of K-Means and Bisect K-Means. In addition, our CS-VS approach (4) is based on K-Medoids, a variance of K-Means clustering algorithm. In the following subsections, we are going to review these algorithms and works that are closely related to our work.

2.2.1 K-Means Clustering Algorithm

The K-Means clustering algorithm partitions a set of objects into k clusters (k is provided) so that the resulted intra-cluster similarity is high but the inter-cluster similarity is low. It starts by randomly selecting k objects as the initial means. Each of the other remaining objects is then assigned to one of these k means of cluster to which it is the most similar. The means of clusters are updated after all objects are assigned. The process

iterates until the criterion function converges. Typically, the following criterion is used:

$$E = \sum_{i=1}^{k} \sum_{o \in C_i} |o - m_i|^2 \tag{2.6}$$

Where E is the sum of the square error for all objects in the data set, k is the number of clusters, o is the representation of a given object, and m_i is the mean of cluster C_i. The complete algorithm follows:

(1) Choose k objects as initial cluster means (or centers)

(2) Repeat

(3) assign each remaining object to the cluster to which the object is the most similar based on the mean of the cluster

(4) update the cluster means, i.e., calculate the mean value of the objects in each cluster

(5) until there is no change in any cluster

The runtime complexity of this algorithm is $O(nkt)$, where n is the number of objects, k is the number of clusters, and t is the number of iterations. Normally, $k << n$ and $t << n$. The method often terminates at a local optimum. It is sensitive to noise data since a small number of such data can substantially influence the mean value and hence affect the quality of resulted clusters. The following algorithm, K-Medoids clustering algorithm, can be used to replace the K-Means to reduce the sensitivity to noise.

2.2.2 K-Medoids Clustering Algorithm

The K-Medoids clustering algorithm is a variance of the K-Means algorithm. Instead of finding the mean value of the objects in a cluster as a reference point, it uses an

actual object as the center (called medoid) of each cluster. The remaining objects are then assigned to these clusters represented by these medoids based on their similarities with the medoids. The process terminates as the following criterion converges:

$$E = \sum_{i=1}^{k} \sum_{o \in C_i} |o - m_i|^2 \tag{2.7}$$

Where E is the sum of the square error for all objects in the data set, k is the number of clusters, o is the representation of a given object, and m_i is the medoid of cluster C_i. The complete algorithm follows:

(1) Choose k objects as initial cluster medoids (or centers)

(2) Repeat

(3) assign each remaining object to the cluster with the nearest medoid

(4) for each medoid m

(5) for each non-medoid object o

(6) Swap m and o and compute the total cost of the configuration

(7) Select the configuration with the lowest cost

(8) until there is no change in any cluster

The runtime complexity of this algorithm is $O(k(n-k)^2 t)$, where n is the number of objects, k is the number of clusters, and t is the number of iterations. Obviously, it is not as scalable as the K-Means algorithm. However, the K-Medoids algorithm is desirable when the mean of a cluster cannot be defined, such as when categorical attributes (or features) are involved, or the insensitivity to noise is a major concern.

2.2.3 Bisecting K-Means Clustering Algorithm

The bisecting K-Means is a simple version of K-Means algorithm. It starts with a single cluster of all the objects and continually splits a (chosen) cluster using K-Means with $k = 2$, until the desired number of k is reached. The complete algorithm follows:

(1) Repeat

(2) Pick a cluster to split

(3) Split the chosen cluster into two using K-Means

(4) until the k clusters are produced

Steinbach et al. in [96] state that there is not a big difference between the possible methods for selecting a cluster to split and choosing the largest remaining cluster to split. Step 2 involves using K-Means clustering algorithm which is reviewed in Subsection 2.2.1 . The runtime complexity of this algorithm in terms of the number of objects n is $O(n)$.

2.2.4 MCL

The MCL (Markov Cluster algorithm [47]) is a graph-based clustering algorithm. It is based on the graph clustering paradigm that if there are *natural* clusters in a graph, then they have the following property: *A random walk in the graph that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited.* The idea of MCL is to simulate flow within a graph, to promote flow where the current is strong, and to demote flow where the current is weak. If clusters are present in the graph, then the current across borders between different clusters will wither away, thus revealing the clusters in the graph. The complete MCL algorithm is as follows:

(1) Given an adjacency matrix M representing a weighted graph along with $\{e_i\}_{i=1}^{\infty}$ *and* $\{r_i\}_{i=1}^{\infty}$

(2) Let $T_1 = M'$

(3) Repeat

(4) $T_{2k} = (T_{2k-1})^{e_i}$

(5) $T_{2k+1} = \gamma_{r_k}(T_{2k})$

(6) *k=k+1*

(7) until T_{2k+1} *is a (near-)idempotent matrix that contains the clusters*

Where $e_i \in N$ and $e_i > 1, i = 1, 2, ...$, $r_i \in R$ and $r_i > 0, i = 1, 2, ...$; M' is a column-normalized M, that is, the element at the $p-th$ row and $q-th$ column, $M'_{pq} = \frac{M_{pq}}{\sum_i M_{iq}}$, γ_r is called the inflation operator with power coefficient r. It is defined as $(\gamma_r(M))_{pq} = \frac{(M_{pq}^r)}{\sum_i (M_{iq})^r}$.

The runtime of MCL is $O(n^3)$ where n is the number of nodes of the graph. However, the matrices T_i are generally very sparse, or at least the vast majority of the entries are near zero. Pruning in MCL involves setting near-zero matrix entries to zero, and can allow sparse matrix operations to improve the speed of the algorithm vastly. One advantage of MCL is that it does not need the user to provide the number of clusters that fits the situations of the references clustering and ontology clustering that will be discussed later in this dissertation. And in both situations, the numbers of nodes are ignorably small, therefore, runtime is not an issue at all.

2.2.5 Other Related Document Clustering Approaches

In [70], Larsen and Aone described a document clustering algorithm that is similar to K-Means. However, they did extra work on seed selection (selection of initial means), center adjustment by adding a damping parameter for the average function in finding the cluster mean, and cluster refinement by splitting each cluster to two then joining the closest pairs. But the authors did not compare their results with other approaches such as traditional K-Means. Nevertheless, in this paper, the authors mentioned using part of a vector in VSM model to represent a document. They used a default length of 25 and did experiments on other lengths as well, with a conclusion that the longer the vectors they used, the higher the quality of the clustering will be. It is different from the conclusion in Chapter 5 of this dissertation. We point out that at a certain point, the quality will turn worse when the vectors get longer.

In [91], Saracoglu et al. presented an algorithm for similar documents search (or document retrieval). The steps it used are similar to our CS2CS linear clustering discussed in Chapter 5. That is, it first does clustering over the existing documents, then finds the means of each cluster to represent that cluster. When an input document is presented, it will be compared to the mean of each cluster to find the cluster(s) and hence the "candidate documents," the simlarities between the input document and the "candidate documents" are then calculated to order the candidates before being returned to the user. However, we have a more delicate approach in selecting features to represent a cluster, and it is shown to be better than simply using means of clusters.

In [101], not only did Yoo and Hu do a comprehensive study and concluded that

K-Means and Bisecting K-Means perform better than other algorithms in document clustering, but they also used MeSH ([18]) in their experiments and found that it does improve the clustering quality for biomedical documents. However, they used MeSH to find semanticly similar terms and replace them by a MeSH descriptor term. In our approach, not only do we use MeSH to find similar terms, but we also increase the weights of those terms which leads to better results.

2.3 Feature Selection

The major problem with VSM [90] is the high dimensionality of vectors that makes the algorithms based on VSM computationally expensive. Feature selection can be used to reduce the dimensionality. Feature selection is a process that selects a subset of original features. Strictly speaking, feature selection is involved in every clustering algorithm. This is because to cluster a given set of objects, one needs to decide on which feature(s) of those objects the clustering is going to be conducted. The selected features are usually a subset of all the features of each object in question.

In the context of document/text clustering, stop words removal is the first step of feature selection which discards those common words such as “a” and “the”. Then IDF ([90]) could remove other common words across the data set if TF-IDF is used. Furthermore, one can use a subset of a vector in VSM to represent a document. For example, Larsen and Aone in [70] choose the top terms in a vector based on their weights computed from TF-IDF. The length of the vectors is set heuristically. This subset could also be obtained using other models or strategies instead of VSM. For example, Beil et

al. in [35] proposed a text clustering method using frequent terms. The problem is the setting of the threshold of term frequency. If it is too big, many small clusters will be overlooked, thus resulting in low clustering quality; if it is too small, frequent terms will lose their meaning.

Some other popular methods used for feature selection in the context of document/text clustering are document frequency and term strength [100], entropy-based ranking method [44], and term contribution [72]. Document frequency is the number of documents in which a term occurs in a data set. It could be considered as a simple version of TF-IDF. The term strength is computed based on the conditional probability that a term t occurs in document d_j given it occurs in document d_i, that is $P(t \in d_j | t \in d_i), d_i, d_j \in D \cap sim(d_i, d_j) > \beta$, where β is the threshold of similarities between documents. To calculate term strength of each term, one needs to find the similarity of each pair of documents and hence, the runtime complexity of this process will be $O(n^2)$, where n is the number of documents to be clustered.

The entropy-based ranking method ranks terms by the entropy reductions when they are removed. The entropy is defined as follows.

$$E(t) = -\sum_{i=1}^{n}\sum_{j=1}^{n}(S_{ij} \times log(S_{ij}) + (1 - S_{ij}) \times log(1 - (S_{ij}))) \tag{2.8}$$

Where S_{ij} is the similarity between documents d_i and d_j, and it is defined as $S_{ij} = e^{-\alpha \times dist_{ij}}$, where $dist_{ij}$ is the distance between the documents d_i and d_j after term t is removed, and $\alpha = -\frac{ln(0.5)}{|dist|}$, where $|dist|$ is the average distance among the documents after term t is removed. Its runtime complexity is also $O(n^2)$, where n is the number of documents.

The term contribution methods ranks the terms according to their contributions to the similarities between documents. It is defined by this equation $TC(t) = \sum_{i,j \cap i \neq j} f(t, d_i) \times f(t, d_j)$, where $f(t, d_i)$ is the TF-IDF weight of term t in document d_i. The runtime complexity of this feature selection process is also $O(n^2)$, where n is the number of documents.

2.4 Use of Citation

Citations have been playing an important role in literature writing, and more particularly, in scientific research and publications. As Blaise Cronin [1] put it, "Metaphorically speaking, citations are frozen footprints in the landscape of scholarly achievement; footprints which bear witness to the passage of ideas." [41]. Systematic use of citations can be traced back as early as 1873, when the Frank Shepherd Company [9] began its legal service by publishing its citators - lists of all the authorities citing a particular case, statute, or other legal authority. However, in the context of scientific literature, there had not been formal research on citations until the 1950s.

Starting with Eugene Garfield's [7] Citation Indexes for Science [51] in 1955, research on citations began to draw more and more attention and effort from scientific communities. Two other scientists who have made significant contributions to this area are Henk Moed [15] and Blaise Cronin [1]. Whereas Garfield has done breakthrough work on citation index such as the paper mentioned above, journal impact factor [53] and [52], and funding Institute for Scientific Information (ISI), both Moed and Cronin have done outstanding research on bibliometric measurement ([74] – [79], [41] – [43]).

With the foundation on citation research laid by these three giants, researchers around the globe have been able to explore other aspects and use of citations, such as using citations to build citation networks, to do document clustering, as well as more research on citation indexing, ranking journals or papers using citations. In this section, we present a comprehensive review on research topics and applications focusing on different aspects of citation and discussing future possible topics on citations.

2.4.1 Citation indexes/networks

A citation index is an ordered list of cited articles, each with a list of citing articles. The citing article is identified as a source, and the cited article as a reference ([61]). A citation index allows users to easily establish which later documents cite which earlier documents. One can use citation indexes to build a citation network. For example, starting from the newest citation index, we can build a citation network by tracing back to the oldest papers along citations. A citation index can be thought of as a two-layer or shallow citation network, while a citation network can be considered as a multi-layer citation index.

Inspired by Shepherd's Citations ([9][31]), Garfield proposed a bibliographic system for science literature in [51]. Its intention was to use a citation index to offer "a new approach to subject control of the literature." Besides the advantages of a citation index, such as evaluating the significance of a particular work, and the coding of citation entries, preparation/realization of the citation index were also discussed in this paper. With this idea, Garfield founded the Institute for Scientific Information in 1960, that maintains

citation databases covering thousands of academic journals, including a continuation of its longtime print-based indexing service the Science Citation Index (SCI), as well as the Social Sciences Citation Index (SSCI), and the Arts and Humanities Citation Index (AHCI). ISI was acquired by Thomson Scientific & Healthcare in 1992, and then became Thomson Scientific ([33]) that now provides the online academic service - Web of Science ([34]). According to their website, Web of Science covers over 10,000 of the highest impact journals worldwide, including Open Access journals and over 110,000 conference proceedings in areas of the sciences, social sciences, arts, and humanities, with coverage available back to 1900.

More citation index systems have been developed and readily available since SCI. Another popular commercial general-purpose citation index system is Scopus ([30]) that is published by Elsevier. It is available only online and similarly combines subject searching with citation browsing and tracking in the sciences, social sciences, arts, and humanities. According to their website, Scopus indexes 16,500 titles from more than 4,000 international publishers. It has 100% coverage of Medline titles and its coverage is over 99% complete as of 1996 on the issue level. It also indexes abstracts back to 1823.

Besides these two commercial citation index systems, we also want to discuss some notable free-accessible ones- CiteSeerX [4], PubMed [24], Google Scholar [14], and RePEc (Research Papers in Economics [28]) .

The CiteSeerX system provides citations and the function to search for scientific literature, primarily in the fields of computer and information science. It is the next generation of CiteSeer ([3]) with new architecture and data models to better meet the needs

of the research community. CiteSeer was developed in 1997 at the NEC Research Institute, Princeton, New Jersey, by Steve Lawrence, Lee Giles, and Kurt Bollacker. It was the first digital library and search engine to provide automated citation indexing and citation linking using the autonomous citation indexing method [71]. In the paper *CiteSeer: An Automatic Citation Indexing System* [55], Giles et al. claim that CiteSeer autonomously locates, parses, and indexes articles found on the World Wide Web. It thus has some significant advantages to traditional commercial citation indexes (TCCIs). First, it can index articles as soon as they are available on the web (as long as the hosting web servers allow crawling) so that researchers can keep up to date in their relevant fields. Secondly, it requires no manual effort during indexing. Thirdly, it can be used to make a more informed estimation of the impact of a given article by making the context of citations easily and quickly browsable as well as countable. Nevertheless, they also identified a couple of disadvantages compared to TCCIs. First, it does not cover the significant journals as TCCIs do. However, this disadvantage can be gradually overcome as more journals become available online and agreements with publishers to index their journals are reached. The second disadvantage is that CiteSeer cannot distinguish subfields as accurately as TCCIs since it retrieves this information automatically instead of manually. This could be improved by accumulating more articles and updating algorithms. We will have more detailed information on this in Subsection 2.4.6.

The MEDLINE (Medical Literature Analysis and Retrieval System) database contains more than 18 million records of citations and abstracts created by the U.S. National

Library of Medicine (NLM) from approximately 5,000 selected publications [17], covering biomedicine and health from 1950 to the present. A distinctive feature of MEDLINE is that the records are indexed with NLM's controlled vocabulary, the Medical Subject Headings (MeSH [18]) for information retrieval. The 2009 version of MeSH contains a total of 25,186 subject headings, also known as descriptors. Descriptors are arranged in both an alphabetic and a hierarchical structure. Most of these are accompanied by a short description or definition, links to related descriptors, and a list of synonyms or very similar terms (known as entry terms). Because of these synonym lists, MeSH can also be viewed as a thesaurus.

PubMed is a free search engine to access the MEDLINE database. In addition, PubMed also contains ([19])

1. In-process citations that provide a record for an article before it is indexed with MeSH and added to MEDLINE or converted to an out-of-scope status
2. Citations that precede the date that a journal was selected for MEDLINE indexing (when supplied electronically by the publisher)
3. Some OLDMEDLINE citations that have not yet been updated with current vocabulary and converted to MEDLINE status
4. Citations to articles that are out-of-scope (e.g., covering plate tectonics or astrophysics) from certain MEDLINE journals, primarily general science and general chemistry journals, for which the life sciences articles are indexed with MeSH for MEDLINE
5. Some life science journals that submit full text to PubMedCentral and may not yet

have been recommended for inclusion in MEDLINE although they have undergone a review by NLM, and some physics journals that were part of a prototype PubMed in the early to mid-1990's

6. Citations to author manuscripts of articles published by NIH-funded researchers

Google Scholar is a free web search engine that indexes the full text of scholarly literature across an array of publishing formats and disciplines. Released in beta in November 2004, the Google Scholar index includes most peer-reviewed online journals of the world's largest scholarly publishers. According to [82], it has the following advantages:

1. It provides international coverage of journals and scholarly resources.
2. There is no bias due to subjective selection of journals.
3. Besides journal papers, it also indexes preprints, technical reports, theses, dissertations, and conference proceedings. It contains links to the full text in approximately half of the results.

Disadvantages include

1. Language bias - it does not index complex script languages such as Japanese and Chinese.
2. Some results are not scholarly material such as library tours and student handbooks.
3. It does not offer a publisher list, a journal list, or any clues about the time-span or the disciplinary distribution of records [62].

RePEc - Research Papers in Economics, started in 1997, is a collaborative effort

of hundreds of volunteers in 57 countries to enhance the dissemination of research in economics. RePEc is an online open library [68] that is open for contribution (third parties can add to it), and for implementation (many user services may be created). Conventional libraries (including most digital libraries) are closed in both directions. Using its IDEAS database, RePEc provides links to 752,000 full text articles for 2009. Among them, 638,000 are freely downloadable. It uses CiteSeer algorithms in the process of identification and parsing of references.

A couple of significant differences among these four citation index systems are 1) MEDLINE is manually indexed, while indexing in the other three is done automatically. 2) CiteSeerX and Google Scholar show the number of citations of each article in the search results, along with the link to the list of citing articles. This enables users to quickly evaluate the popularity of the cited article and trace those citing articles. The other two do not have this feature.

Almost as early as citation index was proposed, citation network began drawing researchers' attentions. Actually, in [51] about citation index, Garfield mentioned its potential use in historical research, and thus implied the building of a citation network. However, the citation network had not been systematically studied until 1964 when the book *The Use of Citation Data in Writing the History of Science* [54] was published. In this book, Garfield et al. discussed their findings in whether citation data, in particular, citation network, could help identify key events in the history of science. With the history of DNA as an example to apply their models on, they concluded that, even though the citation network cannot replace human memory and evaluation in writing the history of

science, it can definitely “reveal historical dependencies which can be easily overlooked by the historian” and help to identify “key events, their chronology, their interrelationships, and their relative importance” in writing the history of science.

The citation network can also be used to find other useful characteristics of scientific researches. The concept “research front” was originally introduced in [84] and refers to the body of articles that scientists actively cite in a given field, which Price believes, distinguishes the scientific literature from nonscientific literature, and thus enabling science to accumulate much faster than nonscience. Price also observed an interesting phenomenon-“immediacy factor.” There seems to be a tendency for scientists to cite the most recently published articles; hence, papers are considered obsolete after a decade.

Almost all the online citation index systems, such as CiteSeer and Google Scholar, have a hidden network of their indexed articles that can be traced forward in terms of the time line of their publication date by following their “Cited by” or “Citation” feature links. Nevertheless, the citation network building and visualization are still research topics to be fully explored. CiteSpace [38] is one of the most popular results of such research. CiteSpace is a Java application for analyzing and visualizing citation networks. Its primary goal is to facilitate the detecting and analysis of emerging trends in a knowledge domain. It also can be used to identify the nature of a research front by first extracting terms from titles, abstracts, descriptors, and identifiers of citing articles in a dataset and finding the sharp growth rate of their frequencies. The intellectual base, defined as cited articles [83], can also be determined along with the research front. CiteSpace could potentially be used

by a wide range of users to explore the dynamics of a specialty in terms of a time-variant mapping from a research front to its intellectual base, as well as help find other interesting aspects of a research community. [40] and [39] present two applications of detailed citation analysis by the aid of CiteSpace.

2.4.2 Bibliometric Measurement

Intuitively, the number of citations is a good measure for ranking papers. The more a paper has been cited, the better it is, or at least we can say the more popular it is. The same argument can be used for a journal or a conference, as well as the performance of a research group or institute. So, not surprisingly, this research topic on citations came up almost as early as the citation index did. In *New Factors in the Evaluation of Scientific Literature through Citation Indexing* [53], Garfield pointed out that using an absolute number of citations to a journal to determine its importance is not much more sophisticated than using the quantity of articles it published. Rather, using the ratio of number of citations to the number of articles it has published could get a more meaningful measure of the importance of a journal. In revisiting this topic in [52], he ranked 100 journals with the highest impact using this measure over two-year, seven-year, and 15-year periods. As expected, top journals retain their prominent rankings over these three different periods. However, significant changes did happen to some journals. Journals in slow-moving fields moved up when measured in the long-term and all letters journals moved downward in the long term. Also, a few highly cited "Citation Classics" made some journals improve in the long term ranking.

The Journal Citation Reports [16], a by-product of the Science Citation Index (now a division of Thomson Scientific), annually publishes statistical information on the citation data of journals indexed. It shows the relationship between citing and cited journals, and helps in measuring journals' influence. However, as Cameron in [37] studied, there are serious methodological issues in the application of citation analysis to scholarly evaluations. To such a problem, a universal citation database might be a solution. A universal citation database would value all forms of publications equally and thus, allowing the impact of works to be judged without measurement bias.

Compared to journal ranking, ranking papers in a given field by citations count seems much more reasonable. It could be the most important reason why the "cited by" or "citation" feature provided by Google Scholar or CiteSeerX are so welcomed by the scientific communities. However, as Redner in [87] pointed out, the citation distribution provides a much more complete measure of popularity than the total number of citations. Redner also observed that the number of papers with x citations, $N(x)$, has a large-x power law decay $N(x) \sim x^{-\alpha}$, with $\alpha \approx 3$.

Moed and Cronin both did research on measuring academic performance of individuals or groups [42][43][74][75][76][77][78][79]. Realizing citation analysis plays an important role in such bibliometric measurement, both also acknowledged its limitation, therefore suggesting it should be used with other information such as "qualitative knowledge about the scholars" and their "subdisciplines" [78], or "to complement other information, both quantitative and qualitative" [42].

2.4.3 Citation Function Analysis

In [80] on the in-depth study of the quality of citations, Moravcsik and Muruge-san examined each reference made by a paper from the following aspects: a) conceptual or operational (the reference is a concept or theory, or is a tool or physical technique used in the referring paper), b) organic or perfunctory (the reference is truly needed in understanding the referring paper or is it mainly an acknowledgement), c) evolutionary or juxtapositional (the referring paper is built on the foundations provided by the reference or an alternative to it), and d) confirmative or negational (the reference is correct or not claimed by the referencing paper). In their study, they found that one-third of the references are redundant. There are slightly more conceptual references than operational ones, 60% of the references are evolutionary, 40% juxtapositional, two-fifths of them are perfunctory, and one-seventh of them are negational.

In [67], Kostoff categorized references into the following subjective functions. a) Bookmark - for the efficiency of presentation, awareness of related work; b) Intellectual heritage linkage - a link to intellectual heritage foundation showing historical context of unique contribution; c) Tracking research impacts - to convince research sponsors; d) Self-serving purpose. Kostoff introduced two concepts in explaining self-serving purpose - the “Citation club,” where each member cites the other members regularly, and the “Pied piper effect,” where citation clubs could exclude competitive concepts that threaten existing mainline infrastructures.

Interestingly, both papers were motivated by investigating the validity of the citation counting as a measurement of scholarly work. Both papers concluded that there

are limitations of such a measurement due to different functions of the references served in the referring papers and authors' biases. The MacRoberts in [73] concluded likewise. In addition, they also discussed two different philosophies regarding scientific papers - the traditional scientific view that is behind citation counting. This view affirms that the scientific paper is value free and that nature writes papers, not human beings. Hence, scientific papers are objective and rational. Another view is social constructivism. This view maintains that science was found to be "subjective, contingent, social, and historical". While a scientific paper presents a story, "the citations present an array, but not the only array possible."

2.4.4 Analysis of Relations Between Papers

Using a citation index, one can build citation networks (or literature networks). A citation network, in turn, can help historical research of science, or other research in a given scientific field. However, it would be more useful if we can obtain more information between a citing paper and cited papers. For example, if we can find the relations between a citing paper and cited papers, or the function of a cited paper as discussed in Subsection 2.4.3, we can label the citation network and hence, researchers would be able to get richer information from such a citation network.

Teufel et al. in [98] redefined the citation functions into four top level classes with a total of twelve different categories. Then with a supervised machine learning framework, they automatically classified a citation into one of these twelve categories using both shallow and linguistically-inspired features. Their experimental results reached 57%

on average in F-Measure.

In [81], a neologism (citances) was first introduced to mean the sentence(s) surrounding the citation within a document. Nakov et al. proposed the use of citances as a tool for semantic interpretation of a bioscience text. They believe that citances in bioscience text usually state known biological facts discovered in the cited papers. Moreover, the citances describe these facts in a more concise way in the citing papers than in the original papers. Thus, the citances could be a potentially valuable resource in mining bioscience literature. They addressed three issues for processing citances: determining text span, identifying the different topics, and normalizing or paraphrasing citances.

2.4.5 Scientific Document Clustering

Both [36] and [99] (our previous work) presented the use of citations in scientific literature clustering. The former used citation graph information to discover a set of words that are most informative in terms of identifying citation relationships, and then emphasized those words in a text-based clustering stage to improve the quality of topical clustering.

However, the later used a different aspect of citations – citation semantics in literature clustering. A two-level model was introduced. The first level is to cluster and label references of each scientific paper of a given collection to get citation semantics. The second level is to combine the vector space similarity measure and the "Citonomy" similarity measure that includes similarities between titles, keywords, citation semantics, and co-citation, to do paper level clustering. Promising results reported that at least a 5%

average improvement was achieved in the F-Measure.

Some other works also considered citations in text classifications. For example, [103] used co-citation information together with abstract, title, or abstract plus title to do text classification. CiteSeerX also utilized co-citation information in their citation index system. However, none of them considered citation graphs, or citation semantics – labeled clusters of references. Tong et al. in [99] argued that papers in the same field most likely would cite the same kinds of previous work, but not necessarily the same work. Hence, considering similarity between citation semantics in scientific paper clustering is better than simple co-citation counting.

2.4.6 Bibliographic Attribute Extraction

In section 1, we mentioned that some citation index systems such as CiteSeerX and Google Scholar do indexing automatically. This means they extract citation information without human intervention. Because of this, the accuracy of automatic citation information extraction plays an important role in those citation index systems. In fact, a tool for extracting citation information is useful in all the other applications on citations. If a scientific paper is stored in a tagged format, such as XML (eXtensible Markup Language), then citation information extraction is just a trivial issue. However, there are still a lot of scientific papers stored in the plain text format. Some of them are obtained through the OCR (optical character recognition) process. It needs focused research to come up with useful tools to extract citation information such as the author's name, paper title, and publisher, etc., from those plain text papers.

Takasu in [97] proposed a rule-based system - an extended Hidden Markov Model called DVHMM to extract bibliographic attributes from OCR-processed reference strings. Methods for both reference alignment and reference parsing were discussed, and this model can be trained with non-aligned pairs or aligned pairs. Accuracy of extracting bibliographic attributes using either kind of training data reached more than 80% except for attributes volume and number.

In [45], Day et al. presented a knowledge-based approach for citation information extraction. They adopted an ontological knowledge representation framework called INFOMAP to automatically extract the reference metadata. They reported 97.8% overall average accuracy of citation extraction for six major reference styles. However, the phase of knowledge representation in INFOMAP is basically a manual process, and the quality of such a representation directly affects the accuracy of their approach.

Both [57] and [58] focus on name disambiguation, that is, to solve name ambiguities caused by two reasons: an author may have multiple names in different citations and multiple authors may share the same name. Han et al. in [57] presented two supervised learning approaches, while Han et al. in [58] discussed an unsupervised approach. All utilize three types of citation attributes: co-author names, title of the paper, and title of the journal. One approach in [57] uses the naive Bayes probability model (a generative model), another uses the Support Vector Machine (SVM – a discriminative model). The naive Bayes model achieved higher accuracy than the SVMs did with 73.3% compared to 65.4%. The unsupervised approach displayed in [58] is K-way spectral clustering. They used it with a QR decomposition (a decomposition of a matrix into an orthogonal and an

upper triangular matrix) for cluster assignment. They showed that the spectral methods outperform K-Means for the data sets they collected. They achieved a 61.5% to 64.7% average accuracy, and observed that the more features (co-author names, paper, and publication title words) used in author classification, the better the classification accuracy.

2.5 Ontology and Ontology Clustering

An ontology is an explicit specification of a conceptualization ([56]). In other words, an ontology is defined as a formal representation of the knowledge by a set of concepts within a domain and the relationships between those concepts. Ontologies could be divided into domain ontologies and upper ontologies. A domain ontology, or domain-specific ontology, models the specific domain. It represents the particular meanings of terms as they apply to that domain. Whereas an upper ontology (or foundation ontology), is a model of the common objects that are generally applicable across a wide range of domain ontologies.

An ontology usually consists of classes (concepts), properties (attributes), relations, and instances. Ontologies are commonly encoded in ontology languages such as RDF (Resource Description Framework [29]), RDF Scheme [27], OWL (Web Ontology Language [22]), and DAML+OIL [5].

Many ontologies have been published through the last decade, notably in biomedical domains. Here are some of the most popular ontologies (or collections) – OBOs (Open Biological and Biomedical Ontologies [21]), GO, Gene Ontology [13], MeSH (Medical

Subject Headings [18]), FMA (Foundational Model of Anatomy) [10], ChEBI (Chemical Entities of Biological Interest [2]), SNOMED CT (Systematized Nomenclature of Medicine – Clinical Terms [32]), FOAF (Friend of a Friend [11]), UMLS (Unified Medical Language System [92]), and Dublin Core (an ontology for documents and publishing [6]).

As the domain knowledge grows dramatically, especially in the biomedical domain, ontologies catch more and more attention because of their obvious advantages in knowledge discovery and management. Nevertheless, they also post a new challenge for the community - the interoperability between ontologies. This is because ontologies have been developed for different purposes and covering different aspects (e.g., literature indexing and retrieval, electronic patient records, and statistical reports on mortality and billing), and in different subdomains (e.g., diseases, genomes, molecular biology, micro-organisms, diagnoses, medical devices, procedures, and drugs). Yet, attempts to represent the whole medical domain are usually limited in scope (GALEN) [86] or lack a strong organizational structure, as in the Unified Medical Language System (UMLS). The main cause for these limitations arises from the fact that different research groups rely on heterogeneous research data sources. There have been some previous efforts on how biological resources such as Gene Ontology and GenBank [12] can be mapped to the medical information. Particularly, knowledge mapping in biological and medical ontologies is essential for the future integration of diverse biomedical domains, e.g., public health and genomic research. There is an urgent need for a mechanism to build interoperability between ontologies that are semantically related, but have been developed by

different groups and for different purposes.

In order to identify meaningful relationships among related subdomains (e.g., identification of genes responsible for a disease, development of drugs for their treatment or prediction of a pathogen's susceptibility to a drug), it is essential to know what ontology sources exist and what information they contain. Furthermore, we need to comprehensively analyze relationships between these ontologies (differences and similarities between species, how mutations affect functioning of different components in different organisms [69]), including the extent of overlapping information within them. Identifying related information among heterogeneous ontology sources and classifying them according to their relevance is an important challenge.

Existing methods for integration of ontologies use structural and semantic methods; however, there is still room for improvement. Most ontologies are organized around a concept hierarchy as the backbone with additional rules, axioms, or other constraints. Linking multiple ontologies is a difficult task because it requires a comprehensive understanding of domains to be linked. These differences occur because different ontology designers may bring different world views to the task, conceptualizing the world at different levels of granularity and abstraction. Such differences are well known semantic problems. When integrating two ontologies, the existence of synonyms and homonyms causes problems in integration. Synonyms across ontologies that are lexically unrelated may be missed, and lexical matches that are merely homonyms may be erroneously designated as being related. From an application perspective, identifying related ontologies and linking or clustering them together is very important. To our knowledge, no one has

applied the clustering technique in analyzing the relations among ontologies. Thus, it is of interest to analyze how related ontologies overlap, and how to cluster them into an ontology network, which will be discussed in Chapter 6.

Increasingly, we are also seeing the emergence of distributed scientific processing. The Semantic Web provides an important platform for this activity of biomedical information exchange to take place. Nevertheless, there are significant difficulties to be resolved before seamless interoperability and interchange can occur. Existing semantic approaches for linking are promising; however, they are computationally expensive and impractical for large scale ontologies. Several existing solutions for integrating and interoperating ontologies (using reasoners like FaCT [8] and Racer [26]) rely mainly on complex and complicated processes such as reasoning and logic-based approaches. In addition, having strong semantic modeling expertise across multiple sub-domains is a real challenge. Thus, there is a need for pragmatic alternatives to characterize the relationship between multiple biomedical ontologies.

CHAPTER 3

OVERALL FRAMEWORK – CITONOMY

The Citonomy framework is a semantic framework that utilizes the semantic information presented in documents to do document clustering. In contrast to traditional document clustering algorithms with the VSM model where all terms were treated equally, it takes into account the semantic contexts of terms in document clustering and hence, improves the accuracy of clustering. The definition of Citonomy follows.

Definition 3.0.1. *Citonomy* Citonomy is the framework of document clustering considering the semantics of documents. Given a set of documents, we first map the document space D to the semantics matrix space SM: $D = \{d_1, d_2, ..., d_n\} \rightarrow SM = \{(\vec{v_1}, sm_1), (\vec{v_2}, sm_2), ..., (\vec{v_n}, sm_n)\}$, where $\vec{v_i}, i = 1, ..., n$, is a vector in the vector space model, $sm_i = (T_i, C_i, K_i)$, T_i, C_i and K_i are the title, citation semantics, and keywords of d_i. We can further map SM to $DV = \{d\vec{v}_1, d\vec{v}_2, d\vec{v}_n\}$ and then to $CV = \{c\vec{v}_1, c\vec{v}_2, ..., c\vec{v}_k\}$, where $d\vec{v}_i, i = 1, ..., n$ and $c\vec{v}_j, j = 1, ..., k$, are the document and cluster feature vectors, respectively. We do clustering on the space SM, or DV and CV .

Among the semantic information of each document that includes the title, keywords, citation semantics (reference clusters and their labels), and co-citation information, the citation semantics is the most important part. Its definition is given as follows:

Definition 3.0.2. *Citation Semantics* The citation semantics of a scientific document d_j is defined as two matrixes Min and M_{out}. M_{in} is the matrix of terms found in titles

and surrounding sentences of documents citing d_j with each row for one citing document. M_{out} is the matrix of terms found in the titles of documents cited by d_j and the surrounding sentences where they are cited, with each row storing sorted terms as the label of each cluster of references.

However, to use M_{in}, one has to search thoroughly in a reliable citation index system to get all information of documents citing d_j. Also, as observed in [87], about 47% papers are never cited. Especially, the chance of being cited for new papers (say, published within six months) is almost zero. Based on these factors, it is reasonable and pragmatic to exclude M_{in} from citation semantics when doing document clustering. Thus, in this dissertation, we only consider M_{out} as the citation semantics of a scientific document.

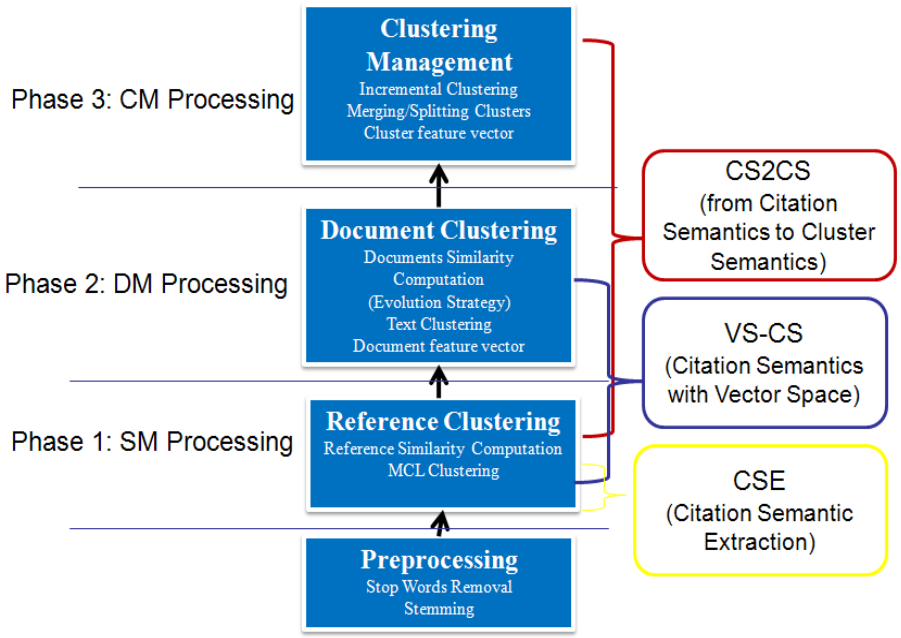

Figure 1: Citonomy – the Overall Framework

There are three major phrases in Citonomy framework. They are shown in Figure 1. Phrase 1 is the SM processing, which deals the issue of extracting the semantics of documents. Phrase 2 is the DM processing, which deals issues of document representations and document clustering. Phrase 3 is the CM processing, which deals cluster management. The issues evolved in Citonomy framework will be further explained in the following sections.

3.1 Preprocessing

This is the first step for most document/text clustering algorithms. It usually involves stop words removal and stemming. Stop words are words like “the” and “a” that do not contribute to and even are noise to document/clustering. Stemming is the process of reducing words to their stem, base, or root form. The stem does not need to be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. In other words, we consider the different forms of a word as the same in document clustering. For example, “depending” and “depends” both would be considered and hence, be stemmed into “depend”, which is reasonable. We use the Porter Stemming algorithm [88] to do word stemming.

3.2 Citation Semantics Extraction

This is the major issue involved in Phrase 1 of Citonomy. We extract citation semantics using reference clustering and labeling. Given a pair of paper titles, it is reasonable to conclude that they are semantically related if they have matching lexical tokens

or phrases. We refer to these as intrinsic matches based on explicit lexical evidence. When these paper titles are found in the context of the list of references of a journal paper, additional semantic evidence can be used to infer relatedness between them. We refer to this as extrinsic or implicit evidence. These are generally related to the specific context of each citation within the body of the manuscript. The contexts of a pair of citations can be used to derive a metric of the distance between them. In turn, the references can be clustered together to sub-classify the list of references in a scientific document. Once semantic relatedness is established, each semantic group of citations can be labeled by finding lexical similarities either between them or similarity of contextual information.

To cluster the references, we first generated similarities between every two references cited by a paper, defined by formula 3.1. Second, we used the Markov Chain algorithm (MCL) [47] to do reference clustering based on these similarities. Third, we labeled these citation clusters. The detail of each sub-step follows.

$$\mathbf{S}(r1, r2) = \mathbf{S}(t1, t2) + \mathbf{S}(s1, s2) + \mathbf{B}(r1, r2) \tag{3.1}$$

As shown in equation 3.1, the similarity $\mathbf{S}(r1, r2)$ between two references are defined by the similarities between their titles $\mathbf{S}(t1, t2)$ (defined by equation 3.2) and surrounding sentences $\mathbf{S}(s1, s2)$ (defined by equation 3.2), as well as the citation locality (or bracket) information $\mathbf{B}(r1, r2)$. The surrounding sentence of a reference is the sentence in the document body where the reference is cited. $\mathbf{B}(r1, r2)$ is the bracket or citation locality information of two references. For example, if we see "[13, 21]" in a paper, then references 13 and 21 have been explicitly considered to be the same kind of papers by the author. So when we perform clustering of references, it is important to consider this fact.

But they do not necessarily belong to the same cluster in the final clustering results. That is because we cannot fully trust the locality information. First, authors may make mistakes by putting in wrong numbers. Second, the authors' views about some references may be wrong. So we consider all the following three types of evidence when measuring the similarity of every pair of references: titles, surrounding sentences, and locality information. Titles and surrounding sentences are both considered sentences but will be compared separately, that means we will compare title to title and surrounding sentences to surrounding sentences. It makes sense to preserve individual semantics since the reference title is given by the author of the cited paper while the surrounding sentences are written by the author citing that reference. The similarity of two sentences st1 and st2 is computed as follows.

$$\mathbf{S}(st1, st2) = \frac{Count(st1 \cap st2)}{Count(st1 \cup st2)} \tag{3.2}$$

In other words, the similarity between two sentences equals the number of common terms of these two sentences divided by the total number of unique terms in the sentences. Both $\mathbf{S}(t1, t2)$ and $\mathbf{S}(s1, s2)$ in equation 3.1 use equation 3.2 to compute. The value range of $\mathbf{S}(st1, st2)$ will be between 0 and 1, inclusively. And $\mathbf{B}(r1, r2)$ in equation 3.1 will be either 0 or 1. Therefore, the value of the similarity between two references will be between 0 and 3, inclusively.

Once we finish computing the similarity of every two references of a document, we input these similarities to MCL. MCL is an unsupervised clustering algorithm for networks (also known as graphs) based on simulation of (stochastic) flow in graphs. MCL does not need to know the number of potential clusters. It just fits our situation here since

we do not know the number of clusters of the references included by each paper. However, through our experiments, we found out there are about 4 to 5 clusters of references in each paper on average.

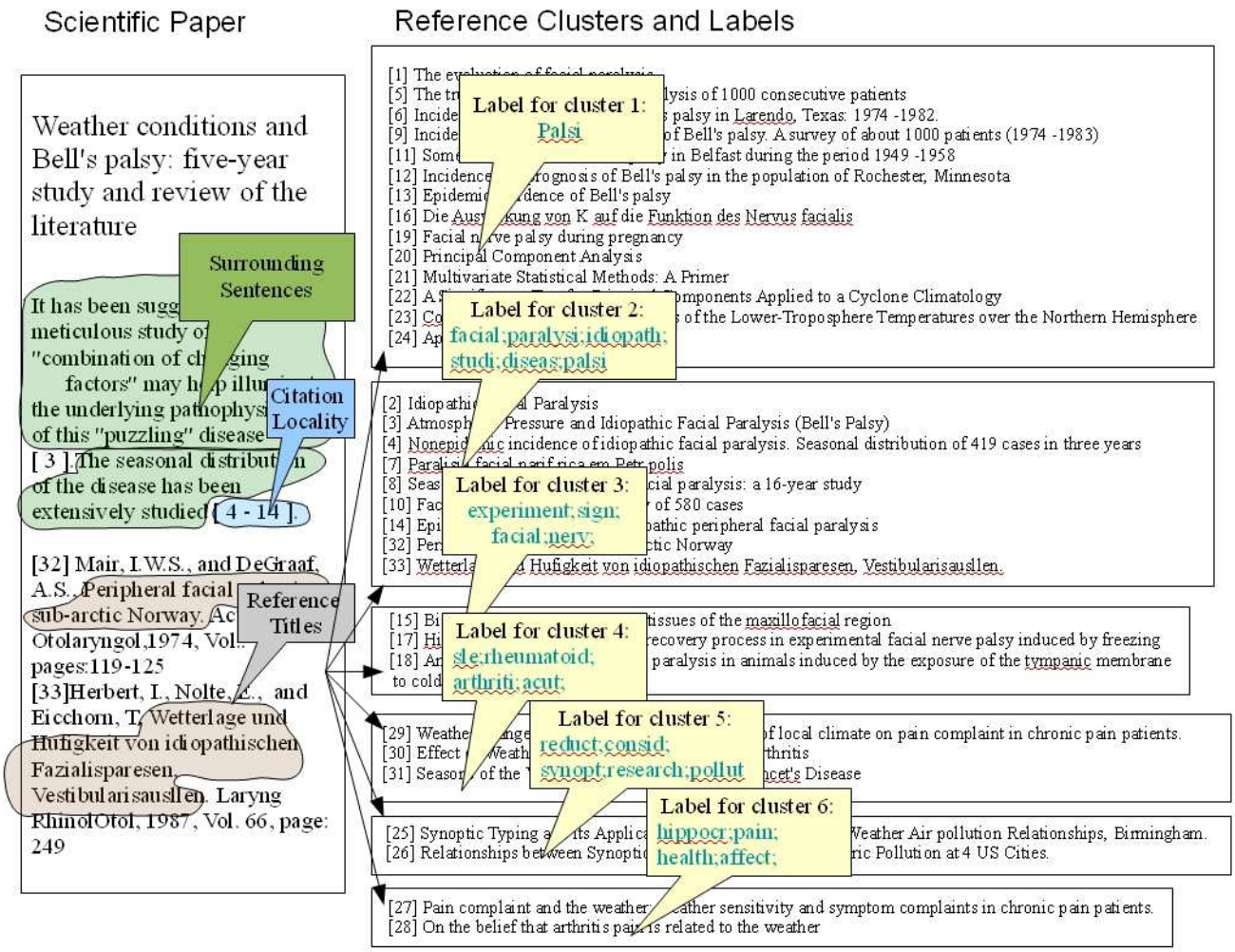

Figure 2: An Example of Reference Clustering and Labeling

We label each cluster by the most frequent terms, namely, we use those terms

that occur in half or more than half of the members (references) of a given cluster. We select terms from both the reference title and surrounding sentences. In practice, one may choose to use single words or phrases as labels. In the experiments, we first used single words as labels, later we also used multi-word terms as labels for the purpose of comparison. Since there could potentially be multiple terms that exceed the criterion (half or more), the user can choose top n terms (such as five or ten terms) as labels. Figure 2 shows an example of citation clustering and semantic annotation. In this example, six citation clusters are identified and each citation cluster is annotated with up to ten most frequent terms.

3.3 Document Clustering and Cluster Management with Citation Semantics

Document clustering and Cluster Management are issues dealt in Phrase 2 and 3 of Citonomy. They are the ultimate purpose of this framework. And the quality of document clustering will be used to evaluate the feasibility and significance of the overall framework. In other words, the accuracy of the resulted clusters will be the major concern in evaluating the Citonomy framework. Nevertheless, the runtime or complexity of the entire process will also be discussed in Chapter 4 and Chapter 5.

We proposed two approaches (CS-VS and CS2CS) to implement document clustering using citation semantics. In the first approach, CS-VS (combining Citation Semantics and Vector Space measures), when calculating similarity of two documents, we use both the similarity between vectors of two documents and the similarity between the citation semantics of these documents. That is, we calculate these two kinds of similarities

separately, then combine them together through either harmonic mean or simple addition. Then we use this measure to do K-Medoids clustering. Note, we also consider the similarity between titles and take into account the information of co-citation. CS-VS is discussed in detail in Chapter 4.

In the second approach, CS2CS (Citation Semantics to Cluster Semantics), a 3-level feature selection is introduced to utilize citation semantics in document clustering. That is, we form feature vectors for single documents and clusters by selecting features for reference clusters (level 1), single documents (level 2), and document clusters (level 3). Then we do document clustering by finding the similarities among these feature vectors.

In both approaches, we need a small amount of documents to be training data in order to find weights in similarity measure (in CS-VS), and initial feature vectors (in CS2CS). A brief comparison between CS-VS and CS2CS is shown in Table 1. The details of them will be unfolded in the following two chapters.

Table 1: Comparison between Approaches of Citonomy: CS-VS and CS2CS

	CS-VS	**CS2CS**
Highlight	Similarity between Citation Semantics	3-Level Feature Selection
Model of Documents	VSM + Citation Semantics + Title + Keywords + Co-citation	Feature Vector (formed from VSM + Citation Semantics + Title + Keywords)
Similarity measure	Combined VSM similarity and semantics similarity	Similarity between feature vectors
Document Clustering	K-Medoids clustering, static, the number of clusters is predefined	CS2CS linear clustering, dynamic, the number of clusters changes, real time clustering
Use of training set	Use evolution strategy on training set to get weights in combining similarities	Get initial cluster feature vectors from training set
Accuracy compared to traditional K-Medoids and K-Means clustering	Improved more than 5% on average	Improved more than 10% on average
Runtime complexity in terms of the number of documents n	$O(n^2)$	$O(n)$ or $O(n\log n)$ with splitting and merging

CHAPTER 4

CS-VS – COMBINING CITATION SEMANTICS AND VSM MEASURES

In this chapter, we present the first approach of using citation semantics in document clustering, that is, CS-VS, combining Citation Semantics and Vector Space similarity measure. In this approach, when we calculate the similarity of two documents, we compute the similarity between their vectors in VSM (Vector Space Model) and the similarity between their citation semantics separately, then combine these two similarities to do document clustering. The major issues dealt in this approach are how to compute the similarity between document semantics and how to combine the semantic similarity with the vector space similarity to achieve higher quality of document clustering. Figure 3 shows the framework of the CS-VS approach. It is also described as follows.

(1) Do stop words removal and stemming on the entire collection of documents including training documents.

(2) For each document in this collection, compute the similarities between every two references using equations 3.1 and 3.2 in Section 3.2.

(3) Input these similarities obtained from step (2) into MCL to get reference clusters of each document.

(4) Label each reference clusters by selecting frequent terms from the cluster members.

(5) Use evolution strategy to obtain weights in equation 4.2(or 4.3), and 4.5 (Section 4.2) from training documents.

(6) Use these weights to calculate the combined similarities of two documents considering both VSM and citation semantics.

(7) Use the combined similarities to do document clustering.

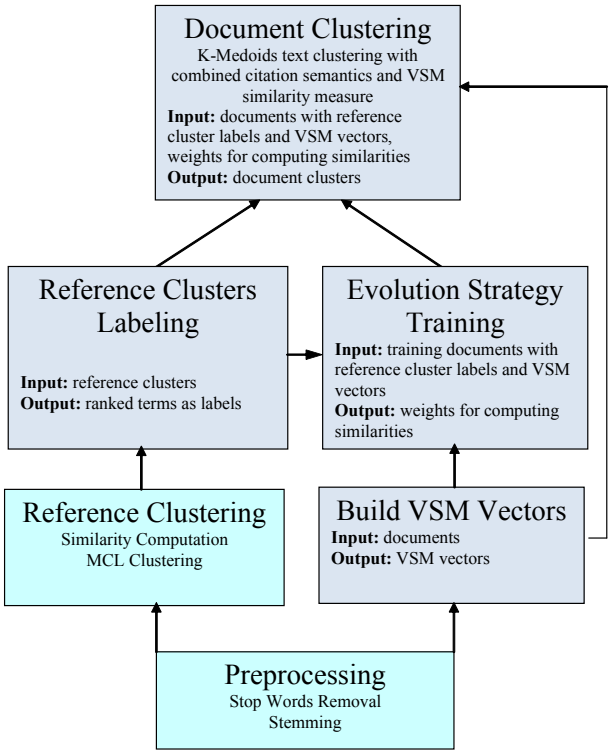

Figure 3: CS-VS – Document Clustering with Combined Citation Semantics and VSM Measure

Note that in this approach, we also considered the similarities between titles and keywords of documents as well as the information of co-citation that are reflected in equation 4.5. Preprocessing is common to all document clustering algorithms and has been described in Section 3.1 of Chapter 3. Reference clustering and labeling has also been discussed Chapter 3. All the other parts of CS-VS will be discussed in detail in the following sections and they are organized as follows. First we present the definitions

of key concepts involved in this approach. Then we describe the document clustering with combined similarity measure that is the foundation of this approach. After that, we will discuss the evolution strategy used in the training process. Lastly, we do complexity analysis of CS-VS.

4.1 Key Concepts

The significance of the CS-VS approach is the use of the citation semantic similarity. We first give its definition followed by definitions of co-citation and K-Medoids clustering.

Definition 4.1.1. *Citation semantic similarity* The Citation semantic similarity is the similarity between the citation semantics of two documents. Regarding the CS-VS approach, it is the similarity between reference clusters of the two documents involved.

The citation semantic similarity is obtained by comparing the labels of reference clusters and with the consideration of the size of each reference cluster. The details of computing citation semantic similarities are described in Section 4.2.

Definition 4.1.2. *Co-citation* The co-citation of two documents is the reference that is cited by both documents.

The number of co-citations of two documents is the number of references shared by them.

Definition 4.1.3. *semantic similarity* The semantic similarity of two documents is the linear combination of the citation semantic similarity, similarity between the tiles, similarity between keywords, and the co-citations of two documents.

The semantic similarity is computed by equation 4.5 and will be further explained in the next section.

Definition 4.1.4. *K-Medoids* K-Medoids is the process of partitioning objects into k clusters, where actual objects are picked to represent the clusters, each remaining object is clustered with the representative object (called "medoid") to which it is the most similar. The assigning process is iterated to minimize the following total absolute-error.

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j| \tag{4.1}$$

Where k is the number of clusters, p is the point in space representing an object in cluster C_j, and o_j is the medoid of cluster C_j.

K-Medoids is a variance of K-Means. More detailed information about both algorithms can be found in Chapter 2. Instead of finding the mean of all the objects in a cluster to represent it, in K-Medoid clustering, we use an actual object in the cluster to represent that cluster. Due to the citation semantic similarity being used in this CS-VS approach, we will use K-Medoids as the clustering algorithm for our document level clustering.

4.2 Document Clustering with Combined Similarity Measures

In CS-VS, we will combine the vector space similarity measure and the citation semantic similarity measure in calculating the similarities between documents. Due to the special property of citation semantics, there is no suitable way to find the "mean" of the citation semantics of documents. Therefore, instead of using K-Means, the most popular clustering algorithm, we use K-Medoids (Definition 4.1.4) to do document clustering. With K-Medoids clustering, we use a document to represent the medoid (or centroid) of a

document cluster. And hence, our major issue here is how to calculate the combined similarity between every two documents. The remaining part of this section will be dedicated to the discussion on the similarity measure in the CS-VS approach.

Similarity Measure In CS-VS, we utilize the citation semantics in document clustering by combining the similarity $\mathbf{S}_{sm}(d1, d2)$ between semantics and the similarity $\mathbf{S}_{vs}(d1, d2)$ between vectors in VSM. In the meantime, we also consider the similarities between document titles (if both have titles), keywords (if any), and the co-citation information. So the similarity between two documents could be computed by either using the harmonic mean of $\mathbf{S}_{sm}(d1, d2)$ and $\mathbf{S}_{vs}(d1, d2)$ (4.2) or the simple addition of them (4.3).

$$\mathbf{S}_h(d1, d2) = \frac{2W_1\mathbf{S}_{vs}(d1, d2)W_2\mathbf{S}_{sm}(d1, d2)}{W_1\mathbf{S}_{vs}(d1, d2) + W_2\mathbf{S}_{sm}(d1, d2)} \tag{4.2}$$

$$\mathbf{S}_s(d1, d2) = W_1\mathbf{S}_{vs}(d1, d2) + W_2\mathbf{S}_{sm}(d1, d2) \tag{4.3}$$

Where $\mathbf{S}_{vs}(d1, d2)$ is the similarity between the corresponding vectors of these two documents in VSM, and $\mathbf{S}_{sm}(d1, d2)$ is the similarity between the semantics of these two documents including citation semantics, tiles, keywords, and co-citations. They in turn can be obtained through the following formulas.

$$\mathbf{S}_{vs}(d1, d2) = \frac{\vec{v_1} \cdot \vec{v_2}}{\parallel \vec{v_1} \parallel \parallel \vec{v_2} \parallel} \tag{4.4}$$

$$\mathbf{S}_{sm}(d1, d2) = W_3\mathbf{S}_t(d1, d2) + W_4\mathbf{S}_{cise}(d1, d2) + W_5\frac{2N_{co}}{N_{r1} + N_{r2}} + W_6\mathbf{S}_k(d1, d2) \tag{4.5}$$

Where $\mathbf{S}_t(d1, d2)$ is the similarity between the titles of these two documents, which can be computed using equation 3.2, $\mathbf{S}_{cise}(d1, d2)$ is the similarity between citation semantics of these two documents, and it can be obtained through equations 4.6 through 4.11, $\frac{2N_{co}}{N_{r1}+N_{r2}}$

is used to quantify the co-citations between these two documents, N_{co} is the number of common references the two documents cite, N_{r1} and N_{r2} are the total number of references of $d1$ and $d2$, respectively, and the last part $\mathbf{S}_k(d1, d2)$ is the similarity between keywords provided by these two documents, which can also be calculated with equation 3.2.

$$\mathbf{S}_{cise}(d1, d2) = \frac{1}{2}\sum_{i=1}^{M}\mathbf{S}_{Li}(\frac{1}{N_{c1}} + \frac{1}{N_{c2}}) \tag{4.6}$$

$$\mathbf{S}_{Li} = Max(\frac{2N_{cli1}}{N_{tli1}}mR_{i1}, \frac{2N_{cli2}}{N_{tli2}}mR_{i2}, ..., \frac{2N_{cliN}}{N_{tliN}}mR_{iN}) \tag{4.7}$$

$$mR_{ij} = \frac{Min(R_{ri}, R_{rj})}{Max(R_{ri}, R_{rj})} \tag{4.8}$$

$$R_{rk} = \frac{N_{crk}}{N_r} \tag{4.9}$$

$$M = Min(N_{c1}, N_{c2}) \tag{4.10}$$

$$N = Max(N_{c1}, N_{c2}) \tag{4.11}$$

WhereN_{c1} and N_{c2} are the number of clusters of document $d1$ and $d2$ respectively, R_{rk} in equation 4.9 is the ratio of the number of references in cluster k to the number of total references of a document, R_{ri} and R_{rj} are calculated using this equation, mR_{ij} is the meta ratio of R_{ri} and R_{rj}, which is used to adjust the similarity of two reference clusters. Its maximum value will be 1. The reason for using the meta ratio instead of the simple ratio is that the sizes of two similar reference clusters might vary greatly, yet their relative sizes compared to the total number of references of the documents that they belong to may not differentiate much. $N_{clij}, j = 1, ..., N$ is the number of common terms shared by the labels of cluster i (in document $d1$) and cluster j (in document $d2$), and

$N_{tlij}, j = 1, ..., N$ is the number of total terms in labels of cluster i (in document $d1$) and cluster j (in document $d2$).

To calculate $\mathbf{S}_{cise}(d1, d2)$, we first find the document with fewer number of reference clusters, say, $d1$, that is, $M = N_{c1}$,$N = N_{c2}$, according to equations 4.10 and 4.11. Then for each reference cluster in $d1$, we compare its label (which could have multiple terms) with the label of each cluster in document $d2$, to find the most similar cluster. The maximum similarity is calculated using equation 4.7. If there is only one term allowed for each label, $\mathbf{S}_{Li}$ could only be either 0 or 1. However, we use multiple terms (such as five or ten terms) to label each cluster that provides richer semantics. After getting the maximum similarities for all the reference clusters in document $d1$, we can compute the similarity between the citation semantics of document $d1$ and $d2$ using equation 4.6.

Let us use the example as shown in Figure 4 to further explain how to calculate the semantic similarity. In this example, the total number of references of document d1 is 22, d2 24. The number of reference clusters of d1 is 4, 3 for d2. Thus, we take each cluster label in d2 to find the most similar one in d1. For example, the first cluster label (CL_{21})in d2 contains “t5”, “t7”, and “t3”. And the cluster contains 10 references. The first cluster label (CL_{11}) in d1 contains “t1”, “t2”, “t3”, and “t8”. So the similarity between these two reference clusters would be $S(CL_{21}, CL_{11}) = \frac{2}{7}\frac{\frac{6}{22}}{\frac{10}{24}} \approx 0.187$ which is shown in Figure 4. Similarly, we can calculate the similarities between CL_{21} and the other three clusters of d1. They are 0.392, 0.181, and 0.0, respectively. In other words, CL_{21} is most similar to the second reference cluster of document d1, and the similarity is 0.392. Likewise, we can find that the second reference cluster of d2 is most similar to the first reference cluster

of d1 with a similarity 0.515, and the third reference cluster of d2 is the most similar to the third one of d1 with a similarity 0.227. Therefore the citation semantics between d1 and d2 is $\frac{1}{2}(0.392 + 0.515 + 0.227)(\frac{1}{3} + \frac{1}{4}) \approx 0.331$.

The similarity between these two titles can be easily figured out as 0.375. The similarity considering co-citation is $\frac{2}{22+24} \approx 0.043$. Using equation 4.5, and supposing $W_3 = W_4 = W_5 = 1$, and $W_6 = 0$ (no keyword), we get the semantic similarity between documents d1 and d2 as $0.375 + 0.331 + 0.043 = 0.749$.

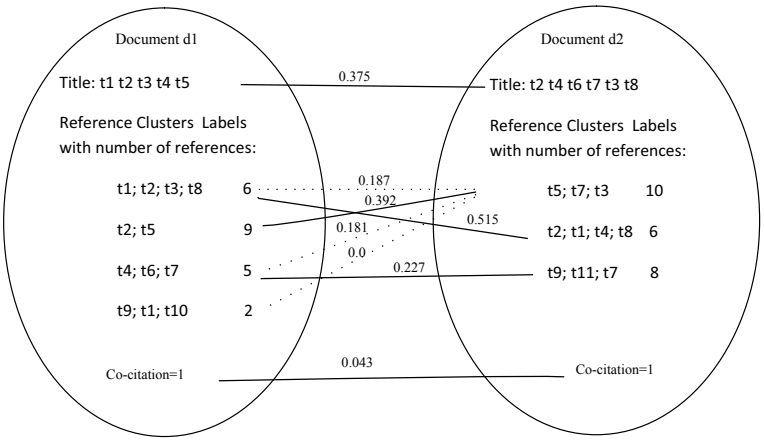

Figure 4: An Example of the semantic similarity of Two Documents

4.3 Evolutionary Strategy Training

We designed an automatic training model using evolution strategy ([85], [93]) to obtain the weights of the similarities, namely, W_1 and W_2 in equations 4.2 and 4.3, W_3, W_4, W_5, and W_6 in equation 4.5. Evolution strategies are used in technical optimization problems when no analytical objective function is available, and no conventional

optimization method existed. Thus, users have to rely only on their intuition or a trial-and-error strategy.

According to [94], evolution strategies can solve a wide range of constrained and unconstrained non-linear optimization problems and produce better results than many conventional, highly complex, non-linear optimization techniques. However, the objective function for which the evolution strategies are applied should support strong causality. In other words, small changes in the parameters must result in small changes in the function value. Experiments also suggest that the simplest version of evolution strategies that uses a single parent-single offspring search works best.

In our training model, we adopt the simple version of evolution strategies. Its procedure is shown in Figure 5. It is described as follows.

(1) Assign an initial value (1.0 in our experiments) to each of these weights. Set a threshold of the average F-Measure and the maximum number of generations.

(2) Use these weights to do document clustering on the training data and get the average F-Measure of resulted clusters of all the collections in the training data. If it is higher than or equal to the predefined threshold, stop. Otherwise, go to next step.

(3) Create a new set of values for these weights by adding a random variable a(0,1) of the standard normal distribution to each weight.

$W_i' = W_i + a(0,1)$

(4) Use these new weights to do document clustering on the training data, get the average F-Measure of the resulting clusters of all the collections in training data.

(5) Compare the F-measure associated with the offspring parameters (the new weights)

with those associated with the parent parameters (the old weights). If the F-Measure for the offspring is higher than that for the parents, replace the parents with the offspring, remembering the new F-Measure as the highest so far. Otherwise, keep the parents.

(6) Go to step 3, and repeat the process until a satisfactory F-Measure is reached, or a specified number of generations is finished.

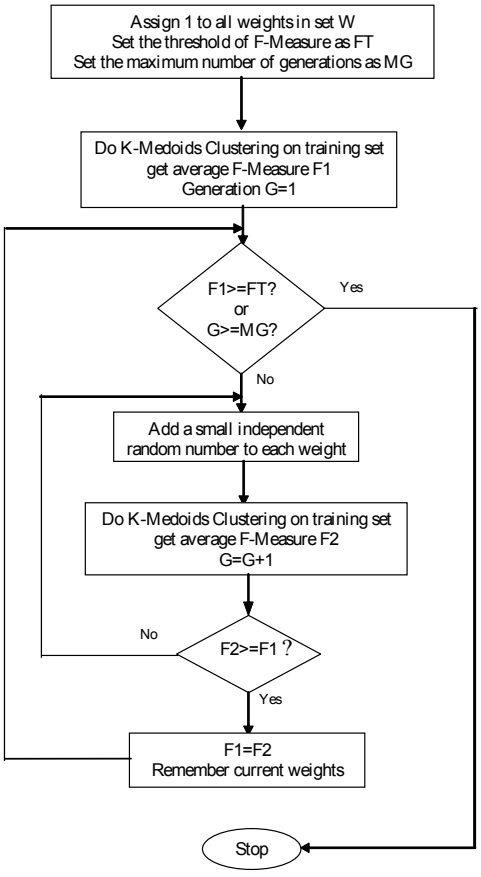

Figure 5: The Evolution Strategy Process in CS-VS

Notes: 1) At step (1), instead of assigning 1.0 to each weight, we can also use a random number out of a certain range, say 1 to 100. However, because of the property

of the evolution strategy and based on our observation, this would not change the performance of this model.

2) At step (3), since a(0,1) is generated by a standard normal distribution function, the values added to these weights are independent and thus, most likely different, which is intended by evolution strategies where each parent parameter mutates independently.

3) The user will provide the expected value of the F-Measure and the number of generations in order to let the training process stop in allowable time.

4) We can use this evolution strategy to obtain these weights altogether or separately. First, we can use these weights to do document clustering by combining the vector space measure and the semantics measure, the training process will produce the best combination of these weights. Secondly, we can also get the three weights (W_3, W_4, and W_5)($W_6 = 0$ since there is no keyword) of semantics measure first by doing document clustering using only this measure. The training process will produce the best combination of these three weights, and then we can use the training process again to obtain the other two weights (W_1 and W_2) with these three fixed. However, using these weights to test data, our experiments show that those weights obtained altogether produce better results (as presented in Chapter 7). This is because the weights obtained together reflect the complete information (citation semantics and vector space) of these documents better. Table 2 shows a demo of the changes of weights and the F-Measure in the process of evolution strategy.

Table 2: Example of the Evolution Strategy with the Threshold of F-Measure = 85%; the Threshold of Generations = 100.

Generation	W1:W2 (W3:W4:W5)	F-Measure (%)	Best W1:W2 (W3:W4:W5)	Best F-Measure (%)
1	1:1 (1:1:1)	81	1:1 (1:1:1)	81
2	1.7:2 (1:2.1:0)	80.2	1:1 (1:1:1)	81
3	2.3:3.5 (1.9:2.3:1.2)	78.7	1:1 (1:1:1)	81
4	4:2.6 (1.6:3.1:0)	81.3	4:2.6 (1.6:3.1:0)	81.3
5	5.6:4.9 (0.6:4.2:0)	81.5	5.6:4.9 (0.6:4.2:0)	81.5
6	5.1:6.3(1.1:3.4:0.8)	80.7	5.6:4.9 (0.6:4.2:0)	81.5
32	6.9:7.2 (0.5:3.3:0.6)	80.9	7.4:9.1 (0.2:3.7:0.4)	81.7
33	8:6.1 (0.5:2.6:2.5)	82.1	8:6.1 (0.5:2.6:2.5)	82.1
70	13.1:1.5 (0.9:7.1:0)	86.6	13.1:1.5 (0.9:7.1:0)	86.6

4.4 Runtime Complexity Analysis

The runtime of this approach consists of four parts: Preprocessing, reference clustering and labeling, training process, and document clustering. Since the preprocessing (stop words removal and stemming) is common to every document clustering algorithm, and it is linear regarding the number of documents, we do not include in this analysis.

As for reference clustering and labeling, since each document only goes through this process once, it is also linear in terms of the number of documents. However, the runtime is quadratic with respect to the number of references that includes the runtime of computing the similarity of every pair of references (quadratic), the runtime of MCL clustering with these similarities (Quadratic), and the runtime of labeling (linear). Since the runtime of both the training process and document clustering process depends on the algorithm used for document clustering, we discuss this algorithm in detail in the following paragraphs.

Comparison studies such as [96] have shown that the bisecting and regular K-Means algorithms perform best in text clustering regarding both accuracy and runtime.

However, K-Means requires the calculation of the "mean" of a group of objects in terms of the predefined measure. In this approach, since the semantic measure is involved, there is no ideal way to define the mean of semantics of a group of reference clusters. Therefore, we use the K-Medoids algorithm, a variance of K-Means, to do document clustering. Instead of finding the "mean" of a group of objects, K-Medoids finds an actual object that is the centroid of the group regarding the predefined measure. Because it uses the actual objects, K-Medoids performs better then K-Means on data with outliers - objects with extremely large values. These objects will distort the distribution of data by affecting the "mean" greatly in K-Means clustering. The K-Medoids algorithm follows.

*(1) Randomly choose k documents in the collection **C** as the initial medoids (centroids).*

(2) Assign each remaining document to the nearest cluster concerning the similarity between this document and the medoids. Calculate and record the sum of all the similarities (SS).

(3) For each mediod d_m

For each non-medoid document d_{nm}

Swap d_m and d_{nm}, assign other documents to the new medoids and compute the new total similarity SS_{new}

if ($SS_{new} > SS$)

$SS = SS_{new}$*;*

replace d_m with $d_(nm)$

(4) repeat (2) and (3) until no medoid changes

The complexity of this process is $O(k(n-k)^2t)$, where k is the number of clusters,

n is the number of documents, and t is the number of iterations. Since k and t are usually much smaller than n, the complexity of the K-Medoids clustering algorithm is essentially quadratic. It is the toll of being insensitive to the noise.

In the training process, since the number of iterations g of evolution strategy could be explicitly preset, or controlled by setting the threshold of the objective function, in our case, the F-Measure, g is usually much smaller than the number of documents. So the runtime of training depends on the algorithm of document clustering. That means, it is quadratic in terms the number of document in training set.

Considering all the steps together, the complexity of this approach is $O(n^2)$, where n is the number of documents to be clustered.

CHAPTER 5

CS2CS – FROM CITATION SEMANTICS TO CLUSTER SEMANTICS

In this chapter, we present another approach of Citonomy, CS2CS - Citation Semantics to Cluster Semantics (Definition 5.1.6), to utilize citation semantics in document clustering. CS2CS is based on a 3-Level feature selection - the feature selection from reference clusters (level 1, Definition 5.1.7), the feature selection from single documents (level 2, Definition 5.1.8), and the feature selection from document clusters (level 3, Definition 5.1.9). Through this 3-level feature selection, we form document feature vectors (Definition 5.1.4) and cluster feature vectors (Definition 5.1.5). In the previous chapter, we discussed the approach CS-VS. The experimental tests (presented in Chapter 7) on CS-VS show that it significantly and consistently improved the quality of document clustering. However, CS-VS does not solve the runtime problem since it uses the K-Medoids clustering algorithm whose complexity is quadratic in respect to the number of documents. However, with these feature vectors, CS2CS can do linear document clustering and hence, it does not have the runtime issue as CS-VS does. Figure 6 shows the framework of CS2CS. Its brief description follows.

(1) Do stop words removal and stemming on the entire collection of documents including training documents.

(2) For each document in this collection, compute the similarities between every two references using equations 3.1 and 3.2.

(3) Input these similarities obtained from step (2) into MCL to get reference clusters of each document.

(4) Label each reference clusters by selecting frequent terms from the cluster members. This is the level 1 feature selection.

(5) Using evolution strategy to obtain weights in equation 4.2(or 4.3), and 4.5 from training documents.

(6) For each single document in existing clusters, using the weights obtained from step (5) to form the feature vector of each single document. This is the level 2 feature selection.

(7) For each existing cluster, form the feature vector of the cluster using the feature vectors of all the documents inside that cluster. This is the level 3 feature selection.

(8) Linear Document clustering

(9) Check for Document Clusters Splitting and Merging

(10) For each new document, repeat steps (8) and (9). Note that the first five steps are the same as those in CS-VS. The other steps are specific to CS2CS. By using feature vectors, not only can CS2CS cluster documents in linear time, but it also improves the quality of clusters significantly over traditional document clustering algorithms. Furthermore, with CS2CS, we can obtain the label (semantics) of each cluster. Lastly, with a little sacrifice of runtime (from $O(n)$ to $O(nlongn)$), CS2CS can dynamically decide the number of clusters according to the contents of clusters.

The rest of this chapter is organized as follows. We first give the definitions of the key concepts related to CS2CS. Then we discuss the details of level 2 and level 3 of the 3-level feature selection. (Level 1 is the same as the labeling of reference clusters that

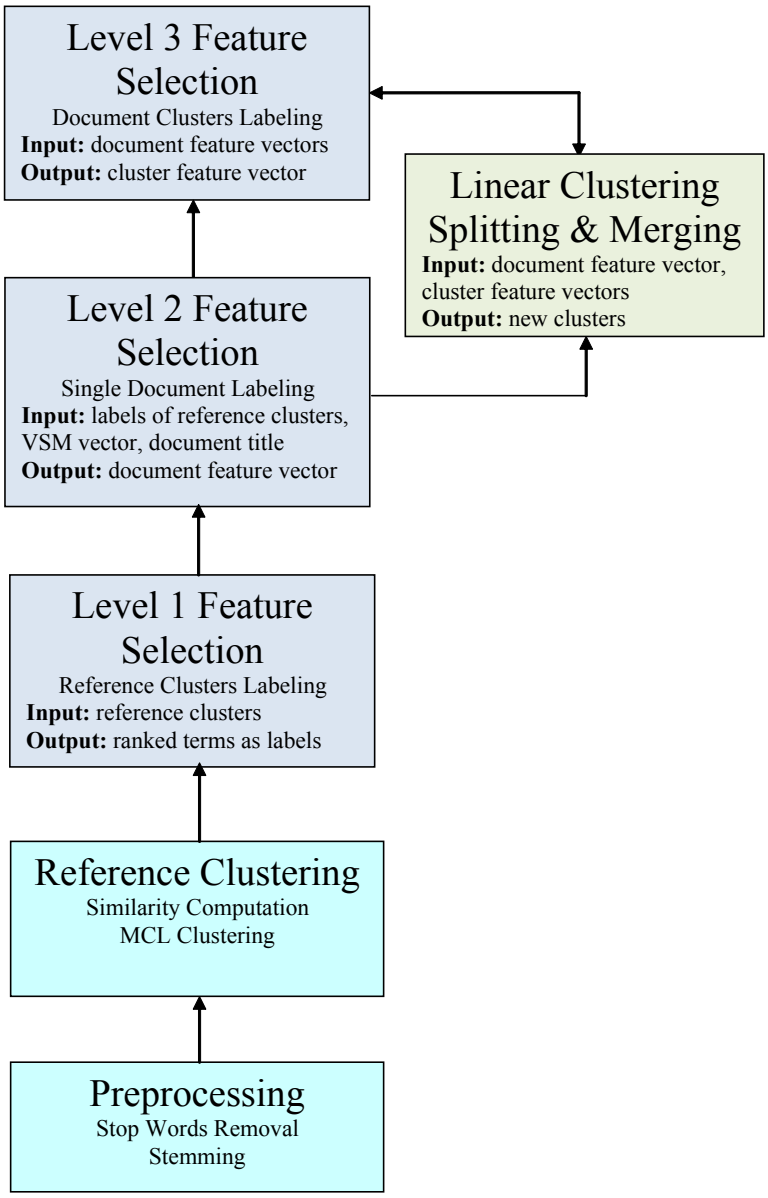

Figure 6: CS2CS – Document Clustering with 3-Level Feature Selection

was discussed in Section 3.2 of Chapter 3.) Following that, we present the algorithm of linear document clustering. Then we discuss the cluster splitting and merging. That is followed by discussions on selection of lengths of feature vectors, use of ontology, and

fuzzy clustering. Lastly, we wrap up this chapter with the complexity analysis of CS2CS.

5.1 Key Concepts

CS2CS uses feature vectors to do document clustering. We define two kinds of feature vectors – Document Feature Vector and Cluster Feature Vector. The following are the definitions of these feature vectors and cluster semantics.

Definition 5.1.1. *Feature* In the context of document clustering, a feature of a document is a term (or token) that occurs in the document.

A term could consist of multiple words or a single word. Depending on different requirements of situations, one can choose to use only single-word terms, or include multi-word terms. Generally speaking, compared to single-word terms, using multi-word terms ends up with more accurate results, but takes more runtime. This is because concepts could be multi-word and single-word. Including multi-word terms allows more real concepts to take part in the process of clustering, and hence more precise results. On the other hand, including multi-word terms will increase the lengths (or dimensions) of the feature vectors (Definition 5.1.2) that leads to a longer runtime.

Definition 5.1.2. *Feature Vector* A feature vector $\vec{v}$ is a list of terms Γ together with their weights.

Definition 5.1.3. *Length of Feature Vector* the length (or size) of a feature vector $\vec{v}$ is the size of Γ that is the set of terms the feature vector has.

Figure 7 shows an example of a feature vector. Its length is 5.

Definition 5.1.4. *Document Feature Vector* The feature vector $\vec{dv}$ of a document d, called

Feature	Weight
Term1	0.5
Term2	0.4
Term3	0.3
Term4	0.2
Term5	0.1

Figure 7: An Example of a Feature Vector

the Document Feature Vector, is a list of terms Γ together with their weights, and $\Gamma \subseteq \Phi$, where Φ is the set of terms in document d.

A document feature vector is a feature vector formed by the terms in a document. The weight assigned to each term takes into account the locality of that term. Figure 8 shows an example of a document feature vector and its formation. In this example, we use $W_1 : W_2 : W_3 : W_4 = 1 : 1 : 1 : 1$ as the weights shown in formula 5.1. The average weight in the vector of VSM is 0.25 in this example. That is $W_{avg} = 0.25$ in formula 5.1. Taking Term1 for example, since it occurs once in the title, twice in the reference cluster labels, and has a weight 0.5 in the vector of VSM, its weight in the document feature vector is 1+2+0.5/0.25=5. In the same way, the reader can figure out the other terms' weights in the document feature vector. Section 5.2 covers the detailed description of the process of forming document feature vectors.

Definition 5.1.5. *Cluster Feature Vector* The feature vector $\vec{cv}$ of a cluster C, called the Cluster Feature Vector, is a list of terms Ψ together with their weights, and $\Psi = \bigcup_{i=1}^{m} \Gamma_i$, where m is the number of documents in cluster C, Γ_i is the set of terms of the document feature fector of document d_i .

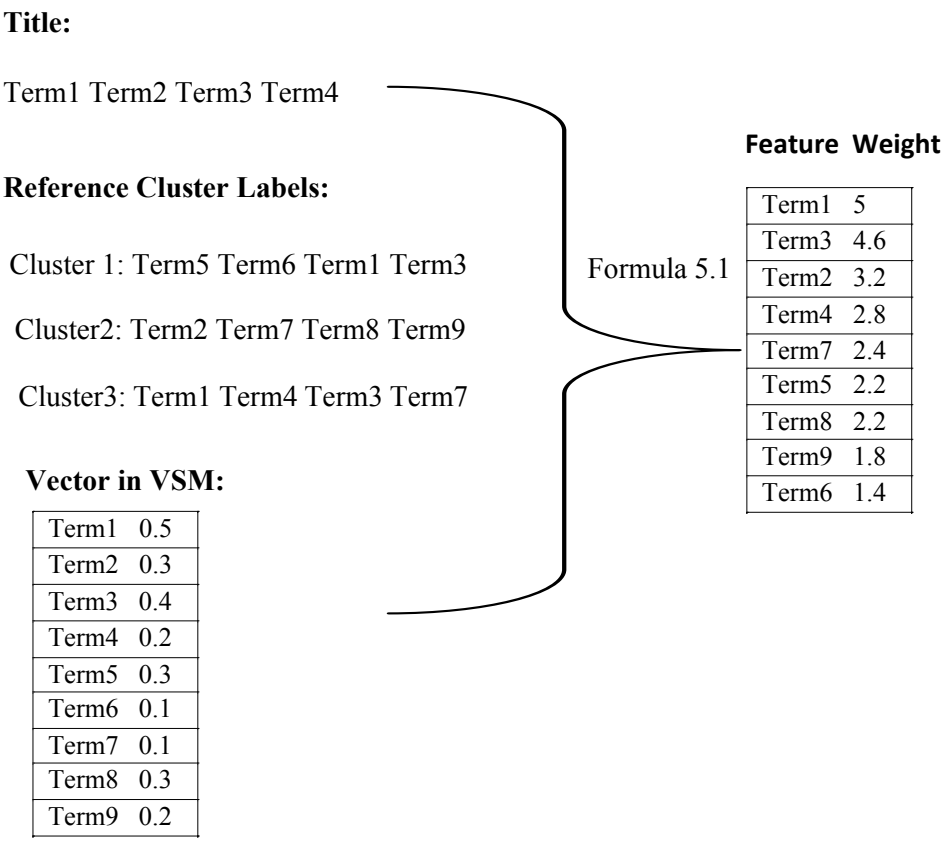

Figure 8: An Example of a Document Feature Vector and Its Formation

A cluster feature vector is formed by the document feature vectors. Figure 9 shows a demonstrative example of a cluster feature vector and its formation. The terms' weights in the cluster feature vector are determined by counting the occurrences of terms in the document feature vectors. For example, Term1 occurs in all three document feature vectors, so its weight in the cluster feature vector is 3. Note that this cluster feature vector is before normalization of its weights. Section 5.3 covers the details on the construction of cluster feature vectors and the process of their normalization.

Definition 5.1.6. *Cluster Semantics* The cluster semantics of a cluster C is the ranked list

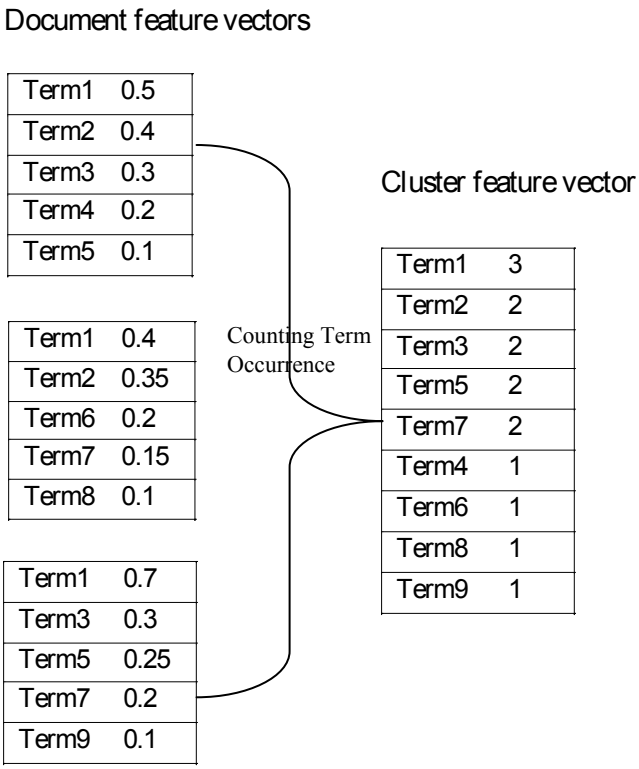

Figure 9: An Example of a Cluster Feature Vector and Its Formation

of terms Ξ, and $\Xi \subseteq \Psi$, where Ψ is the set of terms of the cluster feature vector of cluster C.

The cluster semantics is the ranked terms of the cluster feature vector or a subset of it. Since they are used for visually labeling a cluster, we do not need to include the weights of the terms.

Definition 5.1.7. *Level 1 Feature Selection* It is the process of selecting terms Γ_r from each reference cluster C_r to be the label of each cluster, and $\Gamma_r \subseteq \Phi_r$, where Φ_r is the set of terms covered by C_r.

Definition 5.1.8. *Level 2 Feature Selection* It is the process of selecting terms Γ from a

document d to form the feature vector of d, and $\Gamma \subseteq \Phi$, where Φ is all the terms in d.

Definition 5.1.9. *Level 3 Feature Selection* It is the process of selecting terms Psi from the document feature vectors inside cluster C to form the feature vector of C, and $\Psi = \bigcup_{i=1}^{m} \Gamma_i$, where m is the number of documents in cluster C, and Γ_i is the set of terms of the document feature vector of document d_i .

Definition 5.1.10. *TF-ICF* TF-ICF is the weight used in cluster feature vectors that is calculated by the two equations 5.2 and 5.3.

TF is used to eliminate the bias towards big clusters, and ICF is used to reduce the effect of common terms across clusters, more precisely, the cluster feature vectors.

5.2 Feature Selection for Single Documents

In this step, we select significant terms to form the feature vector of each document. First, we need to sort all the terms of a given document d_j by considering both its vector representation in VSM and the semantic information including title (with weight W_3), keywords (with weight W_6), and citation semantics (with weight W_4). In this approach, we do not take co-citation into account. It is because that co-citation is in the context of two documents, but here we are forming the feature vector for a single document before comparing it to any other document. Using the weights obtained from step 5 we can find the weights of all the terms of each document and hence sort them according to their weight. Then we can select the top x terms together with their weights to form the feature vector of that document.

To calculate the new weight of each term, we consider its TF-IDF value in the

vector of VSM, its occurrences in title, keywords, and labels of reference clusters, together with W_1, W_2, W_3, W_4, and W_6 that are obtained from step 5, the training process (We do not use W_5 that is the weight of co-citation. The reason is in the previous paragraph.) For example, if we have weights $W_1 = 5$, $W_2 = 1$, $W_3 = 1$,$W_4 = 10$, $W_6 = 0$ (there is no keyword provided in the data set we used). Suppose we have the word "Web" that occurs in the title once, in the reference cluster labels twice, and its TF-IDF value is 0.03. Then its total weight would be $(1 * W_3 + 2 * W_4) * W_2 + 0.03 * W_1 = 21.15$. Since these weights were intentionally used for combining vector space and Citonomy similarity measure (Equations 4.2 and 4.5), which is the measure used in CS-VS, they only provide a rough estimation of the weights that we use in computing new weights of the terms in a single document. In other words, we need to do some adjustments. In particular, the TF-IDF values are usually small with a large number of documents. For example, in the data set we used with about 700 documents, the average of the TF-IDF values is 0.0019. So the actual formula we used to calculate the total weight of each word (term) is as follows:

$$W_{ttl} = (O_1 * W_3 + O_2 * W_4) * W_2 + W_{TF-IDF} * W_1 / W_{avg} \tag{5.1}$$

Where O_1 and O_2 are occurrences of a term in the title and labels of reference clusters, respectively, W_{TF-IDF} is the TF-IDF value of that term, and W_{avg} is the average of all terms' TF-IDF values of the data set.

Besides the computation of weights of terms in forming feature vectors of single documents, there is another issue worth discussion. That is, the choice of the length of the feature vector of a single document. In other words, how many top terms shall we use to form the feature vector to best represent a document, to have the best result document

clustering? We will discuss this in Section 5.6.

5.3 Feature Selection for Document Clusters

Once we find the feature vectors of all the single documents of a cluster, we can use them to form the feature vector of the cluster. That is, we use all the terms from these feature vectors of all the single documents to form the feature vector of the cluster they belong to. The weight of each term in the cluster feature vector is its occurrence in all the document feature vectors in the cluster. Note here we ignore the terms' weights in document feature vectors. If these weights used, it would be as same as finding the mean of these document feature vectors. The reason of ignoring them is that, they are used to rank the terms within a document. While these weights are useful in comparing the significance between terms within a single document, they are not comparable across documents and therefore, the cluster feature vector would misrepresent the cluster if they were used. For example, suppose cluster C has 100 documents, and TermX only occurs in one of the document feature vectors with a weight 20; TermY occurs in all 100 document feature vectors each with a weight 0.15. If we use their total weights in the cluster feature vector, TermX would have more weight than TermY. However, even TermX is a very significant term in the document to which it belongs, it is not as significant as TermY in respect to this cluster. In other words, it is not as useful as TermY in differentiating cluster C from other clusters. Therefore, occurrence counting is more reasonable then the weights' sum when forming cluster feature vectors from document feature vectors. However, to best represent each cluster, we need to consider all the cluster feature vectors

together and normalize them. We use a 2-dimensional (within each cluster and across clusters) normalization procedure to do this.

First, to avoid bias towards the big cluster, we need to change the weight of each term from a simple occurrence to a term frequency (count/cluster size). We also need to normalize the weight of each term across all the clusters to reduce the effects of common (terms) words across the clusters. We use ICF (inverse cluster frequency) that is shown in equation 5.3, to achieve this goal. So altogether, we use TF-ICF instead of TF-IDF, to normalize the weights of terms in the feature vectors of clusters, while TF-IDF has been used in finding the vector representation of each single document in the VSM model.

Lastly, we want to normalize each feature vector to a unit vector using the Euclidean norm (that is, its length is 1 regarding Euclidean norm), to make similarities between feature vectors easy to compute.

Altogether, the weight of each term in the cluster feature vectors will be calculated using the following three formulas 5.2 (within a cluster), 5.3 (across clusters), and 5.4 (within a cluster), where W_{ij} is the final weight of term j in the feature vector of cluster i, W_{occ} is the number of occurrences of term j in the feature vectors of all the single documents within cluster i, S_i is the total number of documents in cluster i, k is the number of clusters, and x is the length of the feature vector of cluster i. Figure 10 shows an example of three cluster feature vectors before and after TF-ICF normalization.

$$W_{ij1} = \frac{W_{occ}}{S_i} \tag{5.2}$$

$$W_{ij2} = \frac{W_{ij1}}{\sum_{m=1}^{k} W_{mj1}} \tag{5.3}$$

$$W_{ij} = \frac{W_{ij2}}{\sqrt{\sum_{l=1}^{x} W_{il2}^2}} \tag{5.4}$$

We do not use a logarithm to calculate ICF as commonly used in calculating IDF. Even

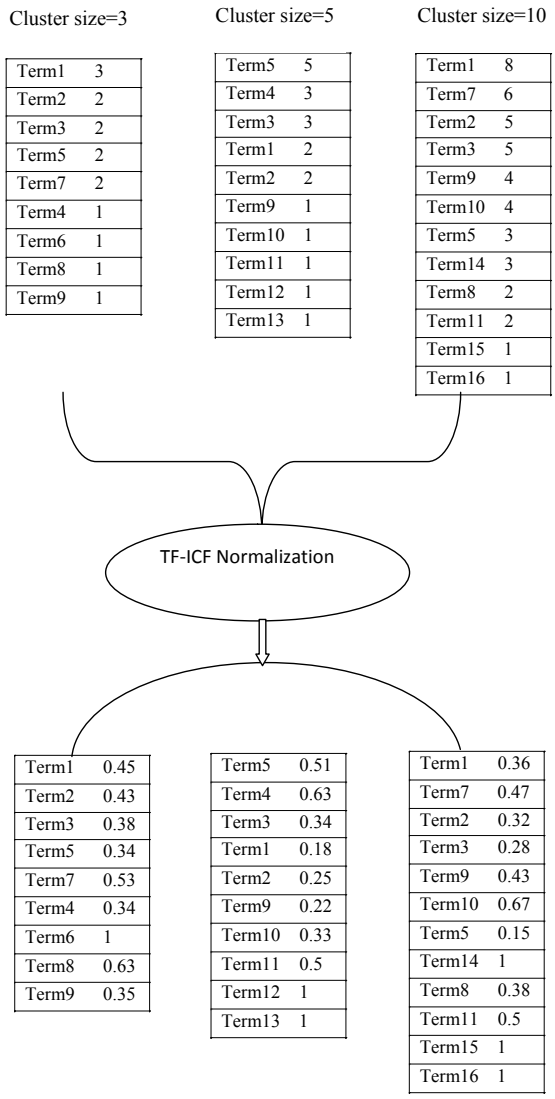

Figure 10: An Example of TF-ICF Normalization of Cluster Feature Vectors

though a term word occurs in all the feature vectors of these clusters, we do not ignore it completely as IDF does ($log1 = 0$). The argument for using IDF in building vectors

of documents is that if a term occurs in every document, then it will not contribute in clustering these documents. But in our situation, even though a term occurs in all the feature vectors of clusters, it may have different weights in these feature vectors, it will still be useful when calculating the similarity between every two feature vectors of these clusters. If this term is also in the feature vector of the new document, it will contribute to the similarity between the feature vector of a new document and the feature vector of one of these clusters. Therefore, it helps the document clustering and updating. Otherwise, if we remove this term from all the feature vectors, we will lose some information and hence, cause poor clustering results. This is really the most important step in finding best features of a cluster. In our experimental results (Subsection 7.3.7), we can see this normalization has great advantage over IDF like normalization.

The feature vector of each cluster is similar to the vector of the center of each cluster, but not the same thing, since we get this feature vector not by calculating the mean of all the vectors in the cluster, but rather by extracting significant words (terms) from every document in the cluster. To understand the feature vector of a cluster, one can imagine there is a container holding all the documents of that cluster, and the feature vector of that cluster is the label written on that container indicating what kind of material it stores.

5.4 Linear Document Clustering

The algorithm of this part is as follows.

(1) *For the new document*

(2) *do level 2 feature selection to get the document feature vector*

(3) *For each cluster*

(4) *Compute the similarity between the document feature vector of the new*

(5) *document and the cluster feature vector*

(6) *Assign this document to the cluster to which it is most similar regarding their*

(7) *feature vectors*

(7) *Update the feature vector(s) of the cluster(s) to which the new document was*

(8) *just added with level 3 feature selection.*

For each new document, we use the procedure described in Section 5.2 to obtain its feature vector. Then we normalize it to a unit vector using the Euclidean norm as shown in equation 5.4. Comparing the similarities between this feature vector and those of the k clusters, we can decide which cluster the document belongs to. In the case of fuzzy clustering, a degree of belonging could also be obtained at the same time when computing similarities. Also, if the similarity between the new one and each existing one is too low, say, lower than a predefined threshold, or lower than the minimum similarity between all existing feature vectors, it may form a new cluster by itself. We use Cosine coefficient as the similarity between the two vectors $\vec{v_i}$ and $\vec{v_j}$ that are computed according to the formula 4.4. However, since the involved vectors are all unit vectors, the bottom part of the fraction will always be 1 and hence, could be ignored. That is, we can use the following simplified formula to calculate the similarity between these two feature vectors,

where $\vec{v_i} \cdot \vec{v_j}$ is the inner product of $\vec{v_i}$ and $\vec{v_j}$.

$$Similarity(\vec{v_i}, \vec{v_j}) = \vec{v_i} \cdot \vec{v_j} \tag{5.5}$$

Once the new document is added to one cluster (or more than one in the case of fuzzy clustering), we need to update the feature vector of the cluster(s) to which the new document was just added. This could be done after inserting each new document, or a certain number of documents, depending on different applications or situations. Our experiments show there is no considerable difference regarding the overall runtime. By looking at the terms in the feature vector(s) of newly added document(s), we can easily update the feature vector of the cluster(s). For each cluster, we keep track of both the normalized cluster feature vector and the one before being normalized (we call it a raw cluster feature vector). For those terms that exist in the raw feature vector, we increase each of their weights by 1; for those terms not found in the raw feature vector, we add them to the raw feature vector with weight 1. Then we use formulas 5.2, 5.3, and 5.4 to normalize all the raw cluster feature vectors into unit feature vectors.

Figures 11 and 12 show a demonstrative example of the cluster feature vectors before and after adding a new document and the similarities between them. Since they are all unit feature vectors, the similarities between them are calculated with formula 5.5. Note the new document was added to the cluster represented by the cluster feature vector at the top.

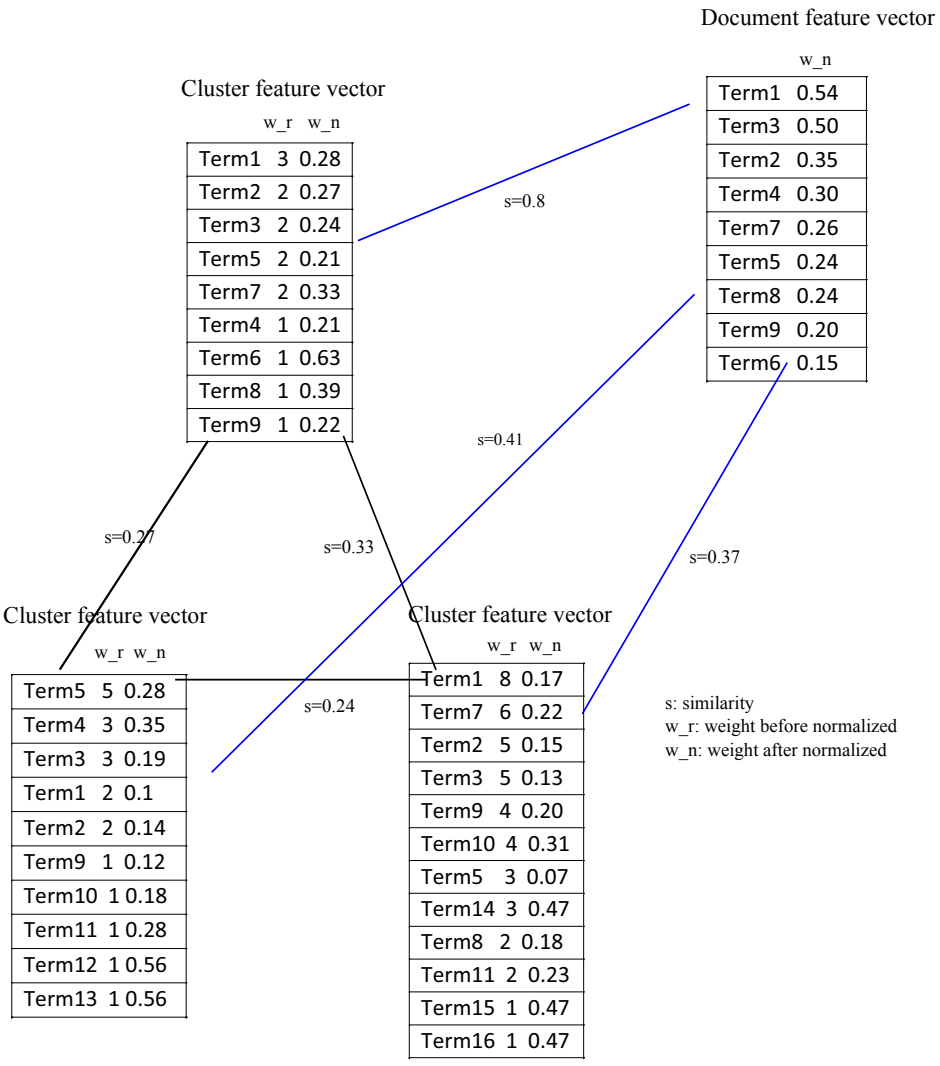

Figure 11: An Example of CS2CS Clustering – Before Adding a New Document

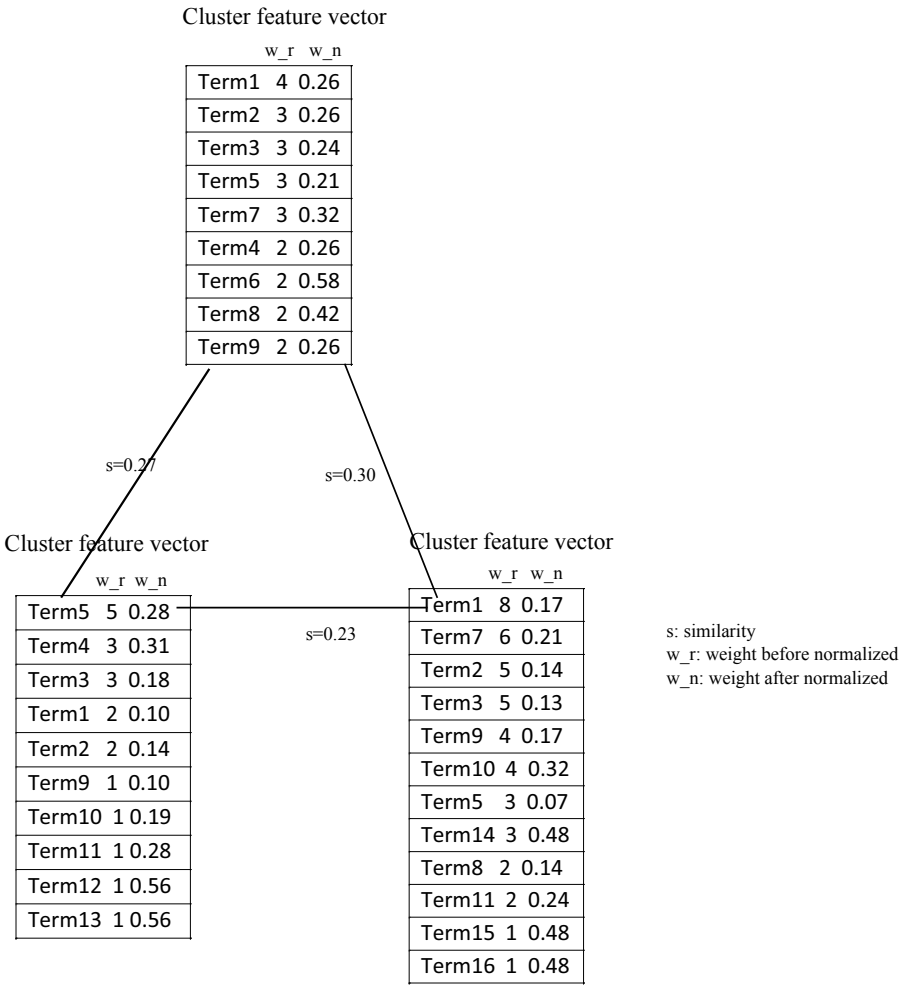

Figure 12: An Example of CS2CS Clustering – After Adding a New Document

5.5 Document Clusters Splitting and Merging

The algorithm of this part is as follows.

(1) Compute the similarity S_{ic} between the current cluster feature vector and the

(2) initial feature vector

(3) if equation 5.6 satisfied

(4) Split the cluster into two by comparing the feature vector of each

(5) document to the current and initial feature vectors of the cluster

(6) for each of the other unchanged clusters

(7) compute the similarity S_{cc} between it and the newly formed cluster(s)

(8) using their feature vectors

(9) if equation 5.9 or 5.10 satisfied

(10) Merge these two clusters and form the new feature vector using

(11) the level 3 feature selection

After updating the cluster feature vector(s)of the cluster(s) where the new document(s) have been added, we will compare the current feature vector(s) with the initial feature vectors, as well as the feature vectors of other clusters. Through these comparisons, we decide whether to split or merge clusters. The user can choose when to check for splitting and merging. In default, we do this check whenever the number of documents doubles.

Splitting If the feature vector ($\vec{v_2}$) of the newly updated cluster is so different from its original one ($\vec{v_1}$), that is, the similarity between their feature vectors is close to 0, or less than a predefined threshold, it will be the candidate to be split. But we also take into account the sizes (numbers of terms) of these two feature vectors and the sizes of the current and original clusters. We will split a cluster if the the inequality formula 5.6 holds, where cs_1 and cs_2 are the size of the original and current clusters, respectively. In other words, if the similarity between the current and original feature vector becomes too small, or the size of the feature vector increases a lot, we may split the cluster into two.

However, both could be the result that too many new documents were just added to this cluster, which could be normal change. In that case, we may not split the cluster. Figure 13 shows a demonstrative example of splitting. Inside the clusters, "dx*", "dy*", and "dz*" mean the documents of category "x", "y", and "z", respectively.

$$Similarity(\vec{v_1}, \vec{v_2}) \cdot \frac{\frac{cs_2}{cs_1}}{\frac{Size(\vec{v_2})}{Size(\vec{v_1})}} < split - threshold \tag{5.6}$$

We use the initial feature vector as the feature vector ($\vec{v_1}$) of one of the newly formed clusters by splitting, the current feature vector $\vec{v_2}$ as the other one. Then we assign each member document inside the big cluster to either cluster depending on the similarities between its feature vector and these two cluster feature vectors. Moreover, we may also want to look at the documents in other clusters to see if they belong to these two new clusters. In other words, for each document d in any other cluster c, we compute the similarity between the feature vector of d and the feature vector of cluster c, the similarity between the feature vector of d and $\vec{v_1}$, and the similarity between the feature vector of d and $\vec{v_2}$, to see whether d should stay in c or go to one of the newly formed clusters.

If we do not change the other clusters, that is, if we only split a cluster when inequality 5.6 is satisfied without looking at other clusters to further update newly formed clusters, then we have the following theorem regarding the quality of the clusters after splitting.

Theorem 5.5.1. *If the splitting of cluster c separates the documents of two categories A and B (A is the category represented by the label of cluster c before splitting) into clusters c_A and c_B (correctly labeled), and the number of documents of A in c was less than or equal to that of B, the average precision of c_A and c_B is higher than that of cluster c.*

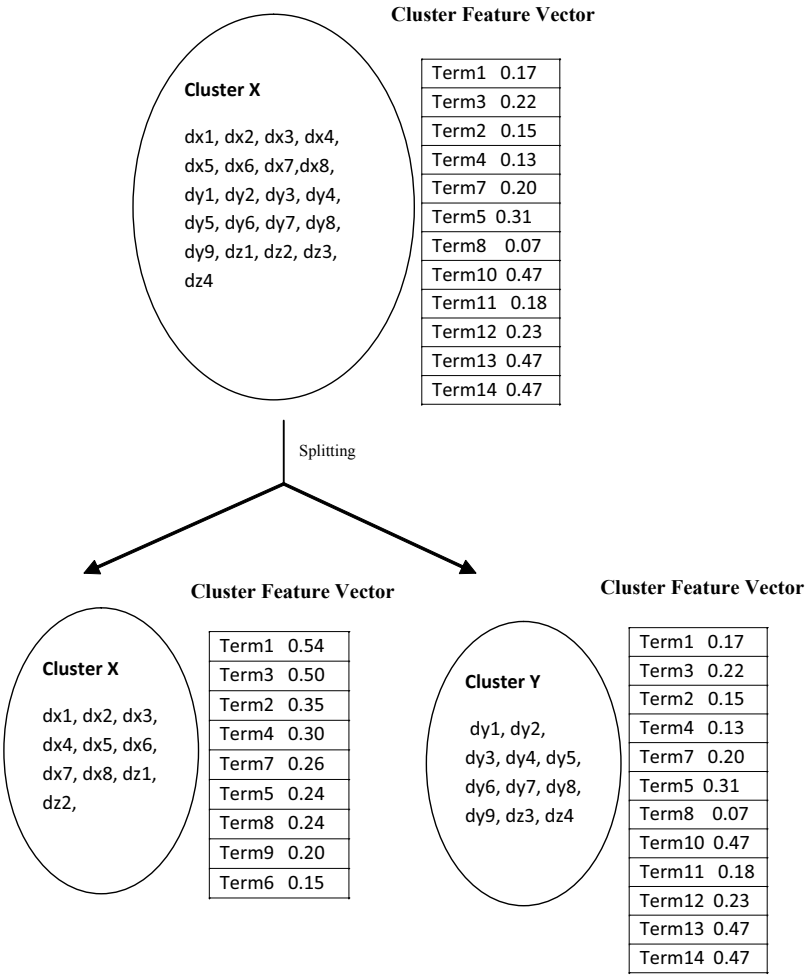

Figure 13: An Example of Cluster Splitting

Proof Suppose the number of documents of A, B, and other categories in cluster c are n, m, and l, respectively. From the assumption of this theorem, we know $n < m$. Also, suppose the numbers of documents of other categories in c_A and c_B are l_1 and l_2 after splitting (so $l = l_1 + l_2$), the precisions of c, c_A, and c_B will be $\frac{n}{n+m+l}$, $\frac{n}{n+l_1}$, and $\frac{m}{m+l_2}$, respectively. Our task is to show the following inequality.

$$\frac{n}{n+m+l} < (\frac{n}{n+l_1} + \frac{m}{m+l_2})/2 \tag{5.7}$$

After multiplying both sides by $(n+m+l)(n+l_1)(m+l_2)$ and some cancelations, we get the following inequality.

$$n^2l_2 + nl_1l_2 < 2nm^2 + m^2l_1 + nl_2^2 + ml_1^2 + nml_1 + 3nml_2 + ml_1l_2 \qquad (5.8)$$

If $l_1 < l_2$, then $LHS \leq m^2l_2 + ml_1l_2 < RHS$. If $l_2 \leq l_1$, then $LHS \leq m^2l_1 + ml_1^2 < RHS$. In other words, as long as $n \leq m$, inequality 5.8 always holds and hence, we complete the proof.□ Note that neither the relation between l and n nor the relation between l and m affects our conclusion.

Based on this theorem, we can easily conclude that the average precision of all the clusters will also increase after splitting if we do not change the other clusters. From our experiments we notice that, even though a splitting does not separate the documents of two categories neatly, in other words, they may still mix a little in resulting clusters, the precision regardlessly increases due to the significant decrement of the denominator in one of the precisions.

Merging If two clusters (c_i and c_j) are getting closer, we will merge them. For the newly updated cluster, we get the similarities between its current feature vector ($\vec{v_{i2}}$) and the feature vector ($\vec{v_{j2}}$) of any of the other clusters. We also get the similarities between its initial feature vector $\vec{v_{i1}}$ and the initial feature vector $\vec{v_{j1}}$ of any of the other clusters. Even if the ratio is less than 1 (decreasing), but it is slower than the ratio of the total size increasing, we may also consider merging them. That is, we will check the two inequalities 5.9 and 5.10. Where $merge - threshold$ and r are two constants that could

be set by the user.

$$\frac{Similarity(\vec{v_{i2}}, \vec{v_{j2}})}{Similarity(\vec{v_{i1}}, \vec{v_{j1}})} > merge - threshold \tag{5.9}$$

$$\frac{Similarity(\vec{v_{i2}}, \vec{v_{j2}})}{Similarity(\vec{v_{i1}}, \vec{v_{j1}})} > r \cdot \frac{Size(c_i) + Size(c_j)}{InitialSize(c_i) + InitialSize(c_j)} \tag{5.10}$$

Figure 14 shows a demonstrative example of merging. Inside the clusters, "dx*", "dy*",and "dz*" mean the documents of category "x", "y", and "z", respectively. The new feature

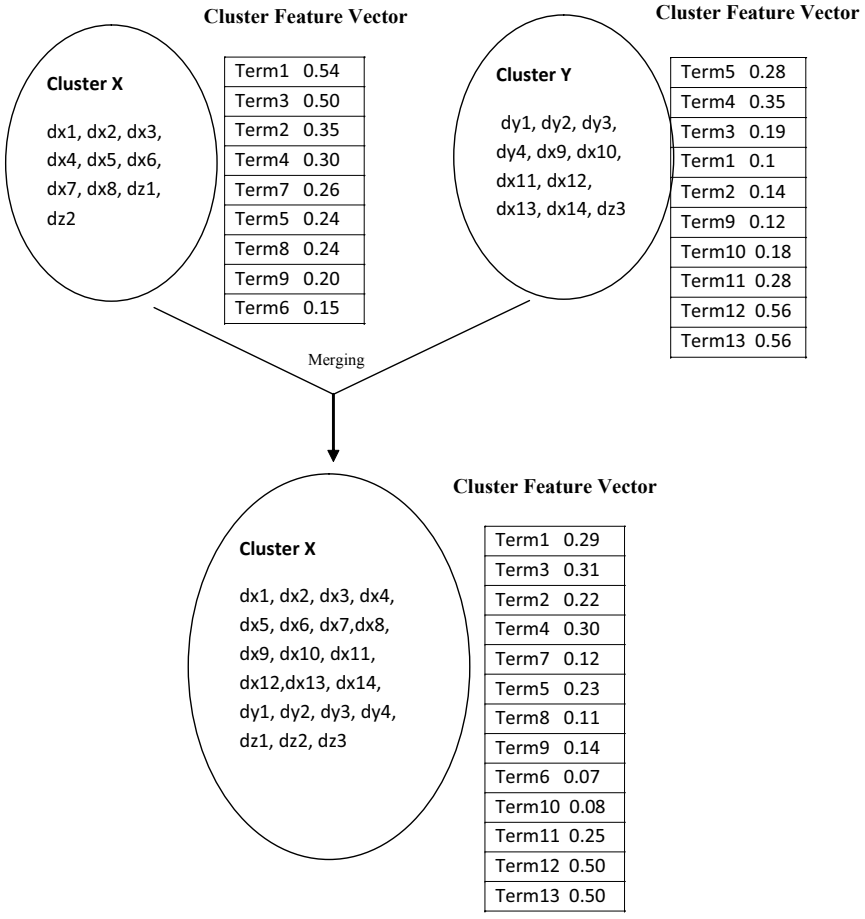

Figure 14: An Example of Cluster Merging

vector will be the mean of the two old feature vectors, and the new feature vector will be

normalized with equation 5.4, since the mean of the two unit vectors is not necessarily a unit vector with respect to the Euclidean norm. After merging, we may also check each document in other clusters to see if they should go to the new cluster or stay in its current cluster. The average recall of the resulting clusters usually increases. And we have the following theorem regarding this aspect.

Theorem 5.5.2. *If either of the following two conditions are met, the new cluster c resulted from merging two clusters c_A and c_B correctly labeled by categories A and B, respectively, will have a recall which is higher than or equal to the average recall of c_A and c_B, considering A and B as the same category after merging.*

(1)All the documents of categories A and B are in the two clusters c_A and c_B.

(2)$m > n$ & $\frac{m_1}{m_2} > \frac{n_1}{n_2}$, or $m < n$ & $\frac{m_1}{m_2} < \frac{n_1}{n_2}$. Where m is the total number of documents of category A. m_1 is the number of documents of category A in cluster c_A, that is, the number of correctly clustered documents of A. m_2 is the number of documents of category A in the other clusters, that is, the number of incorrectly clustered documents of A. n is the total number of documents of category B. n_1 is the number of documents of category B in cluster c_B, that is, the number of correctly clustered documents of B. And n_2 is the number of documents of category B in the other clusters, that is, the number of incorrectly clustered documents of B.

Proof First, if all the documents of A and B are in clusters c_A and c_B, after merging, the recall of the new cluster c will be 1. If all m documents of A are in c_A and all n documents of B are in c_B, the average recall of c_A and c_B is also 1. In any other situations, the average recall of c_A and c_B will be less than 1, and thus proving the theorem

with condition (1).

To prove the theorem with condition (2), first let us suppose that none of the m_2 documents of A is in cluster c_B, and none of the n_2 documents of B is in cluster c_A. With this assumption, we need to prove the following inequality.

$$(\frac{m_1}{m} + \frac{n_1}{n})/2 <= \frac{m_1 + n_1}{m + n} \tag{5.11}$$

By multiplying both sides with $mn(m+n)$ and doing some operations of cancelation, we end up with the following inequality.

$$n_1 m_2 m + n_2 m_1 n <= n_2 m_1 m + n_1 m_2 n \tag{5.12}$$

It can be changed to the following inequality.

$$\frac{n_1}{n_2}(m - n) <= \frac{m_1}{m_2}(m - n) \tag{5.13}$$

If condition (2) met, It is not hard to tell that the inequality 5.13 holds, and hence the inequality 5.11 holds.□

This proof is under the assumption that all m_2 A documents and n_2 B documents are in clusters other than c_A and c_B. Obviously, if some of m_2 and/or some of n_2 fall in c_B and/or c_A, respectively, inequality 5.11 still holds. This is because the numerator of the right side of 5.11 will increase, therefore, it still holds. The left and right hand sides of these inequalities will be equal if $m = n$.

It is easy to understand this theorem with condition (1). To help understand it with condition (2), let us look at the following example. Suppose $m = 200$ and $n = 100$, since $m > n$, if we have $\frac{m_1}{m_2} > \frac{n_1}{n_2}$, the recall will increase. Let $m_1 = 150$, $m_2 = 50$,

$n_1 = 50$, and $n_2 = 50$, then the average recall of the original two clusters will be $r_a = (\frac{150}{200} + \frac{50}{100})/2 = 0.625$, and the recall of the new cluster will be $r_n = \frac{50+150}{200+100} \approx 0.667$. However, if $\frac{m_1}{m_2} < \frac{n_1}{n_2}$, say, $n_1 = 80$ and $n_2 = 20$, then $r_a = (\frac{150}{200} + \frac{80}{100})/2 = 0.775$,and $r_n = \frac{80+150}{200+100} \approx 0.767$.

Note that the condition (2) is the lower bound in guaranteeing that the recall will increase. Sometimes, even if it is not satisfied, the recall may still increase. As in the above example, if $n_1 = 80$, $n_2 = 20$, $m_1 = 150$, and $m_2 = 50$, even though $\frac{m_1}{m_2} < \frac{n_1}{n_2}$, if some of m_2 documents fall into c_B, or some of n_2 documents fall into c_A (which is very likely given documents of these two categories are similar), say, totally 10 of m_2 and/or n_2 documents fall into c_B and/or c_A, then we would have $r_n = \frac{80+150+10}{200+100} = 0.8$, which is higher than $r_a = 0.775$.

In the case of fuzzy clustering, there is another option to decide whether to merge or not. That is, if they have many documents in common, we will merge them into one cluster.

From the above discussions on splitting and merging we can see that, even though our linear clustering algorithm CS2CS uses a fixed number of clusters (training data) as the starting point, it is unlike the K-Means clustering algorithm where the number of clusters are preset. By splitting and merging, it can automatically determine the number of clusters that better reflects the reality of the scientific community, where it is normal that new fields stand out and old fields merge, which results in the new distribution of scientific documents. Therefore, our algorithm is more suitable for realtime document clustering and trend discovering.

5.6 Selection of the Lengths of Feature Vectors

At level 1 feature selection, we choose the top 10 terms to form the feature vector of each reference cluster. Through subjective evaluation (manually checking the significance of the labels)and objective evaluation (comparing the accuracy of resulting clusters using different number of top terms) during our experimental test for our paper [99], this number is a good cutoff regarding the citation semantics.

At level 3 feature selection, the length of the feature vector of each cluster is determined by the length of the feature vector of each document belonging to it and the total number of documents in that cluster. So, the only issue left here is how to determine the length of the feature vector of each document, which is at the level 2 feature selection. A single document could be one in an existing cluster, or the new document to be added to a cluster. In dealing with the length of the feature vector of a single document, we must be aware of the two different situations. This is because we use the feature vectors of single documents to form the feature vector of the cluster they belong to, whereas we use the feature vector of the new document to compare with the feature vectors of existing clusters to decide where to put it. The objective criteria in both situations is which length of the feature vector of a document can lead to the best quality of document clustering.

When forming feature vectors of different clusters, we want each feature vector to be different from all others. We want the distance between every two cluster feature vectors to be as big as possible. Suppose a matrix M is formed with these feature vectors

in its columns, we want at least the following criteria to be satisfied.

$$Rank(M) = k \tag{5.14}$$

Where k is the number of existing clusters. That is, no cluster feature vector would be a linear combination of others. However, our situation here is different from latent semantic analysis [48], where SVD (singular value decomposition [50]) is used to $reduce$ the rank of the term-document matrix, in order to reveal the hidden similarity among documents and hence, to improve the recall in information retrieval. We do not want to reduce the rank of the matrix M. Instead, we want to keep its rank. We have the following theorem about the rank of this matrix.

Theorem 5.6.1. *If the number of unique terms in each cluster is bigger than the number of clusters, the lengths of the feature vectors that can satisfy equation 5.14 are not unique.*

Proof It can be shown by counterexamples. Let us suppose the number of clusters is k. First, since each cluster has more than k unique terms, for each cluster we can find a different term to form its feature vector. Then, the feature vectors will certainly satisfy the equation 5.14. If the theorem is false, we cannot find another length that satisfies equation 5.14. However, if we just add one term that is different from all the existing terms to one of these feature vectors, the resulted feature vectors still satisfy the equation 5.14 and therefore, we complete the proof of this theorem.□

Since the number of unique terms of cluster feature vectors are more than the number of clusters in most cases, there are so many different lengths that can satisfy equation 5.14. The lengths of cluster feature vectors are usually not a problem regarding this equation. Rather, our major concern is to reduce the length of each feature vector to eliminate

noise and to shorten runtime. In the meantime, we do not want to lose useful information. For example, if we have three clusters and the three feature vectors are “social: 1”, “database:1”, and “network:1”, then the length of each feature vector is one. Even though the rank of M will be 3, we may lose useful information that in turn may result in a low accuracy of clustering. Suppose the feature vector of a new document is “social:0.5, network: 0.5 “ with two words. For fuzzy clustering, the new document will go to clusters 1 and 3. Otherwise, it may only go to cluster 3. But if using two words for the feature vectors of these three clusters, they may be “social:1, network: 1”, “database:1, web 0.6”, “network:1, wireless: 0.8”, certainly, the new document should belong to cluster 1. (Notice that the weights of the words in this example will be normalized before comparison.) Therefore, we need to find the cutoff point of the length of the feature vectors of the existing documents. The principle rules of these cutoffs are that we want equation 5.14 to be satisfied (that is easy to achieve), and in the meantime we want to maximize the accuracy of resulted clustering.

While the length of the feature vector of an existing document has to be set heuristically with the requirement of equation 5.14 met, the length of the feature vector of a new document could be found automatically by searching for the following ratio R within a range of lengths $[L_l, L_r]$.

$$R = Max\{R_j, j = L_l, ..., L_r\} \tag{5.15}$$

$$R_j = Max\{\frac{S_i}{\sum_{i=1}^{k} S_i}, i = 1, ..., k\} \tag{5.16}$$

Where k is the number of existing clusters, S_i is the similarity between the feature vector of an existing cluster i and the feature vector with length j of a new document. This

means, we would use the length of the feature vector of a new document that makes it most similar to one of these feature vectors of clusters. In the case of fuzzy clustering, the numerator of 5.16 would be the top x of the similarities of a given length j. Even though the program needs to search a range of lengths, the time used is ignorable given the number k of the feature vectors of clusters is usually small. Furthermore, we designed two algorithms to speed up this search process: Exponential Increment Search and Linear Increment Search. Instead of checking each length in the range $[L_l, L_r]$, we only sample some of them to find the right length in less time. Our experimental results show the differences between using these two sampling search algorithms and the brute force search (check each length within the range$[L_l, L_r]$) are ignorable (as shown in Chapter experimentalResults). And the Exponential Increment Search requires the least amount of time. It is shown below.

(1) R_{max}*=0;*

(2) Increment=1;

(3) For($j = L_l; j \leq L_r; j = j + increment$*){*

(4) *Compute* R_j *using 5.16 with the current length;*

(5) *If(*$R_j > R_{max}$*){*

(6) $R_{max} = R_j$*;*

(7) *Increment = 1;*

(8) *Record the cluster that makes this* R_j*;*

(9) }

(10) *Else*

(11) *Increment = Increment*2;*

(12) }

For the Linear Increment Search algorithm, we only need to replace “Increment = Increment*2” with “Increment = Increment+1”.

Note that for each new document, we actually form two feature vectors. First, we form a feature vector to compare with the feature vectors of the existing clusters to decide where to put the new document. Second, we form another one to update the feature vector(s) of the cluster(s) to which this new document is added. They could be the same or different depending on the length set for the existing documents and that obtained for the new document. However, we could also use the same feature vector of the new document to update the cluster feature vector(s). Our experimental results showed the difference of the clusterings by using the fixed length of existing documents or the same length of the new document was ignorable (Table 35 in Subsection 7.3.7).

5.7 Use of Ontology

A domain ontology maintains the vocabulary of that domain. In other words, terms stored in an ontology are considered the most significant terms by the domain experts. We intuitively assume that if the domain ontology is utilized during the process of feature selection, namely, in adjusting the weights of terms of each feature vector, we would be able to get feature vectors which can better represent the documents in that particular domain. In our experiments, we used MeSH (Medical Subject Headings [18]), a popular ontology in the biomedical domain, in the process of forming feature vectors

of these biomedical documents. We increase the weights of terms found in MeSH. As expected, the results of using MeSH are better than that without MeSH (Table 15 in Subsection 7.3.3).

5.8 Fuzzy Clustering

In contrast to the hard clustering where a document can only belong to one cluster, the fuzzy clustering allows a document to belong to multiple clusters associated with a degree of belonging. In situations where fuzzy clustering (one object belonging to multiple clusters) is needed, our CS2CS clustering algorithm can be easily adapted. The algorithm (which is similar to that discussed in Section 5.4) is as follows.

(1) For the new document
(2) do level 2 feature selection to get the document feature vector
(3) For each cluster
(4) Compute the similarity between the document feature vector of the new
(5) document and the cluster feature vector
(6) For each cluster
(7) Compute the degree of membership of the new document to this cluster
(8) Assign this document with memberships to the top x clusters to which it is
(9) most similar regarding their feature vectors
(10) Update the feature vector(s) of the cluster(s) to which the new document was
(11) just added with level 3 feature selection.

Instead of putting the new document to the cluster whose feature vector is most

similar to the feature vector of the new document, we can put it to multiple clusters with parameter representing the degree of belonging or membership. The degree of membership of document d with respect to cluster C out of the k clusters is calculated with the following equation.

$$D_{dc} = \frac{S_{dc}}{\sum_{j=1}^{k} S_{dj}} \tag{5.17}$$

Where S_{dj} is the similarity between document d and cluster j after the ratio R in equation 5.15 is determined. There are two ways to decide how many and which clusters a document should belong to. First, the user can set how many clusters a document can belong to, say 3, then document d will be put to the three clusters whose similarities with document d are in the top 3 among all the k similarities. Secondly, the user can choose to use a threshold of degree of membership, say D_{min}, if $D_{dc} > D_{min}$, document d will be put to cluster C. Of course, the user can also choose to set the threshold of similarity, but it would require more insight knowledge than setting the threshold of degree of membership.

When updating the cluster feature vector, instead of adding new occurrences to the raw cluster feature vector as discussed in Section 5.4, we add the degree of membership of each term found in the document feature vector of the new document to the raw cluster feature vector, then normalize all the raw cluster feature vector with formulas 5.2, 5.3, and 5.4.

Figures 15 and 16 show a demonstrative example of the cluster feature vectors before and after adding a new document and the similarities between them. Since they are

all unit feature vectors, the similarities between them are calculated with formula 5.5. Degrees of memberships are obtained through formula 5.17. It is the case of simplest fuzzy clustering, that is, a document is assigned to one cluster with the degree of membership. Note the new document was added to the cluster represented by the cluster feature vector at the top in Figure 15.

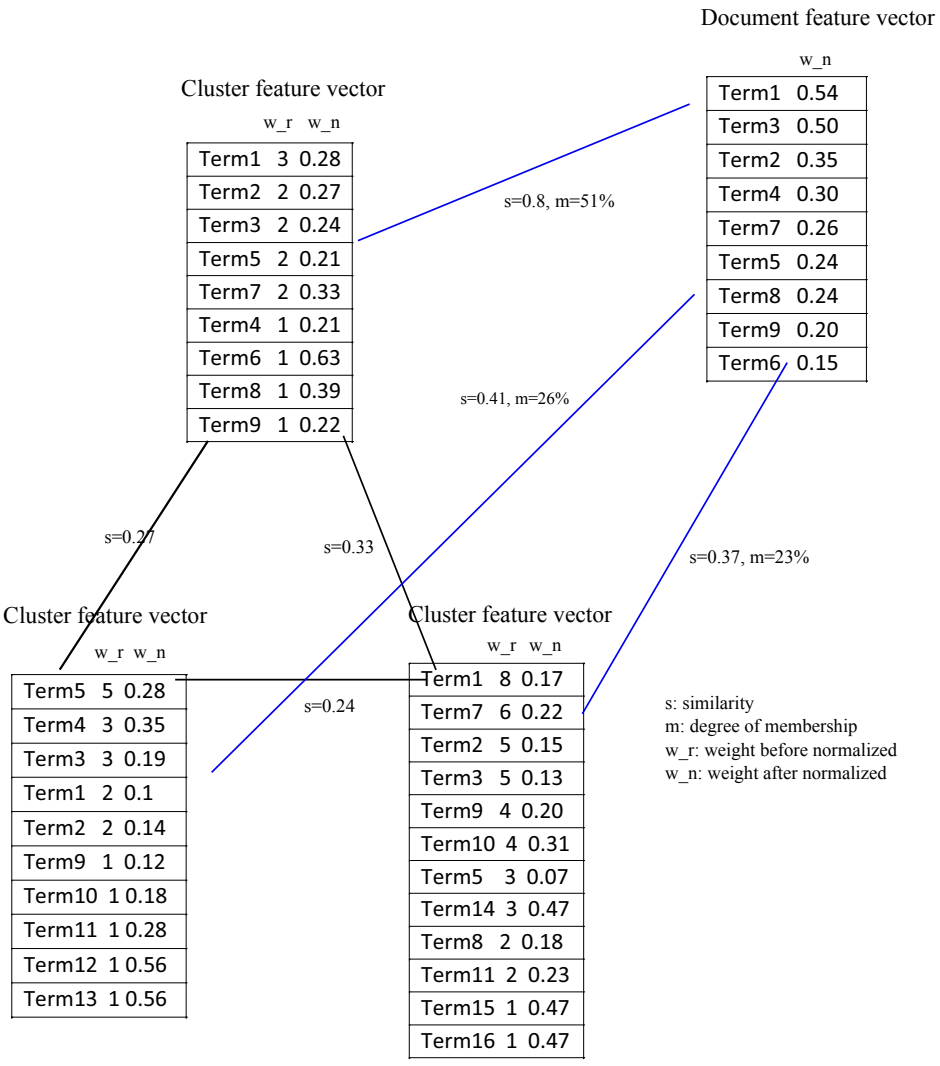

Figure 15: An Example of CS2CS Fuzzy Clustering – Before Adding a New Document

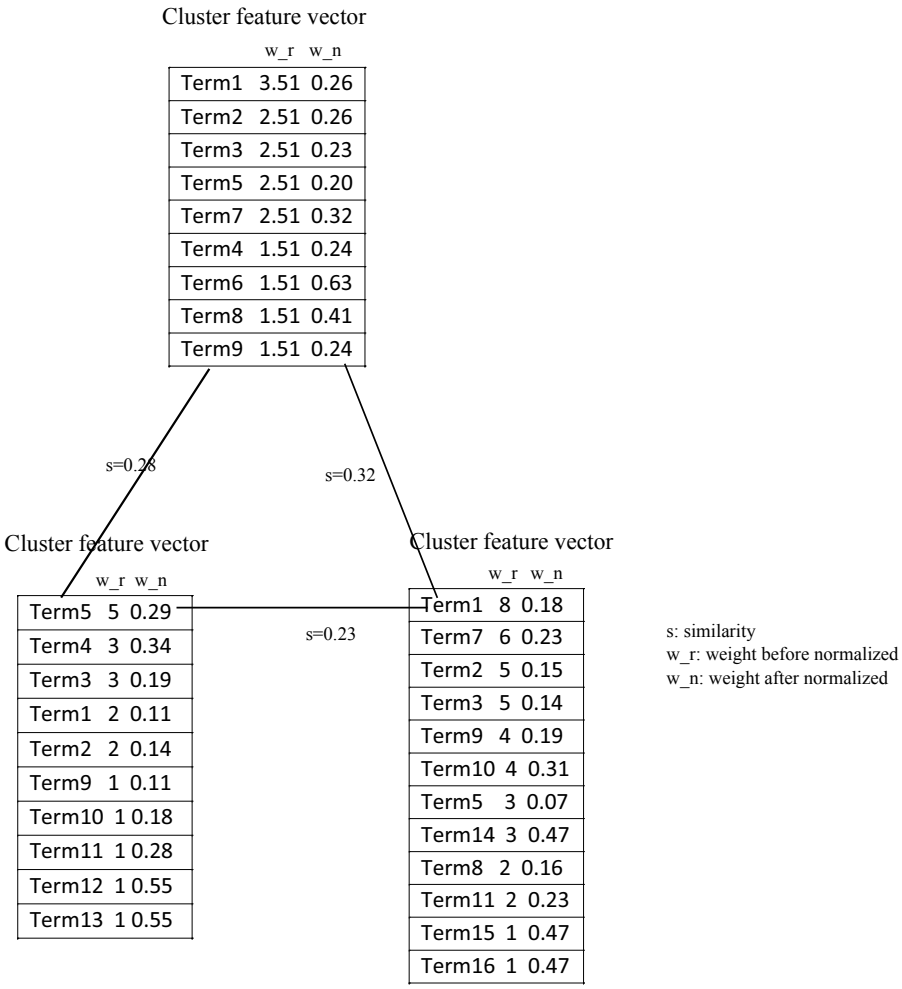

Figure 16: An Example of CS2CS Fuzzy Clustering – After Adding a New Document

5.9 Complexity Analysis

Suppose the training step is used, the overall runtime complexity of CS2CS will be the following:

$$O(k(m-k)^2t + cm + kn) = O(ktm^2 + kn) \tag{5.18}$$

Where m is the number of documents in the training set, k is the number of clusters, n is the number of the new documents to be clustered is n, and t is the number of iterations in the K-Medoids clustering that is used in the training process to find weights. The complexity of the K-Medoids is $O(k(m-k)^2t)$. To find citation semantics (to cluster references and label them), we only need to look at each document once and hence, the runtime for this part is $O(cm)$. For the linear clustering stage the time needed is $O(kn)$ because for each new document we only need to compare its feature vector with the k feature vectors of existing clusters.

Even though we use a training set with $m \approx n/4$ in the following experiments, in practice, the training set will be far less than the set of new documents. That is, we could have $m \ll \sqrt{n}$. For example, if m=100, n could be more than 100,000, or even more than 1 million. The quality of clustering will not decrease due to the increased number of new documents. This is because once we get the initial feature vectors of clusters, they evolve as new documents are added in to better reflect the new contents. The overall runtime of the CS2CS approach is $O(n+m)$ given $m \ll \sqrt{n}$. In other words, it is linear with respect to the total number of documents. Since complexity of K-Medoids algorithm is quadratic ($O(kt(n-k)^2)$) in terms of the number documents n, CS2CS is much faster than K-Medoids. It is even faster than K-Means, even though K-Means is also a linear algorithm. This is because its complexity is actually $O(ktn)$, where t is the number of iterations used. Its coefficient is bigger than that of CS2CS. Our experiments (Table 12 in Subsection 7.3.1)verify this analysis. Even though k-Means may be faster than CS2CS if a bigger training set is used in CS2CS, its accuracy is usually far less than CS2CS.

If we carry out splitting and merging during linear clustering in cases where the categories of new documents are unknown, we need to add k^2n^2 to equation 5.18. That is the worst case when we check for splitting and merging after adding each new document. This is because that, after adding a new document, we need to look at the newly updated cluster to see whether we need to split it, and compare its feature vector with other clusters' to see whether we need to merge them. Once we merge two of them, we may choose to compare the newly merged again with others to see if we need to merge more. This process takes time k^2x, where x is the number of documents involved in splitting and/or merging. So the run time of adding n new documents will end up with k^2n^2 in the worst case if we choose to check for splitting and merging after adding each new document. However, this worst case could be avoided by using a different strategy on when to check for splitting/merging. As a matter of fact, it does not make much sense to check for splitting/merging after adding each new document since the feature vector of one document usually will not affect the feature vector of the cluster too much. That is given, we may check for splitting/merging after adding a significant amount of documents, say after the original set is doubled. In this case, we will add k^2nlogn to equation 5.18. Clearly, the total complexity would be $O(nlogn)$ which is close to linear time regarding the number of documents being added.

Not only does CS2CS run fast, it also uses less memory compared to the other algorithms. Since it uses cluster feature vectors and the document feature vector of the new document to do incremental clustering, it only needs the k cluster feature vectors and the document featrure vector to be in memory and hence, its space complexity is only

Table 3: Complexities of Document Clustering Algorithms

Algorithm	Runtime	Space
K-Medoids	$O(kt(n-k)^2)$	$O(n)$
K-Means	$O(ktn)$	$O(n)$
CS2CS	$O(n)$	$O(k)$
CS2CS with Splitting&Merging	$O(nlogn)$	$O(k), O(n)$

n - number of documents, k - number of clusters, t - number of iterations

$O(k)$. Only when carrying out splitting or merging, the space complexity is $O(n)$. Table 3 summarizes the comparison of complexities (runtime and space) of these document clustering algorithms we just discussed.

CHAPTER 6

INTEROBO: A FRAMEWORK FOR KNOWLEDGE SHARING IN BIOMEDICAL DOMAIN

In the previous chapter, taking MeSH as an example, we have shown ontologies are useful in document clustering in particular, and hence in knowledge discovery in general. In this chapter we are going to analyze the overlapping relationships among ontologies, and provide an interoperability framework for sharing biomedical knowledge across OBO communities. Our ontology modeling methods are comprised of modeling the relations, computing overlapping of the ontologies, clustering ontologies, building ontology networks, and querying and inferencing in the ontology network.

To provide integrated access to data annotated with different ontologies, an important requirement is to relate these ontologies. This is commonly done by cross referencing concepts from these ontologies. Although using a reference ontology to map multiple ontologies is very promising, having an ideal reference ontology is not easy, and it is often hard to find concepts from the reference ontology for mapping between ontologies. In our model, we identify common characteristics of ontologies in diverse biomedical ontology domain (OBO) and cluster them using these features. We also focus on the analysis of semantic relationships that commonly appear in these ontology domains. Our approach differs from and complements the related approaches described later. Specifically, we use pragmatic approaches to characterize and cluster ontologies, without relying on reference or upper ontologies.

In this study (Part of it has been published in [46]), we have evaluated our approach by performing experiments using the OBO dataset to analyze diverse biomedical ontologies. Given the large number of the biomedical ontologies, we focus on the analysis of semantic overlapping relationships inherent in the ontologies. In particular, we measure the similarity between ontologies by considering synonym-based connectivity patterns and analysis of shared concepts and relations across different ontologies. We cluster the ontologies using the developed similarity measures and show quantitative evaluations of the utility of the proposed models.

6.1 Domain Overlapping Model

We propose a domain overlapping model, called the InterOBO (Figure 17) that describes the characteristics and patterns of knowledge sharing between ontologies. We first present some basic definitions that are integral to understanding the domain sharing model.

A Concept-level relation (CR) is a binary relation CR between a concept $c1$ and a concept $c2$. It expresses any kind of relationship between a concept $c1$ and a concept $c2$. The concepts c1 and c2 may be either from the same ontology or from different ontologies. In our study, relationships are defined by the empirical analysis of ontology data. Apart from being similar, concepts may share other aspects, e.g., sharing the same parents, children or siblings. This forces us to think not only in terms of concepts per se, but in terms of edges and other structural aspects of the concepts.

An Ontology-level relation (OR) expresses any kind of relationship between an

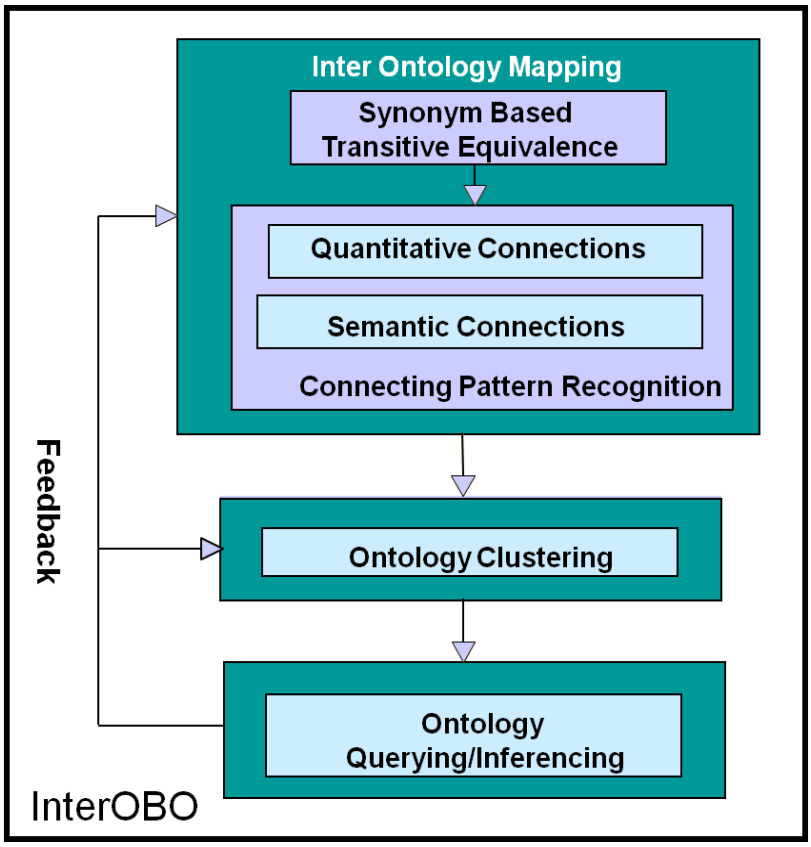

Figure 17: InterOBO Framework

ontology $o1$ and another ontology $o2$. In this chapter, we introduce two types of OR relations. Firstly, OR can be defined as a unified view of relationships between different ontologies. This means that the CR level relationships for different ontologies are accumulated at the OR level and defined as a new "sharing" relationship. Secondly, OR can be defined by synonym relationships in the CR level. We refer to this as "synonym-based transitivity" because some transitivity can be defined even between ontologies.

Our ontology modeling methods are comprised of the following steps: 1) synonym based transitive equivalence, 2) connecting pattern recognition for inter ontology mapping

3) clustering the ontologies based on the overlapping patterns and 4) finally querying and inferencing over the clusters. These procedures are in turn subdivided into specific elementary steps.

The sharing relationship among ontologies is based on the degree of sharing between ontologies. For the sharing relationships, we define the following two specialized relations: a) synonym based transitivity and b) connecting pattern based on overlapping relations. The latter can be measured in terms of concept overlapping and structural (edge) overlapping. The degree of sharing between ontologies is used in the step of clustering ontologies. It is determined by the following two aspects: a) sharing concepts and b) sharing edges or paths. Finally, the clustered ontologies are further structured as an ontology network. This network facilitates to integrate data over the ontology network and discover a path of reasoning from specific capability through the network. There are also feedback channels among clustering component, sharing computing component, ontology query/inferencing component. Note that the inferencing using this ontology patterns and clustering is beyond the scope of this dissertation.

6.2 The Open Biological and Biomedical Ontology (OBO) Domain

The Open Biomedical Ontologies are well-structured controlled vocabularies for shared use across different biological and medical domains. The OBO represents community-based efforts to support a range of ontologies designed for biomedical domains. Some of them are generic and apply across all organisms while others are more restricted to specific domains.

For this study, we have analyzed all the concepts of the 40 OBOs (version June 12, 2006). As shown in Table 4, the total number of the concepts is 13,456. After filtering duplications, we obtained 122,390 unique concepts. The maximum, average, and minimum concept counts per ontology have been computed. Ontology counts for each concept have been computed as well. These data have been extracted from the OBO text and OWL files, and stored into a local database. More detailed information about these

Table 4: The OBO Ontologies

Ontology Features		**Number**
Number of Ontologies		40
Total Concept#		134567
Unique Concept#		122390
Concept# per Ontology	Maximum	39
	Minimum	1
	Average	1.6
Ontology# per Concept	Maximum	9
	Minimum	2
	Average	2.4

40 ontologies are show in Table 5. The following analysis (concepts, synonym, node and edge) has been performed by considering a single type of relation such as IS-A xor Part-of for the sake of simplicity. For instance, GO has both IS-A and Part-of relationships, but we have mainly considered the commoner IS-A relation.

6.3 Ontology Mapping Methods

6.3.1 Synonym Based Transitive Equivalence

In order to provide sharing relations among multiple ontologies, we need to provide an advanced ontology mapping schema. A frequent phenomenon across domains is the presence of homonyms and synonyms. In the ontology mapping process, a concept

Table 5: OBOs in Detail

ID	Ontology Name	Conc.	Syn.	Node	Edge	Type
O1	Adult_mouse_anatomy	2703	0	2971	2820	IS-A
O2	Arabidopsis_development	108	53	108	108	IS-A
O3	Attribute_and_value	1228	0	1260	1248	IS-A
O4	Brenda	2218	1179	2515	2353	IS-A
O5	Cell	761	0	964	964	IS-A
O6	Chebi	12734	23451	14666	14666	IS-A
O7	Dictyostelium_discoideum_anatomy	38	13	73	58	IS-A
O8	Disease	19136	0	19389	19383	IS-A
O9	Emap	13731	0	13731	13731	Part-of
O10	Event	2665	234	3499	3020	IS-A
O11	Evidence_code	130	6	163	140	IS-A
O12	Fly_anatomy	6130	0	13649	7273	IS-A
O13	Flybase_vocab	660	65	664	664	IS-A
O14	Fungal_anatomy	65	15	82	76	IS-A
O15	GO	20733	17181	27574	27560	IS-A
O16	Human_dev_anat_abstract	2314	0	2339	2324	Part-of
O17	Human_dev_anat_staged	8340	0	8362	8339	Part-of
O18	Image	259	30	259	259	IS-A
O19	Loggerhead_nesting	308	2	322	317	IS-A
O20	Mammalian_phenotype	4186	3010	6130	4630	IS-A
O21	Mao	164	45	164	164	IS-A
O22	Medaka_anatomy_development	4358	0	4404	4245	Part-of
O23	MeSH	15337	33297	19525	19525	IS-A
O24	Molecule_role	7255	23588	7641	7393	IS-A
O25	Mosquito_anatomy	1804	3501	2290	2057	Part-of
O26	Mouse_pathology	459	0	459	459	IS-A
O27	Pathway	486	62	554	554	IS-A
O28	Plant_environment	489	308	518	506	IS-A
O29	Plant_trait	761	44	949	865	IS-A
O30	Plasmodium_life_cycle	47	0	98	64	IS-A
O31	Po_anatomy	763	218	785	785	IS-A
O32	Po_temporal	274	996	274	274	IS-A
O33	Psi_mi	194	165	223	212	IS-A
O34	Rex	546	140	1099	671	IS-A
O35	Sequence	1034	251	1171	1094	IS-A
O36	Temporal_gramene	235	168	235	235	IS-A
O37	Worm_development	69	0	69	69	IS-A
O38	Zea_mays_anatomy	179	30	181	141	Part-of
O39	Zebrafish_anatomy	1558	0	2184	1553	Part-of
O40	Fly_development	120	0	124	124	IS-A

in an OBO ontology can be associated with another concept in another OBO ontology if both concepts have a synonym relation. The synonym relation is reflexive, transitive, and symmetric. We propose three types of the synonym relation that can be identified in the ontology to ontology mapping of a concept.

There are many different meanings for the same word. For instance, “PGA” stands for “Polyglandular Autoimmune Syndrome” and the “Professional Golfers’ Association.” Synonyms are used to relate to each other. For instance, some refer to “stomach acid” with “Betaine HCl,” others use “Hydrochloric Acid”. Resolving these semantic problems present across multiple ontologies is a difficult task because it requires a comprehensive understanding of ontologies to be linked and implications of the mapping. These differences occur because different ontology designers may bring different world views to the task, conceptualizing the world at different levels of granularity and abstraction. Such differences are commonly considered semantic problems.

To handle these semantic problems, we have identified three kinds of synonym relationships between ontologies. An ontology $O1$ can be related to another ontology $O2$ through synonyms of concepts. A concept X in $O1$ can be synonymously related to another concept Y from $O2$ if 1) Y is included as a synonym of X in $O1$ or 2) if X is specified as a synonym of Y in $O2$. The confidence in the semantic equivalence of X and Y is strengthened if they are mutually defined as synonyms of each other in their ontologies. Another scenario that is indicative of semantic equivalence is when X and Y are linked through having a common synonym. Note that the case of exact matches between X and Y is trivial and not a case of synonym-based transitive equivalence. Figure

18 shows these relations, also formally defined below.

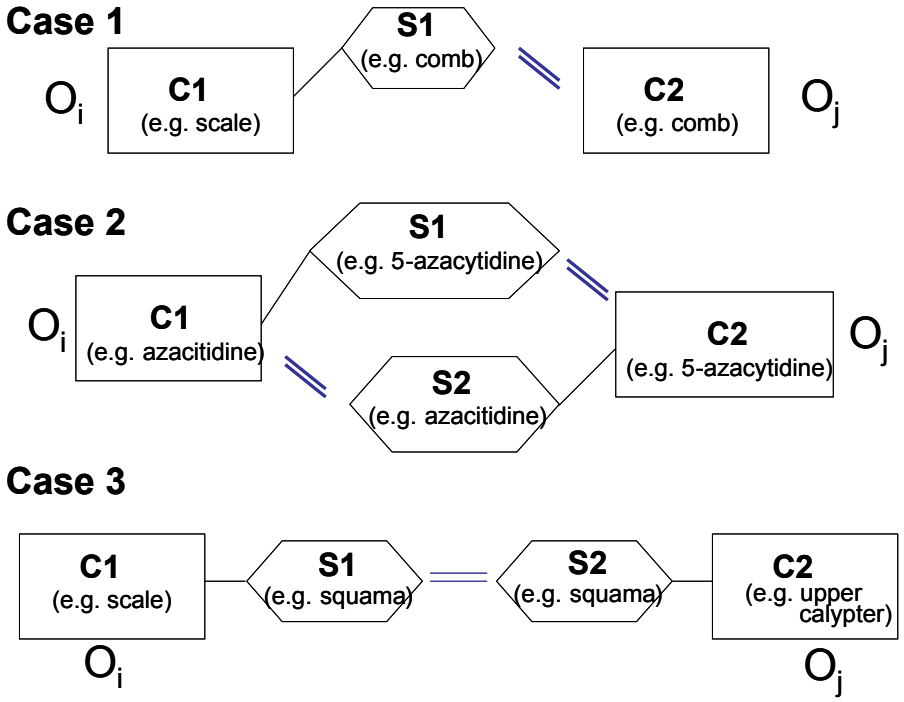

Figure 18: Synonym Relations Between Ontologies

Let $Ci \in Oi$ and $Cj \in Oj$, where Oi and Oj are ontologies, and $Si \in S'$ and $Sj \in S$", where S' and S" be a set of synonyms of Ci and Cj, respectively. The following three cases might be considered for synonym based transitivity across different ontologies. The symbol $\approx$ is used to represent a synonym relation between concepts.

Case 1 If Si is a synonym of Ci (i.e., $Si \approx Ci$) or Sj is a synonym of Cj (i.e., $Sj \approx Cj$) and either $Ci = Sj$ or $Cj = Si$, but $Ci \neq Cj$, then $Ci \approx Cj$ can be transitively retrieved from either $Ci = Sj \approx Cj$ or $Cj = Si \approx Ci$. As an example, Medicine is a concept in MeSH and Drug is a concept in CheBi and Medicine is defined as a synonym of Drug in CheBi. Therefore, Medicine in MeSH is synonymously related

to Drug in CheBi.

Case 2 Si is a synonym of Ci (i.e., $Si \approx Ci$) and Sj is a synonym of Cj (i.e., $Sj \approx Cj$) and $Ci = Sj$ and $Cj = Si$, but $Ci \neq Cj$, then $Ci \approx Cj$ can be transitively retrieved from $Ci = Sj \approx Cj$ and $Cj = Si \approx Ci$. As an example, Polysome is a concept and Polyribosome is its synonym in GO while Polyribosome is a concept and Polysome is it synonym in MeSH. Therefore, Polysome in GO is synonymously related to Polyribosome in MeSH.

Case 3 If Si is a synonym of Ci (i.e., $Si \approx Ci$) and Sj is a synonym of Cj (i.e., $Sj \approx Cj$) and $Si = Sj$, but $Ci \neq Cj$ then $Ci \approx Cj$ can be transitively retrieved from $Ci \approx Si = Sj \approx Cj$. As an example, Heterozygote is a concept and Carrier is its synonym in MeSH and Transporter is a concept and Carrier is its synonym in Molecule role. Therefore, Heterozygote is synonymously related to Transporter.

6.3.2 Ontology Connecting Patterns

We are interested in relating distributed ontologies that share a common domain. In order to relate multiple ontologies, we use the notion of frequently recurring patterns in overlapping ontologies. The idea of patterns has been widely used in building software system. The motivation behind characterizing a pattern is to fully utilize known solutions for commonly recurring problems in a specific context. The focus of research on patterns has so far been mainly on ontology modeling and knowledge reuse such as ontology construction and management. Here we are introducing an initial method for exploiting ontology connecting patterns with the aim of further expanding ontology space

for query and inferencing. Two kinds of connecting patterns are discussed: quantitative connections and semantic connections. The first type of pattern is defined based on the quantity of overlapping while the latter is based on the connecting position of concepts in these ontologies.

Quantitatively connecting patterns The overall connectivity patterns between two ontologies can be identified from analyzing the extent to which concepts from one ontology are mapped to the other. Formally, the connectivity can be defined in terms of linguistic overlapping (concepts and synonyms) and structural overlapping aspects (links and paths). An ontology O can be defined as a set of constituent concepts, relations and properties, namely $< O >$. We now define the size of the concept set, (i.e., $cs(O) = \| < CO > \|$, where $< CO >$ is the set of concepts in the ontology O) and the size of the link set, (i.e., $ls(O) = \| < LO > \|$, where $< LO >$ is the set of the link type (such as IS-A or Part-of) of the ontology hierarchy). We consider two types of the relationships: direct and indirect. The direct relationship defines a parent and child relationship of the given concepts in the hierarchy. The indirect relationship defines a predecessor/successor relationship of given concepts (path) in the hierarchy. The degree of concept overlap $cp(O1, O2)$ and the degree of link overlap $lp(O1, O2)$ is computed by the formulas below:

$$cp1(O1, O2) = \frac{cs(O1) \cap cs(O2)}{cs(O1) \cup cs(O2) - cs(O1) \cap cs(O2)} label(conceptOverlap_1) \quad (6.1)$$

$$cp2(O1, O2) = \frac{cs(O1) \cap cs(O2)}{cs(O1)} \cdot \frac{cs(O1) \cap cs(O2)}{cs(O2)} label(conceptOverlap_2) \quad (6.2)$$

$$lp1(O1, O2) = \frac{ls(O1) \cap ls(O2)}{ls(O1) \cup ls(O2) - ls(O1) \cap ls(O2)} label(linkOverlap_1) \quad (6.3)$$

$$lp2(O1, O2) = \frac{ls(O1) \cap ls(O2)}{ls(O1)} \cdot \frac{ls(O1) \cap ls(O2)}{ls(O2)} label(linkOverlap_2) \quad (6.4)$$

The relationship between ontologies $O1$ and $O2$ can be as follows:

- $O1$ is a subset of $O2$, i.e. $O1 \subseteq O2$, or $O2$ is a subset of $O1$, i.e. $O1 \supseteq O2$.
- $O1$ partially overlaps $O2$, i.e. $\exists x, y, (x \in O1 \wedge x \in O2) \wedge (y \in O1 \wedge y \notin O2)$
- $O1$ is disjoint from $O2$, i.e. $O1 \cup O2 = \phi$

An ontology mapping from $O1 = (cs1, ls1)$ to $O2 = (cs2, ls2)$ is defined as follows: There is a subset ontology mapping from $O1 = (cs1, ls1)$ to $O2 = (cs2, ls2)$ if there exists $cs1 \subseteq cs2$ and $ls1 \subseteq ls2$. There is a partial overlapping from $O1 = (cs1, ls1)$ to $O2 = (cs2, ls2)$ if there exists $\exists a, b, (a \in cs1 \wedge a \in cs2) \wedge (b \in cs1 \wedge b \notin cs2)$ and $\exists c, d, (c \in ls1 \wedge c \in ls2) \wedge (d \in ls1 \wedge d \notin ls2)$

Semantically Connecting Patterns This connecting pattern focuses on representing inter-ontology relationships that might exist between multiple ontologies. For instance, an ontology can be a more specific ontology of another ontology (upper ontology). In this case, there is a super/subclass relationship between these two ontologies. Or there might be a sibling relationship between ontologies. Assume that the ontology Oi and a concept x are given. In the following formulae, $level(x@Oi)$ means the level of the concept x at the ontology Oi and $depth(Oi)$ means the depth of ontology Oi. The Concept Connection Position (CCP) is computed as follows:

$$CCP(x, Oi) = \frac{level(x@Oi)}{depth(Oi)} \tag{6.5}$$

The Ontology Connection Position (OCP) is computed based on the relative position of the concept in two ontologies, indicating the positions of the concept from these two ontology perspectives. Assuming that two ontologies Oi and Oj and a concept x are given, OCP is computed as follows:

$$OCP(x, Oi, Oj) = \frac{CCP(x, Oi)}{CCP(x, Oj)} \tag{6.6}$$

There might be multiple connection patterns in multiple ontologies. Thus, it is necessary to accumulate the connecting patterns and normalize them into an accumulated connection score using a simple weight average formula which summarizes all weighted OCPs. The weight for each pattern can be defined by a domain expert based on the significance of the concept or simply as a uniform weight. The Accumulated Ontology Connection (AOC) score can be computed as follows:

$$AOC(Oi, Oj) = \sum_{i} OCP_i \cdot W_i \tag{6.7}$$

The connection pattern is a frequently recurring pattern observed during the ontology overlapping analysis used to connect an ontology to another. This pattern is mainly based on the location of the concept overlapping between ontologies.

1)Ontology $O1$ is quantitatively connected to Ontology $O2$. Let us assume that a concept in ontology $O1$ is connected to a concept in another ontology $O2$ and $count$ means the number of the common concepts x. In this case, we map the class in $O1$ to the class in $O2$, with the mapping being either equivalent or synonymously equivalent. Given a threshold μ: O1 is quantitatively connected to O2 if $\exists x, (x \in O1 \wedge x \in O2) \wedge (count(x@O1) > \mu) \wedge (count(c@O2) > \mu)$

2)Ontology $O1$ is semantically connected to ontology $O2$. This means that the concepts in $O1$ can be semantically connected to the concepts in $O2$. In this case, a concept x is located at a low level in ontology $O1$ while the same concept x is located at

a high level in ontology $O2$. Note that the synonym patterns described in the synonym-based transitive equivalence section have been incorporated into this semantic connection pattern. For instance, Table 42 in Chapter 7 shows the synonym pattern Cell death . Necrosis. Furthermore, if the number of connecting patterns between O1 and O2 is higher than a certain threshold defined by a domain expert, then the ontology O1 is a specialised ontology of the ontology O2. Then we can say the subconcepts of x in O2 are semantically related to the superconcepts of x in O1.

O1 is semantically connected to O2 if $\exists x, (x \in O1 \wedge x \in O2) \wedge (OCP(x, O1, O2) > \alpha \vee OCP(x, O2, O1) > \alpha)$

Based on these pairwise similarity measures for concept and edge overlapping relationships, we have developed a simple ontology model for clustering overlap in multiple ontologies. This is discussed in the following section Ontology Clustering. These connecting patterns can be essentially used for automatically connecting ontologies and expanding the query space of ontologies, and to retrieve information from available knowledge sources within the ontology space.

6.4 Ontology Clustering

We posit that ontology clustering is a required step for efficient ontology mapping involving the alignment and merging of ontologies. Here we clarify our approach to ontology mapping within the above theoretical framework. An ontology mapping consists of a collection of several relationships between multiple ontologies. Given that ontologies are more closely related to some ontologies than others, ontology mapping can be

clarified through ontology clustering the task of classifying a collection of ontologies into clusters. The guiding principle is to minimize interclass similarity and maximize intraclass similarity, based on the notion of semantic distance. To discover the correlation between ontologies, we used the MCL [47]. We compute and analyze correlation based on the common concepts between different ontologies.

The steps to compute the degree of overlap between ontologies and do clustering are as follows:

- For every pair of the OBO ontologies, determine the set of concepts in common.
- Calculate the overall similarity for each pair of ontologies using the following formulas and store the values into respective summary matrices:
  - probability-based similarity (Approach I) $PS = (A \cap B/A) \cdot (A \cap B/B)$
  - area-based similarity(Approach II) $AS = A \cap B/(A \cup B - A \cap B)$

    In the above formulas, $(A \cap B)$ refers to the number of concepts (or, separately, edges) common to both ontologies, while $(A \cup B)$ represents the total number of unique concepts (or, separately, edges) present in either of the two ontologies under consideration.
- For each of the two 40 by 40 upper-triangle matrices, cluster the ontologies using the MCL algorithm to obtain the respective clustering.

CHAPTER 7

EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter, we present the experimental results for reference clustering (or citation clustering – we use these two terms interchangeably in this dissertation), CS-VS – document clustering using combined vector space and citation semantics measures, and CS2CS – document clustering with a 3-level feature selection. We also analyze the results and discuss their significance.

7.1 Experiments on Reference Clustering

We downloaded all 42 papers from the Search track of recent World Wide Web conference websites: www.www2007.org, www.www2006.org, www.www2005.org, www.www2003.org, and www.www2002.org (Website for WWW 2004 was inaccessible). Based on the nature of contextual information used, we attempted six different approaches – keyword matching, locality clustering, and four MCL clustering approaches, to classifying or clustering the citations for each of the 42 papers separately.

7.1.1 Approach 1: Keyword Matching

In this approach, we use each specified keyword in the paper as a class or cluster label. We try to map each reference title and its surrounding sentence to each keyword. If such a mapping exists, we put this reference into the cluster labeled by this keyword.

The surrounding sentence refers to part of the sentence close to a reference number

in the paper, either before or after the number. For example, in "The threshold algorithm works as follows [12]," the words "The threshold algorithm works as follows" are taken as the surrounding sentence of reference 12. In "Jones et al. [10] examine substitutions that searchers make to their queries," "examine substitutions that searchers make to their queries" is treated as the surrounding sentence for reference 10.

7.1.2 Approach 2: Locality Clustering

In this approach, we use the explicit grouping (we call it bracket information or citation locality) provided in the body of the paper to cluster the references. That is, if two references are mentioned together in a paper, they will belong to one cluster. For example, if we see "[13, 21]", then reference 13 and 21 in that paper, are taken as being in the same cluster.

7.1.3 Approaches 3-6: MCL (Markov Cluster Algorithm) Clustering

In these approaches, we calculate the similarity between every two references r_1 and r_2 as follows.

$$S(r_1, r_2) = S_1(+S_2)(+S_3)(+S_4). \tag{7.1}$$

Where S_1 is the similarity between references titles, S_2 is the similarity between the surrounding sentences, S_3 is the similarity between the combination of titles and surrounding sentences, and $S_4 = 1$ if the two references are mentioned together in the paper such as "[2, 10]". Otherwise, $S_4 = 0$. S_1, S_2, and S_3 are calculated by formula 3.2.

In approach 3, we use only S_1 as the similarity of two references; in approach 4, we use $S_1 + S_2$; in approach 5, we use $S_1 + S_2 + S_3$; and $S_1 + S_2 + S_3 + S_4$ are

used in approach 6. These approaches are referred to as MCL-0, MCL-1, MCL-2, MCL-3 respectively. After calculating the similarities, we use them as the inputs to MCL to cluster these references.

7.1.4 Labeling

Our strategy for labeling the reference clusters is as follows. There are basically two steps. For each cluster, we first compare it to the clusters obtained by approach 1. If half or more than half the number of the references fall into any of those keyword labeled clusters, we use that keyword as the label. Otherwise, we need step two. In this step, we first find the frequency (occurrence) of the term. If a term occurs in half or more than half the number of the members of a reference cluster, and it occurs at least twice, then we use it as one of the labels for this cluster. Here we require a term occurs at least twice to be considered because there might be only two references in a cluster, then every term will occur in at least half of the cluster members. It does not make sense if we use all the terms to be the labels of the cluster. This requirement will avoid such meaningless labeling. In the case of multi-word terms used for labeling, if a term is part of another term, and they have same occurrences, we just keep the longer one. After getting all the labels for a cluster, we then sort the labels according to their scores. The score of term X is calculated as follow:

$$Score(X) = Occurrence(X) * Number\_of\_words(X) \tag{7.2}$$

In this way, we favor longer phrase over shorter ones. For example, suppose a cluster has five references. If “web” occurs in all these references, and “semantic web” occurs in

three of them, according to our equation 7.2, the score of “web” will be 5, and the score of “semantic web” will be 6. Therefore, we rank “semantic web” higher than “web” in the list of labels. In other words, we consider “semantic web” as a more appropriate label than “web” for this particular cluster.

7.1.5 Experimental Results

Through evaluating the results of these approaches, approach 6 (MCL-3) turned out to be the best. We will show the comparisons in next subsection. First we want to summarize the results of using approach 6 (MCL-3).

The number of clusters of references for each paper ranged from 1-10, with an average value of 4.5, and standard deviation of 2.0. We also analyzed the keywords for the papers. The number of keywords in a paper ranged from 0 to 7, with an average value of 3.7, and a standard deviation of 1.7. The following numbers are for all the citations in all papers taken together.

1. Total number of clusters: 190
2. Total number of labeled clusters: 169
3. Total number of clusters labeled by keyword: 34
4. Number of unique labels: 608
5. Number of unique keywords: 128

This demonstrates that as much as 88.9% of the clusters could be automatically labeled by approach 6. This contrasts with only 17.9% of the clusters that could be labeled by keywords based on explicit matches. More than 4-fold new terms could be generated

to describe the citation clusters compared to the number of keywords in the citing paper.

7.1.6 Evaluation

In this subsection, we address the relative performance of the different approaches used for semantic classification of the references. Ideally, the perfect clustering for each set of references needs to be created manually. Once the ground truth is established, the difference between this and a given clustering may be measured by using a combination of information theoretic measures such as average entropy of the clusters and mutual information. However, this method of evaluation is not scalable. Given n references and up to k clusters, there are $k^n/k!$ possible clusterings for every value of k. Further, this has to be repeated for each of N papers. For a typical paper with 30 references and 4 clusters, this represents $4^{30}/4! > 10^{16}$ possible clusterings. We therefore adopted the following two alternative approaches: one is automatic and the other manual.

Automated evaluation The rationale is as follows. We considered the second clustering approach, that specified implicitly by the author(s) in citing multiple references together, as being the basis of the ideal clustering. In essence, an ideal clustering should have a 1:1 correspondence with that specified by the author(s), or should have fewer clusters with the constraint that some or all of the clusters are derived by fusion of the author-specified groupings. In other words, a clustering is considered ideal if it does not split the groups of references specified by the author(s). Another way of stating this is that grouped citations within the body of the document represent either the ideal clustering per se or a sub-clustering of the ideal clustering. Thus, the performance of a given clustering

technique depends on its ability to merge bracket clusters without shuffling them. Based on the above rationale, we evaluated each MCL clustering by calculating the distance of each clustering from the corresponding Locality clustering. The distance D is defined in the following equation.

$$D = \sum_{i=1}^{n} d_i \tag{7.3}$$

Where n is the number of clusters in Locality clustering of a given document, and d_i ($i = 1, , n$) is the corresponding weighted average entropy calculated as follows:

$$d_i = -(M_i/N_r) \sum_{j=1}^{k} (m_j/M_i) log(m_j/M_i) \tag{7.4}$$

Where N_r is the total number of references in the document, M_i is the number of references in the ith cluster of the locality clustering, $m_j (j = 1, , k)$ are the split fragments of the ith cluster which are scattered in a MCL clustering. The smaller the total distance D, the better the corresponding MCL clustering is. Based on the calculation of d_i we know that if the ith cluster is not broken, then $di = 0$. Otherwise, $di > 0$. For example, suppose the ith cluster of the locality clustering is "1, 2, 3" (which means these three references are mentioned together somewhere in the paper). For each MCL clustering, we check to see if this locality cluster was broken or not. For a given MCL clustering, if the locality cluster is broken into [(1) (2, 3)], and the total number of references is 20, then we have $d_i = -(3/20)[(1/3)log2(1/3) + (2/3)log2(2/3)] = 0.1377$. As locality clustering is author-defined, we assume it has 100% precision for this evaluation purpose. However, several of the clusters may be potentially fused with each other on the account of being semantically homogeneous. Thus, an ideal clustering might consist of a hierarchical clustering of the locality clusters. Figure 19 shows a plot of the distance from locality

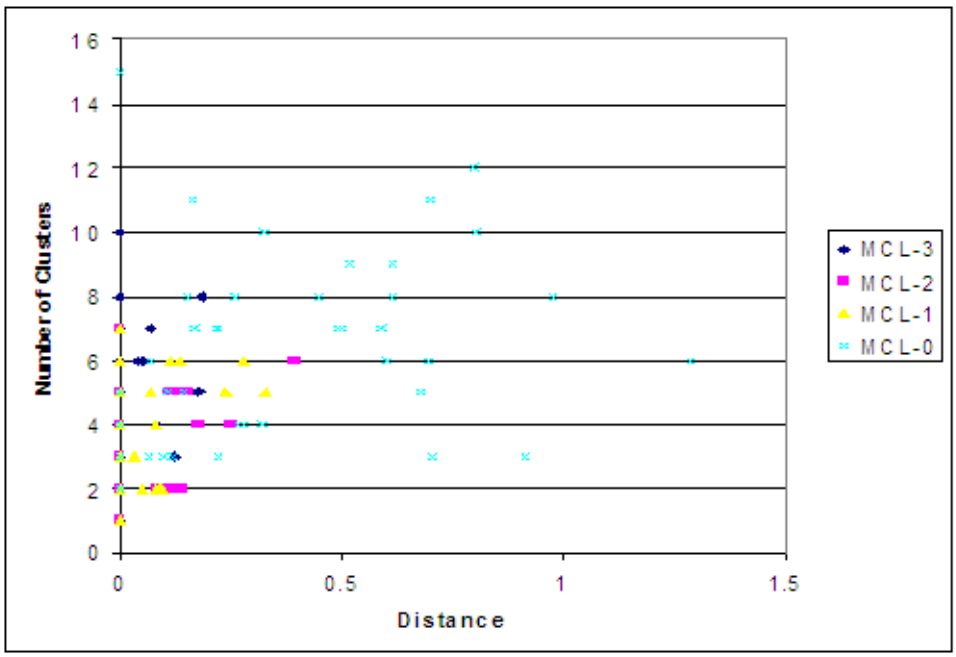

Figure 19: Distances from Different Approaches with MCL to Locality Clustering

clustering versus the number of clusters for each of the four MCL clustering approaches. As expected, the distance from locality clustering progressively decreases as we take more contextual information into account. Figure 20 shows that MCL-3 has the lowest distance on average. MCL-0 is the worst in being furthest away from the locality clustering, with a large variance as well. One limitation of the distance metric presented here is that it can essentially result in a distance for any hierarchical clustering of the locality clusters, as long as none of the original clusters are split. In the extreme case, a single giant cluster consisting of all the references would have a distance of zero from the locality clustering. To account for this, we also performed manual validation as described in the next section.

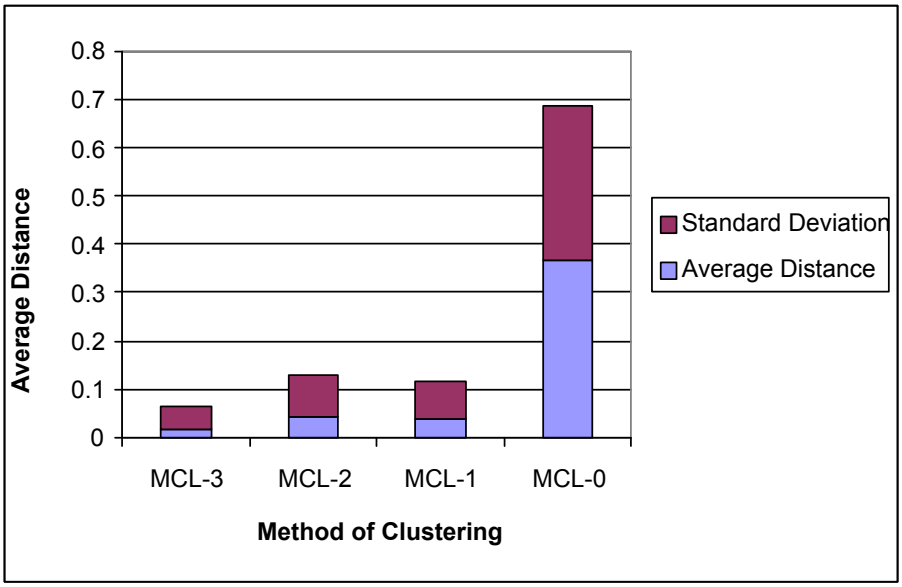

Figure 20: Average Distances from Different Approaches with MCL to Locality Clustering

Manual evaluation In addition to computing distances based on locality clustering for each MCL clustering, we also manually checked all the clusters to calculate a purity score for each clustering of each paper. The purity score P is calculated as follows.

$$P = \sum_{i=1}^{n} p_i \tag{7.5}$$

Where n is the number of clusters in an MCL clustering of one paper, and p_i is computed according to the following equation.

$$p_i = (M_i/N)(m_i/M_i) = m_i/N \tag{7.6}$$

N is the number of references in a paper, M_i is the number of references in the ith cluster, and m_i is the number of references that are considered acceptable for inclusion in the cluster. The higher the score P, the better the corresponding MCL clustering is.

In contrast to the distance metric used in the automatic evaluation above that is bounded only on one side, P is bounded between the values of 0 and 1. The highest score is 1, and a low score indicates a highly heterogeneous cluster. Figure 21 shows the distribution of purity scores for different clustering approaches. Mcl-3 clustering shows the best purity. However, MCL-0 shows similar purity to MCL-1 and MCL-2. Figure 22 shows how the number of clusters is reduced by the other approaches relative to locality clustering. Here too, MCL-3 exhibits high values of purity, while successfully condensing the citations into a smaller number of clusters.

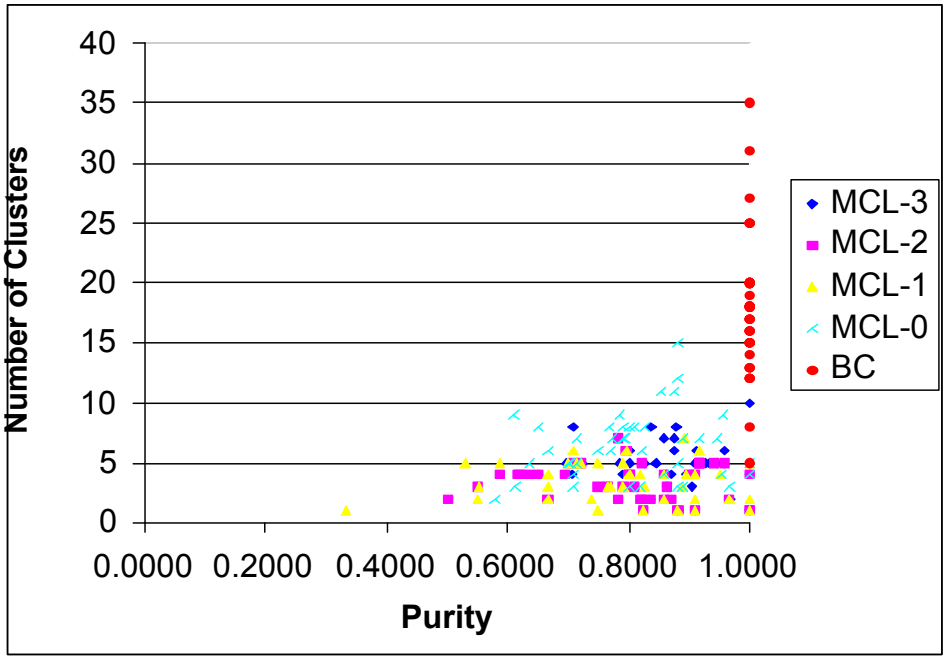

Figure 21: Purities of Different Approaches with MCL

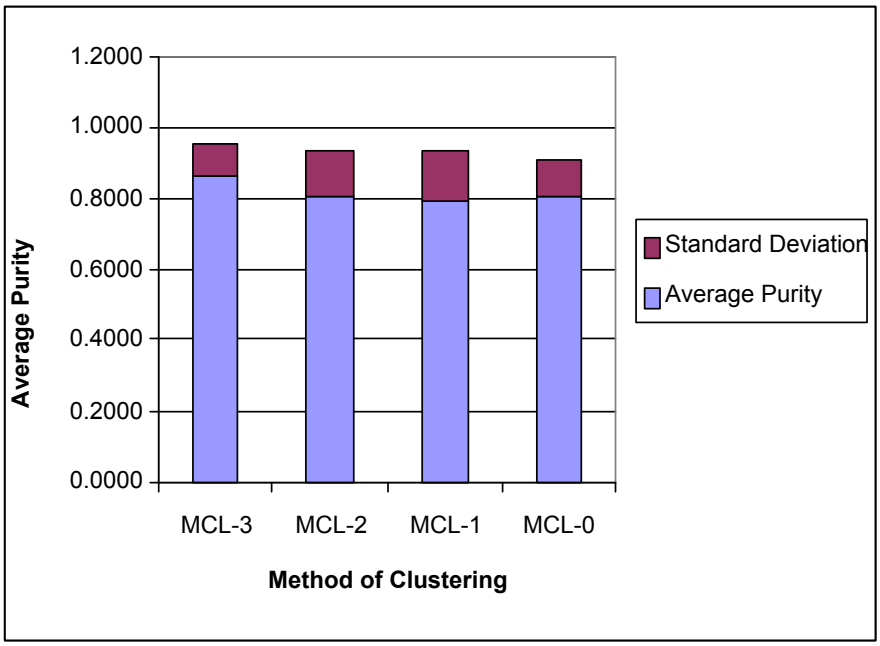

Figure 22: Average Purities of Different Approaches with MCL

7.1.7 Summary

We have presented and evaluated an automated approaches of reference clustering. Through our automatic and manual evaluations, approach 6 (MCL-3) brings us the best result. Besides being used in document clustering which is fully discussed in this dissertation, it can also be used as a summarization technique for scientific documents, for the identification of new terms used in a domain, or simply as a new way to order the citations in a publication.

7.2 Results from CS-VS

We downloaded articles in the biomedical domain from PubMed Central [25]. We chose twelve categories corresponding to topical journals as our original clusters as shown

in Table 6. We evaluated our results based on these original categories.

Table 6: The Document Categories

Category	Number of Documents
Behav Brain Funct	129
BMC Blood Disord	29
BMC Cardiovasc Disord	175
BMC Endocr Disord	38
BMC Neurol	161
BMC Oral Health	73
BMC Plant Biol	201
Cough	31
AIDS Res Ther	70
BMC Biochem	173
BMC Cancer	123
BMC Infect Dis	96

From these articles, we generated multiple document sets as training data by the random selection (Table 7). They are used to obtain appropriate weights for formulas 4.2 (or 4.3) and 4.5, namely, W_1, W_2, W_3, W_4, W_5. Note that there is no keyword provided in these articles and we set $W_6 = 0$. The document sets involved in each combination were used as the ground truth for evaluating our clustering results. The documents in Table 7 are only from the first eight categories of Table 6. The remaining four categories were held back to serve as noise to test the robustness of our approach. There are two testing sets – one only has documents from the same eight categories, the other has documents from all twelve original categories. To find these weights, we applied an evolution strategy to our training process. The results of using the evolution strategy will be discussed in next subsection.

To evaluate the quality of the clustering result of this approach, we use F-Measure [89], also known as F-Score, or the harmonic mean of the precision and recall, to calculate

Table 7: The Training Data

ID	Document Categories	Number of Documents
1	BMC Blood Disord, BMC Cardiovasc Disord	40
2	Behav Brain Funct, BMC Blood Disord	40
3	BMC Blood Disord, BMC Neurol, Cough	58
4	Behav Brain Funct, BMC Blood Disord, BMC Oral Health	60
5	BMC Neurol, BMC Oral Health	234
6	BMC Blood Disord, BMC Neurol, BMC Oral Health, Cough	78
7	Behav Brain Funct, BMC Blood Disord, BMC Neurol, BMC Oral Health, Cough	98
8	Behav Brain Funct, BMC Blood Disord, BMC Endocr Disord, BMC Oral Health, BMC Plant Biol, Cough	119
9	Behav Brain Func t, BMC Blood Disord, BMC Cardiovasc Disord, BMC Endocr Disord, BMC Oral Health, BMC Plant Biol, Cough	139
10	Behav Brain Funct, BMC Blood Disord, BMC Cardiovasc Disord, BMC Endocr Disord, BMC Neurol, BMC Oral Health, BMC Plant Biol, Cough,	158

the accuracy of the resulted clusters. It is defined as follows.

$$F = \frac{2P \cdot R}{P + R} \tag{7.7}$$

Where P and R are precision and recall which are defined in the following equations, respectively.

$$P = N_{cr}/N_{tr} \tag{7.8}$$

$$R = N_{cr}/N_{ct} \tag{7.9}$$

Where N_{cr} is the number of documents which are correctly returned, or they are put into the cluster they belong to (based on the original categories we downloaded); N_{tr} is the total number of documents in a cluster; N_{ct} is the the total number of correct documents a cluster is expected to have. That is, when evaluating the result of the clustering algorithm

over some documents which belong to some categories, we use equations 7.7 through 7.9, to calculate the precision, recall, and f-measure for each resulted cluster. Then we compute the average values which are considered as the accuracy or quality of the clustering over that set of documents.

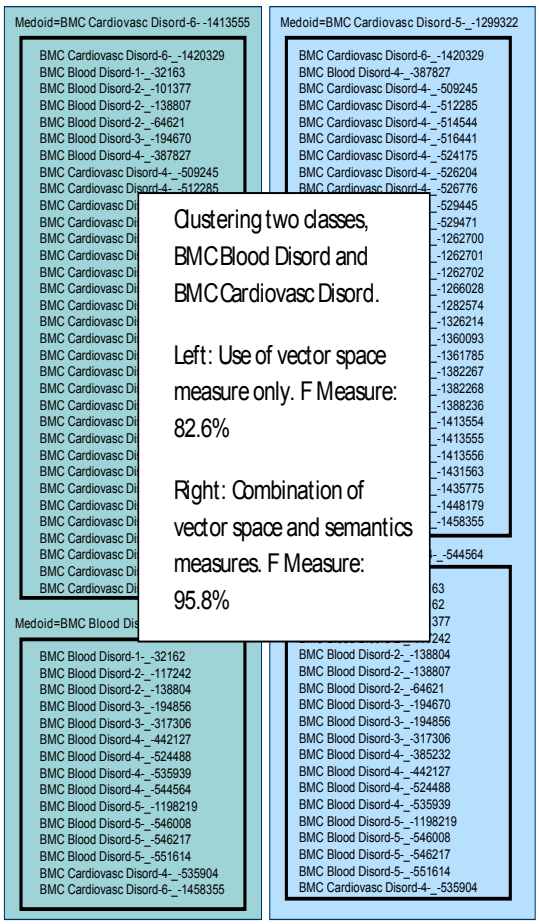

Figure 23: A Sample Result of Document Clustering with CS-VS

As an example, Figure 23 shows the results of both clustering with vector space measure only and clustering with combined vector space and semantics measures. On the

left hand side, for the resulted two clusters, for the cluster with medoid "BMC Cardiovasc Disord-6-_-1413555", the precision $P = 28/34 \approx 0.824$, the recall $R = 28/30 \approx 0.933$, and the F-Measure is $F = (2 * 0.824 * 0.933)/(0.824 + 0.933) \approx 0.875$. Similarly, we can get these three values for the cluster with medoid "BMC Blood Disord-4-_-385232" as 0.875, 0.7, and 0.778, respectively. Therefore, the (average) F-Measure of the result of the clustering using vector space measure only over these data set will be 82.6%. In the same way, we can get the (average) F-Measure of the result (on the right hand side) of the clustering using combined vector space and semantics measures over these data set as 95.8%.

7.2.1 Results of Using Evolution Strategy

To find the weights used in equations 4.2 (or 4.3) and 4.5, namely, W_1, W_2, W_3, W_4, W_5, we applied the evolution strategy in our training process. The detailed discussion of the evolution strategy is in Section 4.3 of Chapter 4. To apply the evolution strategy, we need to have data sets ready. We first used papers of eight categories to construct ten collections as our training data. Then we used papers from the same eight categories to constructed ten collections as our test data 1. Lastly, we used papers from all twelve categories to construct ten collections as test data 2.

We used two different ways to find these weights through the evolution strategy. 1. We tried to find all the five weights simultaneously, by doing clustering on training data sets combining vector space and semantics measures. Table 8 shows the results of this approach. 2. We find these weights through two stages. That means, we find W_3,

W_4, and W_5 first, by doing document clustering on training data sets using the semantic similarity (equation 4.5) only, then we do document clustering by combining vector space and semantics measure to find weights W_1, W_2. Table 9 shows the results of using this approach. The last two rows of both tables are the average and standard deviation.

Table 8: Results of Evolution Strategy - Get All Weights Simultaneously

# G	W3	W4	W5	W1	W2	train	test1	test2
56	2.898	2.414	1.726	4.777	0.152	0.863	0.809	0.767
71	0.915	1.036	0.000	13.146	1.459	0.866	0.800	0.794
68	0.436	0.900	0.338	2.452	0.129	0.851	0.762	0.782
19	0.747	3.983	0.000	5.418	0.888	0.852	0.772	0.793
95	0.994	4.841	2.454	10.575	1.142	0.866	0.790	0.814
60	0.351	1.184	0.444	3.683	1.269	0.862	0.803	0.806
8	0.716	2.449	1.446	2.989	0.208	0.857	0.780	0.837
6	0.112	0.692	0.206	2.726	2.407	0.864	0.788	0.828
17	0.805	1.373	0.000	1.661	0.487	0.851	0.765	0.797
21	1.371	3.957	0.009	3.156	0.040	0.861	0.757	0.762
42.1	0.935	2.283	0.662	5.058	0.818	0.859	0.782	0.798
31.4	0.776	1.502	0.886	3.795	0.764	0.006	0.018	0.024

Table 9: Results of Evolution Strategy - Get Weights Separately

# G1	#G2	W3	W4	W5	W1	W2	train1	train2	test1	test2
10	39	0.203	3.205	0.000	4.540	0.574	0.629	0.861	0.787	0.781
5	54	0.534	5.006	0.782	4.707	0.396	0.605	0.857	0.797	0.792
4	100	0.000	0.776	0.705	3.153	0.000	0.604	0.758	0.719	0.739
8	14	0.658	3.072	0.194	6.326	1.162	0.621	0.855	0.782	0.794
24	4	0.692	2.497	0.086	1.706	0.443	0.602	0.855	0.779	0.790
3	11	0.859	4.509	1.798	2.051	0.113	0.618	0.859	0.780	0.834
11	27	0.939	9.636	3.526	8.697	0.446	0.613	0.859	0.781	0.824
3	19	0.875	4.168	2.330	6.436	0.316	0.616	0.859	0.779	0.812
13	7	0.901	4.507	4.979	3.016	0.383	0.600	0.868	0.747	0.780
14	100	0.000	0.430	0.406	1.770	0.000	0.604	0.758	0.719	0.739
9.5	37.5	0.566	3.781	1.481	4.240	0.383	0.611	0.839	0.767	0.789
6.5	36.3	0.370	2.576	1.676	2.341	0.337	0.009	0.043	0.028	0.031

In both approaches, once we get promising values for these five weights, we use these weights to do clustering on two test sets. Results are also shown in both tables. In

using evolution strategy, we set 100 as the threshold of the number of generations, 85% of the average F-Measure as the stop criteria of document clustering by combining vector space and semantics measure in both approaches, and 60% as the stop criteria of the first stage in the second approach. That means, in both approaches, the training process will stop when either the number of generations reaches 100, or the average F-Measure of clustering reaches 85%. For the second approach, the first stage will stop when either the number of generations reaches 100, or the average F-Measure of clustering reaches 60%.

In each approach, we obtained ten combinations of these five weights. Overall, compared with F-Measure 71.9% and 73.9%, when doing clustering on these two test sets using vector space only, these twenty combinations found through both approaches can improve F-Measure by 5% on both test sets. These performances are consistent, which is evidenced through standard deviations of all the F-Measures (maximum is 0.031 on test data sets). However, there are some differences between these two approaches. First, all ten combinations in the first approach resulted in more than 85% (75.9% when using vector space measure only) on training data within 100 generations, whereas two combinations in the second approach did not reach 85% when evolution process stopped after competing 100 generations. Secondly, the first approach resulted in 78.2% and 79.8% of average F-Measure on the two test sets respectively. These numbers are a little higher than 76.7% and 78.9% obtained through the second approach. Thirdly, the average number of generations was 42.1 in first approach which was less than 47 (9.5+37.5) in the second approach. From these comparisons we may conclude that the first approach is a little better than the second one. However, looking at the second table carefully we found

something interesting.

We noticed, in the second approach, two evolution strategy processes did not reach expected F-Measure 85% on training data. Interestingly, in both combinations, the weights W_3 and W_2 are 0's. This means, when finding the first three weights using semantics measure only, W_3 was assigned 0 and the F-Measure still reached 60% within 100 generations (4 and 14 respectively). But, when finding W_1 and W_2, with W_2 also occasionally being assigned 0, the F-Measure never reached 85% within 100 generations. This may suggest that even though title is not significant in clustering papers (the average F-Measure is 29.9% when doing clustering using title only), it is important. To avoid this situation we can mandate W_3 to be bigger than 0 when finding these three weights. If we remove these two exceptions from Table 2, we will get an average F-Measure of 77.9% and 80.1% which are almost the same as that in the first approach with a lower average number of generations 31.5 (9.6+21.9).

In conclusion, both approaches are consistent and comparable in finding weights. And weights found in both approaches can indeed improve F-Measure of clustering.

The reader may have noticed that in all the combinations of these weights, we always have $W_1 > W_2$. That means the evolutions strategy assigned more weight to vector space measure than to semantics measure. There are two possible reasons behind this. First, is that the vector space vectors which use the entire documents, include more complete information than semantics extracted from titles, references, and co-citation, which are part of the document. This can be seen through Table 10. It shows the F-Measures of the results of clustering using single measures over the training data. The

Table 10: F-Measures of Clustering Using Single Measures

vector space	75.9%
semantics	63.8%
titles	29.9%
co-citations	35.8%
citation semantics	59.3%

F-Measure of using vector space measure only is 75.9%, in contrast to 63.8% of using semantics measure only. Another reason is that W_5 has been assigned a value about 5 in most cases, which implies a higher value for the semantics measure compared to the vector space measure which is normally low with TF-IDF. Therefore, assigning more weight to vector space measure compensates for this difference and hence balances these two measures, which leads to the higher quality of clustering.

7.2.2 Combining Vector Space and Semantics Measure

In this approach, we actually have two ways to combine vector space and semantics measure as shown in equations 4.2 and 4.3. The former one is the harmonic mean of these two measure (as in F-Measure which is the harmonic mean of precision and recall), the latter is the simple addition of them. Since the former combination balances these two measures, we expect a better result by using it. Our experimental results conformed this hypothesis. Figure 24 shows some of the comparisons, where "F-H", "P-H", "R-H" are the F-Measure, precision, and recall, respectively, of using the harmonic mean of vector space and semantic measures, the other three are for simple addition of these two measures. On the x-axis, the labels are the combinations of the five weights

"$W_1 : W_2 : W_3 : W_4 : W_5$" used in equations 4.2 and 4.3. Overall, the results of using harmonic mean is slightly better than using simple addition with about a 2% edge considering F-Measure. So, if not specified, in our experiments we used equation 4.2 to combine these two measures.

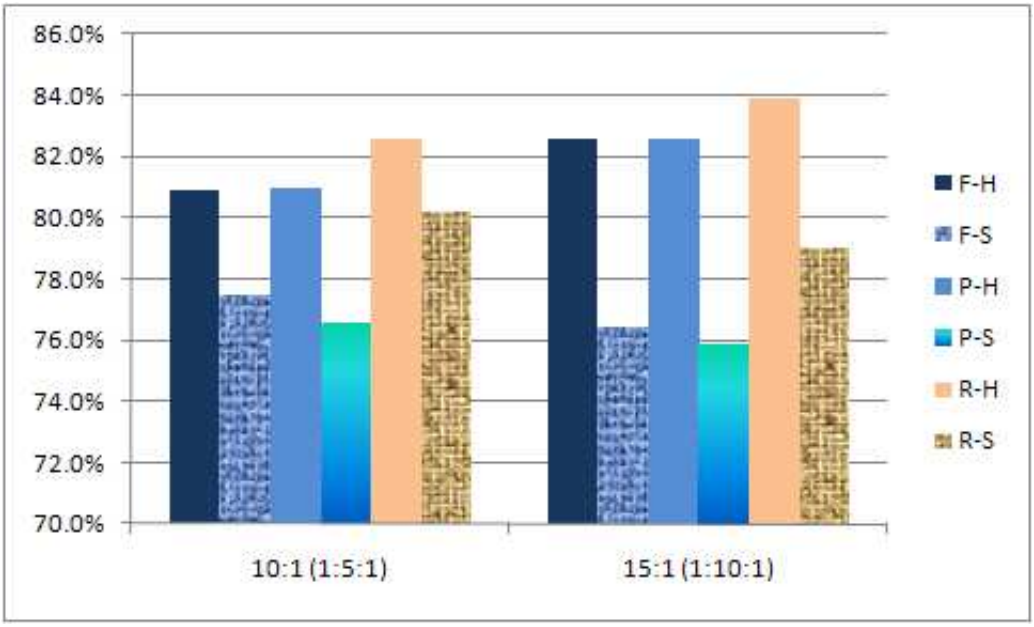

Figure 24: Comparison of Results Using Harmonic Mean and Simple Addition

7.2.3 Results of Using CS-VS on Physics Documents

To test the consistency of the performance of our approach applied to different domains, we downloaded some physics papers from Nature Physics Portal [20]. We selected nine sub-topics from the available collections. Their names along with abbreviations are as follows: Astrophysics (AP), Atomic and molecular physics (AMP), Biological physics (BP), Chemical physics (CP), Condensed-matter physics (CMP), Materials physics (MP),

Nanotechnology physics (NP), Optical physics (OP), and Quantum physics (QP).

From each category, we downloaded around 50 of the most recent papers. Out of these papers, we created training data set, test data set 1, and test data set 2. We used papers from the first seven categories as training data and test data set 1. Then, we added papers from the other two categories (Optical and Quantum physics) as noise to create test data set 2. Training data and test set 1 each consisted of six collections with a number (k) of categories k=2, 3, 4, 5, 6, and 7 respectively. Test data set 2 consists of eight collections with k=2, 3, 4, 5, 6, 7, 8, and 9, respectively. More detailed information of these three data sets and F-Measures of clustering results are shown in Table 11 and Figure 25.

Table 11: Results of Clustering on Physics Documents

Data sets	**Categories**	**Total Number of Documents**	**F-Measure of VS only (%)**	**F-Measure of VS +Semantics (%)**
Training	AP, AMP	57	72.2	79.2
	AP, AMP, MP	75	43.6	74.7
	AP, AMP, CP, MP	104	47.3	49.3
	AP, AMP, CP, CMP, MP	133	33.1	34.4
	AP, AMP, CP, CMP, MP, NP	130	37.6	34.5
	AP, AMP, BP, CP, CMP, MP, NP	128	40.7	36.5
	Average ->		45.7	51.4
Test 1	AP, AMP	48	36.8	62.5
	AP, AMP, MP	53	75.5	77.4
	AP, AMP, MP, NP	73	45.3	53.4
	AP, AMP, CMP, MP, NP	91	39.6	41.0
	AP, AMP, BP, CMP, MP, NP	86	40.8	35.5
	AP, AMP, BP, CP, CMP, MP, NP	145	29.4	28.7
	Average F-Measure		44.6	49.8
Test 2	OP, QP	45	34.3	51.5
	AP, AMP, QP	73	47.2	48.2
	AP, AMP, MP, QP	63	51.5	53.2
	AP, AMP, MP, NP, QP	86	41.8	49.4
	AP, AMP, CMP, MP, NP, QP	97	40.5	42.5
	AP, AMP, BP, CMP, MP, NP, QP	89	45.1	39.5
	AP, AMP, CP, CMP, MP, NP, OP, QP	117	31.3	37.8
	AP, AMP, BP, CP, CMP, MP, NP, OP, QP	192	27.9	35.1
	Average ->		40.0	44.7

Using Evolutionary Strategy, we obtained a weight combination of "$W_1 : W_2(W_3 : W_4 : W_5) = 12.419 : 1.094(5.580 : 7.296 : 3.778)$" (There is no keyword information in

Figure 25: Average F-Measures of Clustering on Physics Documents

this physics collection either), which improved the accuracy of clustering by 5.7% (from 45.7% to 51.4%)over the training data in terms of F-Measure. Worth mentioning, is that F-Measures of clustering using title only, citation semantics only, and co-citation only, are 22.7%, 40.7%, and 21.9%, respectively. They were all lower than clustering over the biomedical documents which are shown in Table 10. However, there is one thing in common, the result of using citation semantics is the best among these three semantic elements.

Using these weights we did clustering on test sets, we also got 5.2% (from 44.6% to 49.8%) improvement compared to that of using vector space measure only on test set 1, and 4.7% improvement on test set 2. These overall results were not as good as the results as we got from biomedical data sets where in many cases the improvements are over 10%. Nevertheless, the performance of our approach is consistent. In most cases, it is much better than using vector space only to do clustering.

Comparing these results and closely examining references in the documents of both domains, we can tell that the quality or clarity of references in physics data sets is not as good as that of Biomedical data sets. Also, granularity of categories in Physics data sets varies more than that in Biomedical data sets. For example, atomic and molecular physics is closer to chemical physics than to astrophysics. These two reasons may make the citation semantics less significant than that in Biomedical data sets.

As a byproduct, the test on physics documents shines the light on another potential use of our approach - to reveal or measure the quality of references in a collection of documents. That is, on the one hand, the semantics measure can help improve the quality of document clustering. On the other hand, the magnitude of the improvement of our approach reveals the quality of the references used in the document collection.

7.3 Results from CS2CS

7.3.1 Comparing CS2CS with Other Approaches

In the experiments of this approach, we used the documents from the same eight biomedical categories as used in the training set of CS-VS. We also used other categories to test out splitting and merging algorithms. We first did experiments using different document clustering algorithms to compare their performance. Table 12 and Figure 26 show the detailed results of using K-Means clustering (K-Means), Bisecting K-Means clsutering (Bisecting K-Means) K-Medoids clustering with vector space similarity measure (K-Medoids(VS)), K-Medoids clustering with combined vector space and semantics measure (CS-VS), linear clustering using feature selection only from vector space vectors

(FV (VS)), and CS2CS model based linear clustering using the 3-level feature selection (CS2CS), to cluster 567 of these 725 documents that are from those eight classes mentioned previously. For the last two clustering algorithms, the other 158 documents that also belong to these eight classes were used as training data. In this section, if not specified, all the values of the F-Measures, precisions, and recalls are the average values of at least five runs on the data collection with the same size but different documents. The

Table 12: Comparison of Results of Different Clustering Algorithms

Algorithm	F-Measure (%)	Precision (%)	Recall (%)	FV_Length1	FV_Length2	Runtime (Seconds)
K-Means	40.1	44	39.3	N/A	N/A	301
Bisecting K-Means	42.4	45.1	41.7	N/A	N/A	325
K-Medoids(VS)	50.7	50.2	54	N/A	N/A	668
CS-VS	55.9	56.3	55.1	N/A	N/A	1219
FV(VS)	59.3	66.4	64.6	10~100	10~100	239
CS2CS	61.9	61.7	72	10~100	10~100	254

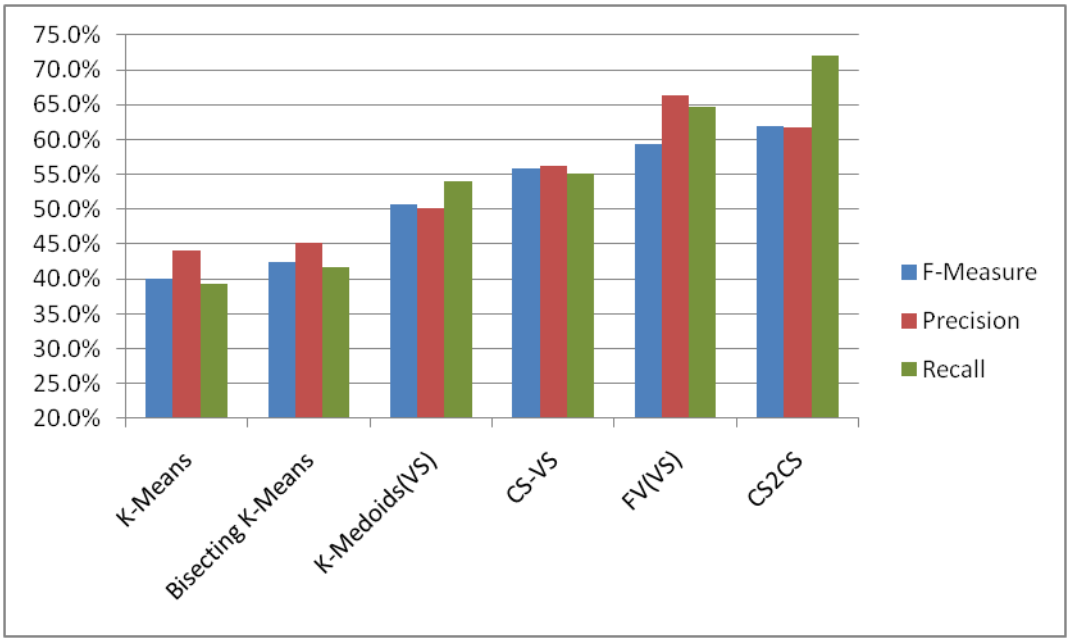

Figure 26: Comparison of Results of Different Clustering Algorithms

original weights "$W_1 : W_2(W_3 : W_4 : W_5)$" are "$10 : 1(1 : 5 : 1)$" that were obtained through the evolution strategy during the training process. These weights are for similarity measure when doing K-Medoids clustering with combined vector space and semantics measures. In doing CS2CS clustering, we do not need these two similarity measures. However, as shown in equation 5.1,these weights (except for "W_6") are used to calculate the weight of each term of a feature vector. Since we adjusted the weights of terms in the vector space by dividing the average TF-IDF weight (smaller than 1), we change the weight for vector space W_1 to 1 accordingly.

FV_Length1 is the length of the feature vector of each single document within an existing cluster. In other words, it is the number of top terms used to form the feature vector of a single document. These feature vectors are used to form the feature vector of the cluster they belong to. FV_Length2 is the length of the feature vector of a new document. This feature vector is used to compare with the feature vectors of existing clusters to decide where the new document goes. Once the new document is put into a cluster, the feature vector with FV_Length1 (not FV_Length2) of this new document will be used to update the feature vector of that cluster. A reasonable estimation of both length are in the range of 10 and 100. If the length is less than 10, we lose too much useful information; if it is bigger than 100, more noise will be included. In either case, the resulting clustering had a lower quality. The last two rows of Table 12 show the average F-Measure, precision, and recall of approaches FV (VS) and CS2CS with the lengths in this range.

Runtime is the time used to cluster these 567 documents that belonged to eight

classes. It did not include the runtime used in the training process in the cases of CS2CS and CS-VS. The training process could be skipped if we set the weights (e.g. “1:1:1:5:1” in this case) heuristically. Or, even if we need the training process, we could use a small training data set (which results in a fixed small training time) without affecting much of the clustering quality, since the feature vectors of clusters evolve as they grow. Therefore the training time is ignorable, should there be a large number of new documents to be clustered.

In this table, we can see CS2CS is better than any others regarding both accuracy and runtime. FV (VS) uses a similar procedure as CS2CS, but only uses vector space vectors to form feature vectors for documents. For FV (VS), the average F-Measure is 59.3%, whereas it is 64.8% in the case of CS2CS. It clearly demonstrates the importance of considering semantic elements in clustering. Another noteworthy point is that the result of FV (VS) is better than that of any other algorithms except for CS2CS. This shows that our strategy of forming feature vectors and normalizing feature vectors effectively retrieved important information and excluded noise in the same time.

7.3.2 Results of Automatically Finding FV_Length2

As discussed in Section 5.6, instead of setting FV_Length2 explicitly, we can search for the best value in real time. Table 13, Figure 27, and Figure 28 show the results of CS2CS using different strategies to search for the length of new documents: brute force search, and two sampling search algorithms, namely, linear increment search and

exponential increment search. The two graphs show the comparison of the average F-Measure, precision, recall, and runtime, respectively. FV_Length1 is the length of the

Table 13: Results of CS2CS with Automatic Finding the Lengths of Document Feature Vectors

FV_Length1	Brute Force Search		Linear Increment Search		Exponential Increment Search	
	F-Measure (%)	Runtime (Seconds)	F-Measure (%)	Runtime (Seconds)	F-Measure (%)	Runtime (Seconds)
10	52.6	440	51.9	257	51.3	233
20	63.4	619	63.3	308	65.4	261
50	66.0	1099	68.6	400	66.5	332
70	67.0	1360	66.2	468	66.7	374
100	73.1	1748	72.8	559	71.1	437
Average ->	64.4	1053	64.5	398	64.2	327

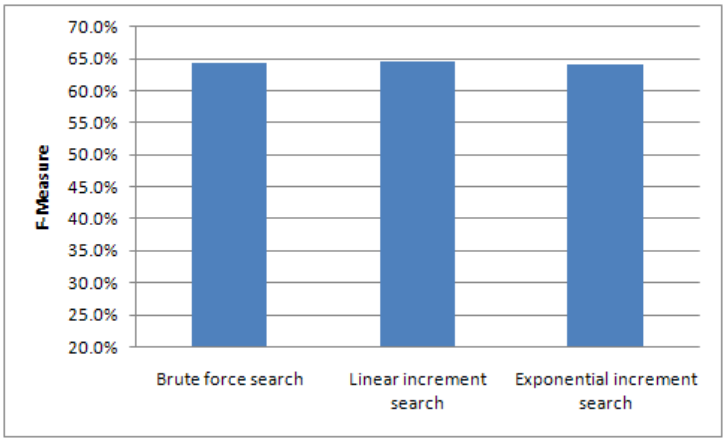

Figure 27: Results of CS2CS with Automatic Finding the Lengths of Document Feature Vectors

feature vector of any existing document used to form the feature vector of the cluster it belong to. From this table, one can tell that the difference among the F-Measures of using these strategies is trivial. But the Brute Force Search takes much longer time than the other two do. Considering the tradeoff between the F-Measure and the runtime, the exponential increment search is the best one. The following are two examples of which

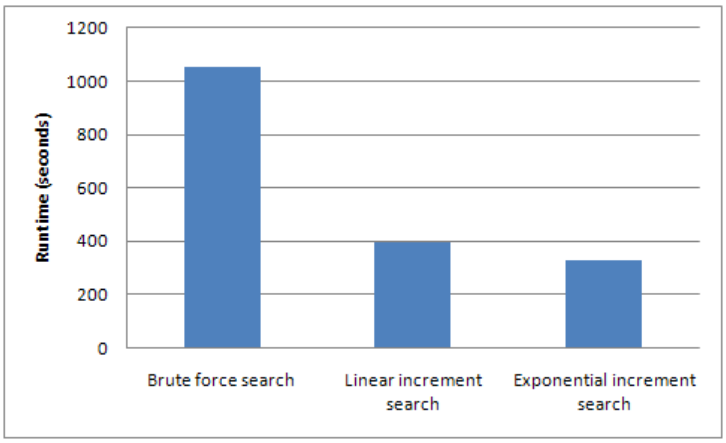

Figure 28: Runtime of CS2CS with Automatic the Lengths of Document Feature Vectors

lengths have been checked between 10 and 100 by using Linear Increment Sampling and Exponential Sampling. In both examples, a fewer number of lengths has been checked

Table 14: Examples of Lengths Checked by These Two Sampling Search

Exponential Increment Search	**Linear Increment Search**
10, 11, **13**, 17, 25, 41, 73	10, 11, **13**, 16, 20, 25, 31, 38, 46, 55, 65, 76, 88
10, 11, 12, 14, 18, **19**, 20, 22, 26, 34, 50, 82	10, 11, 12, 14, 17, **21**, 22, 24, 27, 31, 36, 42, 49, 57, 66, 76, 87, 99

using exponential increment search. Also in the first example, they both found the same best length as 13; while in the second one, the best lengths they found are a little different (19 vs. 21). On average, exponential increment search will check a fewer number of lengths than linear increment search (8 vs. 14.5).

From the results and analysis above we can see, even though a range of lengths of the feature vector of a new document still need to be set heuristically, the range could be very large. This is because using exponential increment sampling we can quickly find a best length within even a very large range. While the runtime is still comparable with that

of using a length manually set, and the F-Measures are consistently higher.

7.3.3 Forming Feature Vector with the Aid of MeSH

To take advantage of MeSH ([18]), we adjusted weights of the terms found in MeSH terms. That is, we increase the weights of terms found in MeSH since MeSH terms are considered as important terms in Biomedical areas. In each feature vector, if a word is found in any MeSH term, we adjust its weight by doubling it. Table 15 shows the results of using MeSH compared to results of not using MeSH. Both use exponential increment search to find the best length of the feature vector of a new document.

Table 15: Results of CS2CS with MeSH and without MeSH

FV_Length1	Without MeSH			With MeSH		
	F-Measure (%)	Precision (%)	Recall (%)	F-Measure (%)	Precision (%)	Recall (%)
10	51.3	51.8	64.8	51.3	50.8	62.2
20	65.4	64.1	74.6	62.4	61.7	73.1
50	66.5	65.1	80.3	72.7	70.0	80.0
70	66.7	65.7	80.8	72.9	69.5	82.7
100	71.1	68.1	82.3	79.5	77.4	83.6
150	73.4	72.7	80.9	77.0	77.1	81.7
Average ->	65.7	64.5	77.3	69.3	67.8	77.2

While the recalls of these two results are almost the same, the F-Measure and precision did increase by using MeSH. This is because MeSH terms are relevant or significant terms in biomedical domain, by assigning more weights to these terms in clustering biomedical documents, we expected to get clusters with higher precisions and hence higher F-Measures even though the recalls may remain the same. From this table, we also see that with FV_Length1=100, we have the highest increment of F-Measures (71.1 vs. 79.5). That means, when we using 100 as the length of the feature vector of existing documents to form the feature vector of the clusters they belong to, we get the best tradeoff

between keeping enough useful information and eliminating noise.

We also investigated the case of multi-word terms with partially or exactly matching MeSH terms. Table 16 is the example of using terms with up to five words. As expected,CS2CS clustering with multi-word terms with partial match takes much longer time than exact match. Since with exact match, we can use a hash table to store MeSH terms, and the search will just take a constant time. But for partial match, we need look at every MeSH term to find the best match, in other words, we need to find the highest percentage of match. For the exact match, we double the weighs of matched terms. For partial match, we multiply the weight of a term by $1+p$, where p is the highest percentage of the match between the term and some MeSH term. Surprisingly, the average F-Measure of the partial match is almost the same as that of exact match even with the high cost of runtime. This is because, by increasing weights of terms with partial match to MeSH terms, we somehow give more weights to some noise terms. However, the average difference between their precisions and recalls are not surprising. With exact MeSH match, we get a little higher precision, while with partial match, we have a little higher recall. Another observation on this table is that, it seems the F-Measure steadily increase with the FV_Length1 except "20". Actually, as we mentioned before, with FV_Lenght1 increase, more noise terms will be included in the feature vector of each cluster. So the F-Measure will go down at certain point. We did try length "200", and got 71.9%, 69.4%, and 82.5% for F-Measure, precision, and recall, respectively. Table 17 shows an example of the words of a feature vector with length 20. It is the document feature vector of document "Assessment of the role of transcript for GATA-4 as a marker of unfavorable outcome in

Table 16: Results of CS2CS Using Multi-word Terms Partially or Exactly Matching MeSH

FV_Length1	Partial Match			Exact Match		
	F-Measure (%)	Precision (%)	Recall (%)	F-Measure (%)	Precision (%)	Recall (%)
10	66.5	64.6	77.9	65.7	66.6	72.2
20	62.8	61.8	75.7	67.3	66.2	76.2
50	69.2	67.2	81.1	71.3	69.3	81.7
70	70.8	69.2	82.8	70.7	68.9	80.9
100	74.7	71.6	85.2	73	71.7	81.3
150	77.3	75	85.8	73.7	72.3	82.9
Average ->	70.2	68.2	81.4	70.3	69.2	79.2

Table 17: Words of a Document Feature Vector Mapped to MeSH Terms

Word	MeSH ID	MeSH Term
carcinoma	A11.251.860.590	Embryonal Carcinoma Stem Cells
marker	D12.644.360.543	01factory Marker Protein
fate		
db		
trigger	C05.651.869.870.800.800	Trigger Finger Disorder
tumor	A11.251.210.190	Cell Line, Tumor
cell	A03.556.124.369.320	Goblet Cells
transgene	B01.050.050.680.136.500	Mice, Transgenic
mutat	E05.393.760.700.300	DNA Mutational Analysis
optic	A08.800.800.120.680	Optic Nerve
rt		
promote	G02.111.570.080.689.675	Promoter Regions, Genetic
malignant	C02.256.466.606	Malignant Catarrh
conserve	D27.505.696.242	Bone Density Conservation Agents
pediatric	H02.163.700	Pediatric Dentistry
transcript	D08.811.913.050.134.440	p300-CBP Transcription Factors
predict	E01.370.378.530.775	Ovulation Prediction
leydig	A05.360.444.849.513	Leydig Cells
bromide	D01.139.300.050	Bromides
mice	B01.050.050.157.040.500	Mice, Congenic

human adrenocortical neoplasms Barbosa Angela", which belongs to the category "BMC Endocr Disord" ("BMC Endocrine Disorders"). They have been sorted according to their weights. And their weights have been adjusted considering MeSH. We can see, among these 20 words, 17 were found in some MeSH terms.

Table 18: Words of a Cluster Feature Vector Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
reinhardtii	0.0638	B01.040.080.925.344.650	Chlamydomonas reinhardtii	Eukaryota
polyp	0.0638	C04.557.470.035.215	Adenomatous Polyps	Neoplasms
polyphosphate	0.0638	D01.248.497.158.730.650	Polyphosphates	Inorganic Chemicals
saito	0.0638			
ecppxc	0.0638			
pbp	0.0638			
exopolyphosphatase	0.0638			
ssp	0.0631			
membership	0.0631	N04.452.122	Committee Membership	Health Services Administration
mtic	0.0631			
mediterranean	0.0631	C16.320.382.625	Familial Mediterranean Fever	Congenital, Hereditary, and Neonatal Diseases and Abnormalities
hereafter	0.0631			
arabia	0.0623	Z01.252.245.500.750	Saudi Arabia	Geographic Locations
xerostomia	0.0623	C07.465.815.929	Xerostomia	Stomatognathic Diseases
dryness	0.0623			
farsi	0.0623			
bardow	0.0623			
vdp	0.0615			
debt	0.0615			
longterm	0.0615			
vocation	0.0615	E02.831.782	Rehabilitation, Vocational	Therapeutics
opportune	0.0615	I01.409.137.500.996	United States Office of Economic Opportunity	Social Sciences
gallagher	0.0615			
aapd	0.0607			
smokeless	0.0607	B01.650.388.100.905.900.874	Tobacco, Smokeless	Eukaryota
gansky	0.0607			
cate	0.0599			
fluorapatite	0.0599			
inhomogenity	0.0599			
gaengler	0.0599			

Table 19: Words of a Cluster Feature Vector Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
polyphosphate	0.0786	D01.248.497.158.730.650	Polyphosphates	Inorganic Chemicals
orthophosphate	0.0786	D08.811.913.696.645.700	Pyruvate, Orthophosphate Dikinase	Enzymes and Coenzymes
ecppxc	0.0786			
polyp	0.0786	C04.557.470.035.215	Adenomatous Polyps	Neoplasms
arabia	0.0776	Z01.252.245.500.750	Saudi Arabia	Geographic Locations
xerostomia	0.0776	C07.465.815.929	Xerostomia	Stomatognathic Diseases
dryness	0.0776			
vocation	0.0765	E02.831.782	Rehabilitation, Vocational	Therapeutics
gallagher	0.0765			
vdp	0.0765			
smokeless	0.0754	B01.650.388.100.905.900.874	Tobacco, Smokeless	Eukaryota
porosity	0.0744	G01.374.710	Porosity	Physical Phenomena
fluorapatite	0.0744			
inhabit	0.0733			
employee	0.0733	N01.824.417.510.300	Employee Retirement Income Security Act	Population Characteristics
nicola	0.0733			
farmer	0.0733	C08.381.483.125.365	Farmer's Lung	Respiratory Tract Diseases
clermont	0.0733			
workforce	0.0726			
career	0.0726	F02.463.785.373.346.400	Career Choice	Psychological Phenomena and Processes
obliterated	0.0722			
traumatol	0.0722			
periapical	0.0722	A14.549.167.646.700	Periapical Tissue	Stomatognathic System
jacobsen	0.0722	C15.378.140.855.440	Jacobsen Distal 11q Deletion Syndrome	Hemic and Lymphatic Diseases
metamorphosis	0.0722	G07.700.320.500.550	Metamorphosis, Biological	Physiological Phenomena
andreasen	0.0722			
calcified	0.0722	C04.182.089.530.690.605	Odontogenic Cyst, Calcifying	Neoplasms
sequela	0.0722			
discoloured	0.0722			
subluxation	0.0722	C11.510.598	Lens Subluxation	Eye Diseases

Tables 18 and 19 show the top 30 terms (single words) of the two cluster feature

vectors of cluster “BMC Oral Health”. If a term match a word in a MeSH term, it is followed by the corresponding MeSH id and MeSH term, as well as the root term, that is the root category the MeSH term belongs to. Table 18 is the result without considering MeSH when forming document feature vectors. Table 19 is the result considering MeSH when forming document feature vectors. In particular, the weight of that term is doubled if it matches a word of a MeSH term. These are the cases that the length of the feature vector of each document is 100, high level weights are 1:1 (1:1:1). Even though you are not an expert in biomedical domain, you can find the positive effect by using MeSH. More MeSH terms were brought up to the top 30 (16 versus 10). Of course, there are still many terms which are not mapped to MeSH terms. This is because that, even though they are not MeSH terms (yet), they are important to this particular cluster (or category) based on the data from this collection. As an interesting example, using MeSH, the feature selection process brought “smokeless” (part of MeSH term “Tobacco, Smokeless”, with ID “B01.650.388.100.905.900.874”) from the 25^{th} position to the 11^{st} position in the cluster feature vector.

Another point we want to mention here is, as we pointed out before, on one hand, using ontologies can help improve document clustering; on the other hand, document clustering can help update ontologies in the sense that it can find new significant terms in a domain or particular categories (subdomains). For example, the terms “*dryness*” and “*ecppxc*”(Escherichia coli exopolyphosphatase, a protein) are in both tables. But neither is a MeSh term. However, based on our results, they could be added to MeSH, especially if the category “Oral Health” is included in MeSH in the future.

Table 20 shows the number of documents of each cluster in our experiments and the actual length of each cluster feature vector with and without considering MeSH in forming document feature vectors. The top 30 terms of the other cluster feature vectors and their mapping to MeSH terms are listed in the appendix of this dissertation.

Table 20: Lengths of Cluster Feature Vectors

Cluster Name	Number of Documents in Cluster		Length of Cluster Feature Vector	
	With MeSH	Without MeSH	With MeSH	Without MeSH
Behav Brain Funct	126	124	2715	3266
BMC Blood Disord	57	65	2175	2710
BMC Cardiovasc Disord	108	96	2585	2935
BMC Endocr Disord	55	47	2151	2247
BMC Neurol	95	100	2592	3085
BMC Oral Health	74	80	2220	2723
BMC Plant Biol	175	174	3267	4030
Cough	35	39	1542	1815

7.3.4 High-level Weights

As mentioned in Subsection 7.3.1, the high-level weights (weights for different parts of a document) were set to "$W_1 : W_2(W_3 : W_4 : W_5) = 1 : 1(1 : 5 : 1)$" based on the training process used in CS-VS. In this subsection, we want to show that we still can get good results without this training process. In other words, we just use the data set in training process as starting set, and use uniform weights (let them be 1:1 (1:1:1)) to get initial feature vectors for the starting clusters. If the result is comparable with that using the weights obtained from training process, we can eliminate the training process. Table 21 shows the results using different weights. In this experiment, we used single-word terms, and the weights of terms matched exactly with MeSH terms were doubled. FV_Length1 is the length of the feature vector of each document that is used to form the feature vector of the cluster it belongs to.

From this table we can see, the average F-Measure of using 1:1 (1:1:1) is even a little better than using 1:1 (1:5:1) which is obtained through training process. Of course it does not mean that the less we assign the weight to citation semantics, the higher the F-measure will be. We have shown at the beginning of this section that without citation semantics, the F-Measure is usually lower than using citation semantics. Furthermore, from this table, we can see, that the highest F-Measure 76.2% happens when weights are 1:1 (1:5:1). However, the results of using these two different weight sets are comparable. Also we notice that the F-Measure is not so sensitive to FV_Length1 when using 1:1 (1:1:1). That is a merit we want since FV_Length1 has to be set heuristically.

Table 21: Results of Using Different Weights

	1:1 (1:1:1)			**1:1 (1:5:1)**		
FV-Length1	**F-Measure (%)**	**Precision (%)**	**Recall (%)**	**F-Measure (%)**	**Precision (%)**	**Recall (%)**
10	68.9	67.3	78.1	63	62.6	75.7
20	66.6	65.4	75.9	68.2	65.7	80.7
50	71.2	69.8	81.3	69.7	67.9	82.8
70	71.5	70.3	83.1	70.8	69.7	82.8
100	71.7	69.7	82.1	76.2	73.7	84.9
150	71.8	69.2	80.9	71.6	69.3	81.6
Average ->	70.3	68.6	80.2	69.9	68.1	81.3
Deviation ->	2.1	1.9	2.7	4.3	3.7	3.1

7.3.5 Confusion Matrix and Fuzzy Clustering

We have shown the average F-Measures of the clustering with different parameters. Now we want to look at each cluster in detail to see what was going on there. Table 22 is the confusion matrix (or matching matrix) of the resulting eight clusters using weights 1:1 (1:1:1), FV_Length1=100, with MeSH. From this table we can see, six out of these eight clusters had high precisions (higher than 70%). There are two reasons why the other two clusters had low precisions. First, there were only a small number of

documents in the original two categories (18 for “BMC Endocr Disord” and 9 for “BMC Blood Disord”). Therefore, the feature vectors of these two clusters extracted from these documents could not precisely reflect the semantics of these two categories as feature vectors of other clusters did, in other words, the boundaries defined by these two feature vectors were not as clear as others and hence, some documents from other categories were “trapped” into these two clusters, which led to low precisions. Another reason is that they are also semantically close to other categories, which causes the misplacement of documents from other categories. For example, “BMC Endocr Disord” is close to “BMC Cardiovasc Disord”, hence nine documents from “BMC Cardiovasc Disord” were put into “BMC Endocr Disord”. Obviously, “BMC Blood Disord” is also close to “BMC Cardiovasc Disord”, so eight documents from “BMC Cardiovasc Disord” were put into cluster “BMC Blood Disord”.

Table 22: The Confusion Matrix of a Sample Clustering

	Actual number of documents of each category									**Precision (%)**
Number of documents in each resulted cluster		**BMC Endocr Disord (18)**	**BMC Neurol (141)**	**BMC Cardiovasc Disord (84)**	**BMC Blood Disord (9)**	**BMC Oral Health (53)**	**BMC Plant Biol (161)**	**Behav Brain Funct (90)**	**Cough (11)**	69.7 (avg)
	BMC Endocr Disord	16	15	9	0	0	3	3	0	34.8
	BMC Neurol	0	66	6	1	0	0	1	0	89.2
	BMC Cardiovasc Disord	0	23	59	0	0	0	1	0	71
	BMC Blood Disord	1	16	8	6	1	0	0	0	18.8
	BMC Oral Health	1	2	0	1	50	0	1	0	90.9
	BMC Plant Biol	0	1	1	0	2	156	0	0	97.5
	Behav Brain Funct	0	14	1	1	0	2	84	0	82.4
	Cough	0	4	0	0	0	0	0	11	73.3
	82.1(avg)	88.9	46.8	70.2	66.7	94.3	96.9	93.3	100	<-Recall (%)

To see relations between clusters, we computed the similarities of every two clusters as shown in Tables 23 and 24. An important observation on these two tables is that the similarities become smaller as the boundaries of clusters become clearer as new documents being added in.

Table 23: Similarities Between Clusters of the Starting Set

	BMC Endocr Disord	**BMC Neurol**	**BMC Cardiovasc Disord**	**BMC Blood Disord**	**BMC Oral Health**	**BMC Plant Biol**	**Behav Brain Funct**	**Cough**
BMC Endocr Disord(20)	1.0000	0.0437	0.0666	0.0583	0.0425	0.0307	0.0370	0.0327
BMC Neurol(19)	0.0437	1.0000	0.0624	0.0508	0.0526	0.0269	0.0593	0.0437
BMC Cardiovasc Disord(20)	0.0666	0.0624	1.0000	0.0341	0.0436	0.0305	0.0339	0.0412
BMC Blood Disord(20)	0.0583	0.0508	0.0341	1.0000	0.0340	0.0489	0.0362	0.0275
BMC Oral Health(20)	0.0425	0.0526	0.0436	0.0340	1.0000	0.0270	0.0359	0.0405
BMC Plant Biol(20)	0.0307	0.0269	0.0305	0.0489	0.0270	1.0000	0.0316	0.0234
Behav Brain Funct(20)	0.0370	0.0593	0.0339	0.0362	0.0359	0.0316	1.0000	0.0335
Cough(19)	0.0327	0.0437	0.0412	0.0275	0.0405	0.0234	0.0335	1.0000

Table 24: Similarities Between Clusters After Adding New Documents

	BMC Endocr Disord	**BMC Neurol**	**BMC Cardiovasc Disord**	**BMC Blood Disord**	**BMC Oral Health**	**BMC Plant Biol**	**Behav Brain Funct**	**Cough**
BMC Endocr Disord(66)	1.0000	0.0396	0.0328	0.0363	0.0282	0.0219	0.0304	0.0231
BMC Neurol(93)	0.0396	1.0000	0.0460	0.0365	0.0312	0.0131	0.0386	0.0283
BMC Cardiovasc Disord(103)	0.0328	0.0460	1.0000	0.0343	0.0343	0.0149	0.0282	0.0304
BMC Blood Disord(52)	0.0363	0.0365	0.0343	1.0000	0.0278	0.0225	0.0218	0.0258
BMC Oral Health(75)	0.0282	0.0312	0.0343	0.0278	1.0000	0.0173	0.0295	0.0356
BMC Plant Biol(180)	0.0219	0.0131	0.0149	0.0225	0.0173	1.0000	0.0187	0.0123
Behav Brain Funct(122)	0.0304	0.0386	0.0282	0.0218	0.0295	0.0187	1.0000	0.0257
Cough(34)	0.0231	0.0283	0.0304	0.0258	0.0356	0.0123	0.0257	1.0000

From these two tables one can easily tell that some clusters are close to each other while some are far away from others. This situation reflects the reality. In any domain, no

experts can set document categories that are evenly divided or distributed. As the number of documents grows, some categories will be close to (even overlap) each other while fall away from others. That is, in most situation, multi-membership of a document is more reasonable. However, for convenience, in many situations, each document is put in one category. Especially in the cases of conference and journal papers, where there are clearly defined tracks or areas, and each paper is usually accepted into one of these tracks or areas. Nevertheless, it is worth looking at this fuzzy clustering issue in our context of linear clustering with feature vectors. The following are two examples of memberships in the process of CS2CS linear clustering.

Example 1 Document Behav Brain Funct-2-_-1483829 (It belongs to category Behav Brain Funct in the original data set) is to be put into the eight existing clusters. With Exponential Increment Search (discussed in Chapter 5 Section 5.6), we found the best length of its feature vector is 15. The terms in its feature vector are "*melatonin; diseas; brain; oxid; cell; patient; antioxid; sleep; alzheim; protein; neuron; effect; acid; amyloid; radic*". The similarities between this feature vector and the feature vectors of eight clusters (calculated with equation 5.5) and the degrees of memberships (calculated with equation 5.17) are shown in Table 25. From this table, we can see that this document is most similar to cluster "BMC Neurol" with similarity 0.0176. In the case of hard clustering, it will be put into this cluster. However, it is also similar to others such as "BMC Blood Disord" (with similarity "0.0138") and "BMC Endocr Disord" (with similarity "0.0135"). In the case of fuzzy clustering, if the user set the threshold of degree of membership to be 10%, then this document would be put into "BMC Neurol",

"BMC Blood Disord", "BMC Endocr Disord", and "BMC Cardiovasc Disord", and "Behav Brain Funct", together with their degrees of memberships. Note, according to its original category "Behav Brain", this document would be *misplaced* into "BMC Neurol" in the case of hard clustering.

Table 25: The Memberships of A Document of Category Behav Brain Funct

	BMC Endocr Disord	BMC Neurol	BMC Cardiovasc Disord	BMC Blood Disord	BMC Oral Health	BMC Plant Biol	Behav Brain Funct	Cough
Similarity	0.0135	0.0176	0.0108	0.0138	0.0039	0.0059	0.0088	0.0066
Degree(%) of Membership	16.7	21.7	13.4	17.1	4.8	7.2	10.9	8.2

Example 2 Document Cough-3-_-2174508 (It belongs to category Behav Brain Funct in the original data set) is to be put into the eight existing clusters. With Exponential Increment Search (Chapter 5 Section 5.6), we found the best length of its feature vector is 10. The terms in its feature vector are "*capsaicin; reflex; cough; oral; chemesthesi; tast; test; capsiat; induc; differ*". The similarities between this feature vector and the feature vectors of eight clusters (calculated with equation 5.5) and the degrees of memberships (calculated with equation 5.17) are shown in Table 26. This document is most similar to cluster "Cough" which is its original category. In the case of hard clustering, it will be correctly put into the cluster where its original category specify. However, in the case of fuzzy clustering, it also belongs to cluster "BMC Oral Health" should the user set the threshold of degree of membership to 10%. Of course, it still has the highest degree of membership in the cluster "Cough".

Table 27 shows the comparison between CS2CS hard clustering and fuzzy clustering (with the simplest case where a document is assigned to one cluster with the degree

Table 26: The Memberships of A Document of Category Cough

	BMC Endocr Disord	BMC Neurol	BMC Cardiovasc Disord	BMC Blood Disord	BMC Oral Health	BMC Plant Biol	Behav Brain Funct	Cough
Similarity	0.0024	0.0048	0.0012	0.0014	0.0081	0.0008	0.0028	0.0561
Degree(%) of Membership	3.1	6.1	1.5	1.8	10.4	1.0	3.6	72.4

of membership). In this example, we used weights=1:1 (1:1:1), single-word terms, and considering MeSH. From this example we can see, the results are comparable. And we got a higher average F-Measure, precision, and recall with fuzzy clustering. Also, if we give partial credits of these *misplaced* documents in calculating precisions, we get even higher precisions which are recorded in the column "Count Membership". However, we will not apply the same adjustment in computing recall. Otherwise, the recalls would be more than 1 in some cases. Keep in mind this is just an example used to demonstrate the idea that our CS2CS algorithm can easily do fuzzy clustering without much change. The difference between their results would be data dependent. That is, on one collection, the hard clustering does better, on another, the fuzzy clustering may do better.

Table 27: Comparison Between CS2CS Hard Clustering and Fuzzy Clustering

	CS2CS Hard Clustering			**CS2CS Fuzzy Clustering**			
FV-Length1	**F-Measure (%)**	**Precision (%)**	**Recall (%)**	**F-Measure (%)**	**Precision (%)**		**Recall (%)**
					W/O Membership	**Count Membership**	
10	68.9	67.3	78.1	70.5	68.7	80.9	77.9
20	66.6	65.4	75.9	72.6	70.8	81.2	81.1
50	71.2	69.8	81.3	71.8	69.3	79.2	82.1
70	71.5	70.3	83.1	73.8	72.1	80.8	83.1
100	71.7	69.7	82.1	74.6	72.6	80.8	82.8
150	71.8	69.2	80.9	74.2	71.9	80.3	81.4
Average ->	70.3	68.6	80.2	72.9	70.9	80.5	81.4
Deviation ->	2.1	1.9	2.7	1.6	1.6	0.7	1.9

Table 28: Confusion Matrix of Clusters Before Splitting

	Actual number of documents of each category										Precision (%)
Number of documents in each resulted cluster		BMC Endocr Disord (18)	BMC Neurol (141)	BMC Cardiovasc Disord (84)	BMC Blood Disord (9)	BMC Oral Health (53)	BMC Plant Biol (161)	Behav Brain Funct (90)	Cough (11)	BMC Cancer (76)	63.4 (avg)
	BMC Endocr Disord	12	3	6	0	0	5	2	0	12	30
	BMC Neurol	0	79	7	1	0	1	2	0	2	85.9
	BMC Cardiovasc Disord	1	24	56	0	0	0	0	0	1	68.3
	BMC Blood Disord	1	6	3	6	2	0	0	0	24	14.3
	BMC Oral Health	1	4	0	1	50	0	1	0	2	84.7
	BMC Plant Biol	3	11	9	0	1	152	1	0	34	72
	Behav Brain Funct	0	10	3	1	0	3	84	0	0	83.2
	Cough	0	4	0	0	0	0	0	11	1	68.8
	79.8(avg)	66.7	56	66.7	66.7	94.3	94.4	93.3	100	n/a	<-Recall (%)

7.3.6 Cluster Splitting and Merging

Splitting To test our strategy of splitting discussed in Section 5.5 of Chapter 5, we included 76 documents of another category “BMC Cancer” into the new document set (it has nine categories now) to be added into the starting set where there are eight categories as before. Tables 28 and 29 show the confusion matrices before and after cluster splitting. In this test we used weights=1:1 (1:1:1), FV_Length1=100, single-word terms, and considering MeSH.

From Table 28 we can see, that around 1/3 (24 out of 76) of the new documents of category “BMC Cancer” go to cluster “BMC Blood Disord” which is understandable since these two categories are semantically close to each other. However, in the case of

hard clustering, this makes the precision of the cluster “BMC Blood Disord” very low (14.3%). This problem could be solved by the splitting procedure we proposed. Table

Table 29: Confusion Matrix of Clusters After Splitting

	Actual number of documents of each category										Precision (%)
Number of documents in each resulted cluster		BMC Endocr Disord (18)	BMC Neurol (141)	BMC Cardiovasc Disord (84)	BMC Blood Disord (9)	BMC Oral Health (53)	BMC Plant Biol (161)	Behav Brain Funct (90)	Cough (11)	BMC Cancer (76)	66.8 (avg)
	BMC Endocr Disord	12	3	6	0	0	5	2	0	12	30
	BMC Neurol	0	79	7	1	0	1	2	0	2	85.9
	BMC Cardiovasc Disord	1	24	56	0	0	0	0	0	1	68.3
	BMC Blood Disord	0	0	0	3	0	0	0	0	3	50
	BMC Oral Health	1	4	0	1	50	0	1	0	2	84.7
	BMC Plant Biol	3	11	9	0	1	152	1	0	34	72
	Behav Brain Funct	0	10	3	1	0	3	84	0	0	83.2
	Cough	0	4	0	0	0	0	0	11	1	68.8
	BMC Cancer	1	6	3	3	2	0	0	0	21	58.3
	70.3 (avg)	66.7	56	66.7	33.3	94.3	94.4	93.3	100	27.6	<-Recall (%)

29 shows the result of this splitting. It results in a new cluster “BMC Cancer”. 21 out of 24 of the *misplaced* documents of “BMC Cancer” in cluster “BMC Blood Disord” have been successfully moved into this new cluster. Moveover, the precisions of both newly formed clusters by splitting are higher than that of cluster “BMC Blood Disord” before being split. This in turn makes the average precision of all the clusters higher than before splitting. Even though we have a little lower recall (actually the recall almost stays the same if we consider the recall for “BMC Cancer” as zero before splitting since we did not have a cluster of “BMC Cancer” at all), the most important thing is that, through this

splitting, we obtained more clearly defined clusters instead of the old ambiguous cluster.

Merging Based on the result of splitting shown in Table 29, we continually add 46 more

Table 30: Confusion Matrix of Clusters Before Merging

	Actual number of documents of each category										Precision (%)
Number of documents in each resulted cluster		BMC Endocr Disord (18)	BMC Neurol (141)	BMC Cardiovasc Disord (84)	BMC Blood Disord (9)	BMC Oral Health (53)	BMC Plant Biol (161)	Behav Brain Funct (90)	Cough (11)	BMC Cancer (122)	66.1 (avg)
	BMC Endocr Disord	12	3	6	0	0	5	2	0	26	22.2
	BMC Neurol	0	79	7	1	0	1	2	0	2	85.9
	BMC Cardiovasc Disord	1	24	56	0	0	0	0	0	1	68.3
	BMC Blood Disord	0	0	0	3	0	0	0	0	5	37.5
	BMC Oral Health	1	4	0	1	50	0	1	0	2	84.7
	BMC Plant Biol	3	11	9	0	1	152	1	0	35	71.7
	Behav Brain Funct	0	10	3	1	0	3	84	0	0	83.2
	Cough	0	4	0	0	0	0	0	11	2	64.7
	BMC Cancer	1	6	3	3	2	0	0	0	49	76.7
	71.7 (avg)	66.7	56	66.7	33.3	94.3	94.4	93.3	100	40.2	<-Recall (%)

documents from category “BMC Cancer”, the confusion matrix of the new result is shown in Table 30. In this new result, 14 out of these 46 documents were added to cluster “BMC Endocr Disord”, that made the precision of this cluster very low (30%). However, since so many documents (26) are from “BMC Cancer”, this makes the similarity of this two clusters getting closer to the extent that we consider merging them. Table 31 shows the confusion matrix after merging with the category of “BMC Endocr Disord” present. Table 32 shows the confusion matrix with category of “BMC Endocr Disord” absorbed into category “BMC Cancer”. In other words, it is the result if we consider these two category

as the same one. Obviously, this will cause both the precision and the recall to increase. Of course, if more documents of “BMC Endocr Disord” are added to this cluster later on, it may be split into two clusters again and thus cluster “BMC Endocr Disord” will be back on.

From Table 31 we see that there also are many (35) “BMC Cancer” documents in cluster “BMC Plant Biol”. However, since this cluster is bigger than the cluster “BMC Endocr Disord” (177 versus 54 documents), the similarity between “BMC Plant Biol” and “BMC Cancer” is still under the threshold of merging. Therefore, we do not merge them at this point.

Table 31: Confusion Matrix of Clusters After Merging with Both Categories Remaining

	Actual number of documents of each category										Precision (%)
Number of documents in each resulted cluster		BMC Endocr Disord (18)	BMC Neurol (141)	BMC Cardiovasc Disord (84)	BMC Blood Disord (9)	BMC Oral Health (53)	BMC Plant Biol (161)	Behav Brain Funct (90)	Cough (11)	BMC Cancer (122)	70 (avg)
	BMC Neurol	0	79	7	1	0	1	2	0	2	85.9
	BMC Cardiovasc Disord	1	24	56	0	0	0	0	0	1	68.3
	BMC Blood Disord	0	0	0	3	0	0	0	0	5	37.5
	BMC Oral Health	1	4	0	1	50	0	1	0	2	84.7
	BMC Plant Biol	3	11	9	0	1	152	1	0	35	71.7
	Behav Brain Funct	0	10	3	1	0	3	84	0	0	83.2
	Cough	0	4	0	0	0	0	0	11	2	64.7
	BMC Cancer	13	9	9	3	2	5	2	0	75	63.6
	71.7 (avg)	n/a	56	66.7	33.3	94.3	94.4	93.3	100	40.2	<-Recall (%)

Table 32: Confusion Matrix of Clusters After Merging with One Category Remaining

	Actual number of documents of each category									Precision (%)
Number of documents in each resulted cluster		BMC Neurol (141)	BMC Cardiovasc Disord (84)	BMC Blood Disord (9)	BMC Oral Health (53)	BMC Plant Biol (161)	Behav Brain Funct (90)	Cough (11)	BMC Cancer (140)	71.3 (avg)
	BMC Neurol	79	7	1	0	1	2	0	2	85.9
	BMC Cardiovasc Disord	24	56	0	0	0	0	0	2	68.3
	BMC Blood Disord	0	0	3	0	0	0	0	5	37.5
	BMC Oral Health	4	0	1	50	0	1	0	3	84.7
	BMC Plant Biol	11	9	0	1	152	1	0	38	71.7
	Behav Brain Funct	10	3	1	0	3	84	0	0	83.2
	Cough	4	0	0	0	0	0	11	2	64.7
	BMC Cancer	9	9	3	2	5	2	0	88	74.6
	75.1 (avg)	56	66.7	33.3	94.3	94.4	93.3	100	62.9	<-Recall (%)

7.3.7 ICF Versus IDF

As we explained in Section 5.3 of Chapter 5, we used ICF as shown in equation 5.3 to normalize the feature vectors across clusters. To demonstrate its importance in finding feature vectors of clusters and hence in our CS2CS linear clustering, here we compare the result of using equation 5.3 to that using IDF like normalization as shown in the following equation.

$$W_{ij1} = W_{ij2} log \frac{k}{|\{c : t_j \in c\}|} \tag{7.10}$$

Where W_{ij1} and W_{ij2} are the weights of term t_j in the feature vector of cluster i after and before this adjustment, respectively. k is the number of clusters. Table 33 shows the sharp comparison of the result using equation 5.3 and the result using 7.10. The result of using ICF is much better than using IDF like adjustment. Just as we analyzed in Chapter

5 Section 5.3, this is because some terms that occur in all the documents of a cluster were eliminated because of the use of logarithm and thus some useful information were lost.

In addition, In Chapter 5 Section 5.3 we argued why we choose occurrence counting over weight sum in forming the cluster feature vectors. Here we also shows the result of using ICF with weight sum in Table 34. Even though the result of ICF with weight sum was better than using IDF like approach, it was still not as good as using ICF with occurrence counting. These results further confirm our analysis on the formation and normalization of cluster feature vectors.

Lastly, regarding the lengths of document feature vectors which are used to form cluster feature vectors, we show the comparison between the results of fixed lengths of document feature vectors and varied lengths of document feature vectors used to form cluster feature vectors. The varied lengths are that of new documents which are determined by Exponential Increment Search, as discussed in Section 5.6 of Chapter 5. For the cluster feature vectors of the starting set, we use fixed lengths of document feature vectors in both cases. Table 35 shows this comparison. From this table, we can see that the average F-Measure of these two are almost the same (70.3% vs. 70.7%).

Figure 29 summarizes these comparisons by showing the related F-Measure, precision, and recall. In all these three tests we used weights=1:1 (1:1:1), single-word terms, and considering MeSH.

Table 33: Comparison of Using ICF and IDF Like Weight Adjustments

	ICF Weight Adjustment			IDF like Weight Adjustment		
FV-Length1	**F-Measure (%)**	**Precision (%)**	**Recall (%)**	**F-Measure (%)**	**Precision (%)**	**Recall (%)**
10	68.9	67.3	78.1	52.8	53.1	63.3
20	66.6	65.4	75.9	51.4	50.7	61.8
50	71.2	69.8	81.3	52.3	51.2	64.5
70	71.5	70.3	83.1	47.9	47.3	60.9
100	71.7	69.7	82.1	45.9	44.7	59
150	71.8	69.2	80.9	42	42.2	53.9
Average ->	70.3	68.6	80.2	48.7	48.2	60.6
Deviation ->	2.1	1.9	2.7	4.3	4.2	3.8

Table 34: Comparison of ICF with Occurrence Counting and ICF with Weight Sum

	ICF with Occurrence Counting			ICF with Weights Sum		
FV-Length1	**F-Measure (%)**	**Precision (%)**	**Recall (%)**	**F-Measure (%)**	**Precision (%)**	**Recall (%)**
10	68.9	67.3	78.1	63.9	63.4	71.7
20	66.6	65.4	75.9	64.1	63.2	76.1
50	71.2	69.8	81.3	51.7	47.5	69.4
70	71.5	70.3	83.1	46.6	43.9	58
100	71.7	69.7	82.1	52.8	53.2	62
150	71.8	69.2	80.9	56.2	55	63.5
Average ->	70.3	68.6	80.2	55.9	54.4	66.8
Deviation ->	2.1	1.9	2.7	7.0	8.0	6.8

Table 35: Comparison of Using Fixed and Varied Lengths of Document Feature Vectors

	ICF with fixed length of document feature vector			ICF with fixed length of document feature vector only for starting set		
FV-Length1	**F-Measure (%)**	**Precision (%)**	**Recall (%)**	**F-Measure (%)**	**Precision (%)**	**Recall (%)**
10	68.9	67.3	78.1	70	68.2	78.2
20	66.6	65.4	75.9	69.3	68	78.5
50	71.2	69.8	81.3	67.7	67	79.1
70	71.5	70.3	83.1	69.7	68	80.6
100	71.7	69.7	82.1	72.9	71.7	79
150	71.8	69.2	80.9	74.3	72.2	80.8
Average ->	70.3	68.6	80.2	70.7	69.2	79.4
Deviation ->	2.1	1.9	2.7	2.5	2.2	1.1

7.3.8 Results of Using CS2CS on Physics Documents

Just as we did for CS-VS which is discussed in Subsection 7.2.3, we also tested CS2CS on physics collection downloaded from Nature Physics Portal [20], to test the consistency of the performance of CS2CS in different domains. We put the nine sub-topics or categories and their abbreviations here again: Astrophysics (AP), Atomic and

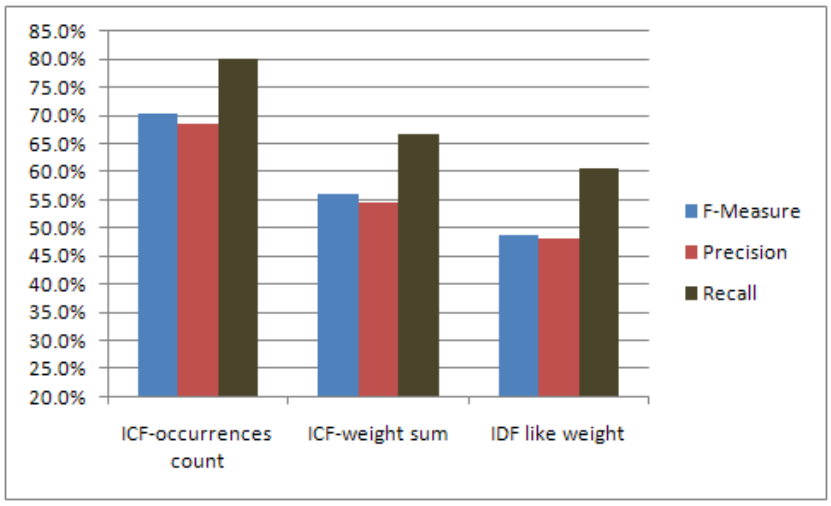

Figure 29: Results of Using Different Weight Normalization Approaches of Terms in Cluster Feature Vectors

molecular physics (AMP), Biological physics (BP), Chemical physics (CP), Condensed-matter physics (CMP), Materials physics (MP), Nanotechnology (NP), Optical physics (OP), and Quantum physics (QP). We divided this collection with 411 papers into two sets. Set 1 contains 90/80 documents with 10 from each category. Set 2 contains the other documents. CS2CS uses Set 1 as starting set, and add documents in Set 2 to Set 1 one by one, the results are for clustering Set 2. The other algorithms do clustering on Set 2 only. Table 36 shows the results of using different clustering algorithms on the physics set 2. Even though the overall F-Measure are all low using these algorithms, CS2CS is still much better than other algorithms.

To investigate the reason of why the results are much lower than that over biomedical documents, we computed the similarities between clusters using their cluster feature vectors. The results are shown in Table 37. Comparing these similarities to those between

Table 36: Results of Using Different Algorithms on Physics Documents

Categories ->	AP, AMP, BP, CP, CMP, MP, NP, OP, QP			AP, BP, CP, CMP, MP, NP, OP, QP		
Algorithm	**F-Measure (%)**	**Precision (%)**	**Recall (%)**	**F-Measure (%)**	**Precision (%)**	**Recall (%)**
K-Means	17	16	21.4	17.7	17.8	20.8
Bisecting K-Means	16.3	16.1	22.3	19.1	18.7	23.6
CS-VS	28.4	29.2	41.2	28.7	29.4	44
CS2CS	33	34.3	35.1	41.1	42.3	43.6

biomedical documents (Tables 23 and 24), it is easy to tell that the similarities between physics document clusters are much higher than that between biomedical document clusters. This means, the boundary of categories of this physics collection is not as clear as that in the biomedical collection. We also notice that the similarities between the cluster AMP are higher than other similarities. Our hypothesis was that if we remove this category, we would get better result. The right half of Table 36 proves our assumption. The results are better than that with all nine categories which are show on the left half of the same table.

Table 37: Similarities Between Physics Document Clusters

	MP	**AMP**	**CMP**	**AP**	**QP**	**OP**	**CP**	**BP**	**NP**
MP(35)	1.0000	0.0504	0.0478	0.0308	0.0353	0.0479	0.0454	0.0568	0.0686
AMP(65)	0.0504	1.0000	0.0783	0.0359	0.0513	0.0760	0.0395	0.0470	0.0496
CMP(74)	0.0478	0.0783	1.0000	0.0326	0.0504	0.0388	0.0369	0.0340	0.0643
AP(41)	0.0308	0.0359	0.0326	1.0000	0.0267	0.0481	0.0395	0.0291	0.0285
QP(69)	0.0353	0.0513	0.0504	0.0267	1.0000	0.0450	0.0286	0.0401	0.0359
OP(43)	0.0479	0.0760	0.0388	0.0481	0.0450	1.0000	0.0345	0.0407	0.0462
CP(25)	0.0454	0.0395	0.0369	0.0395	0.0286	0.0345	1.0000	0.0439	0.0453
BP(28)	0.0568	0.0470	0.0340	0.0291	0.0401	0.0407	0.0439	1.0000	0.0462
NP(31)	0.0686	0.0496	0.0643	0.0285	0.0359	0.0462	0.0453	0.0462	1.0000

7.4 Results from InterOBO

7.4.1 Synonym Based Transitive Equivalence

After analyzing the synonym relations between OBO ontologies, we found 6123 instances of Case 1, 78 instances for Case 2 and 66818 instances for Case 3, that are described in Subsection 6.3.1. Tables 38, 39, and 40 show representative examples of the cases. In these tables, $C1$ and $C2$ stand for the related concepts while $S1$ and $S2$ are the synonyms of concepts $C1$ in Ontology Oi and $C2$ in Ontology Oj respectively.

Table 38: Synonym Transitivity Case 1

Oi	**Oj**	**Instance**	**Example**
O23	O6	236	C1=S2=medicine C2=drug
O23	O24	114	C1=S2=neuroleukin C2=g6pi_human
O22	O16;O17	105	C1=stage 29, midbrain hindbrain boundary (mhb) C2=S1=isthmus
O12	O25	102	C1=S2=episternum C2=proepisternum
O15	O24	98	C1=sodium-translocating f-type atpase activity C2=S1=atp synthase
O6	O23	50	C1=dihydrogen C2=S1=hydrogen
O15	O23	62	C1=phototransduction C2=S1=phototransduction, visible light, light adaptation
O15	O10	55	C1=S2=protein kinase c activation C2=pkc activation signaling
O22	O16;O17;O23	53	C1=stage 22, forebrain C2=S1=prosencephalon
O15	O23	39	C1=actin filament C2=S1=microfilament

Table 39: Synonym Transitivity Case 2

Oi	**Oj**	**Instance**	**Example**
O6	O23	16	C1=S2=l-serine C2=S1=serine
O23	O6	13	C1=S2=azacitidine C2=S1=5-azacytidine
O31	O4	5	C1=S2=nucellus C2=S1=megasporangium
O15	O24	4	C1=S2=pre-replicative complex C2=S1=pre-rc

Table 40: Synonym Transitivity Case 3

Oi	Oj	Instance	Examples
O6	O23	1044	C1=dioxygen(.1+) C2=peroxide S1=S2=O2
O6	O24	375	C1=azo group C2=notc2_mouse S1=S2=N2
O15	O24	300	C1=ha1 clathrin adaptor C2=jun_human S1=S2=AP1
O23	O24	184	C1=heterozygote C2=transporter S1=S2=carrier
O24	O6;O23	64	C1=deca_drome C2=hydroxide S1=S2=HO
O25	O12	55	C1=gonostylus C2=unguis S1=S2=claw
O22	O1;O4;O16; O17;O39	53	C1=stage 20, hindbrain C2=hindbrain S1=S2=rhombencephalon
O22	O16;O17; O39	53	C1=stage 28, hindbrain C2=hindbrain S1=S2=rhombencephalon
O24	O23	42	C1=ifna1_human C2=interferon S1=S2=IFN
O32	O36	39	C1=cotyledon emergence C2=1.01-seedling emergence S1=S2=maize growth stage-1.1

7.4.2 Ontology Connection Patterns

Table 41 shows some of quantitatively connecting patterns captured from multiple ontologies. In this example, the strongest connecting patterns are between Human_dev_anat_abstract and Human_dev_anat_staged, and between Po_anatomy and Zea_mays_anatomy. The three ontologies that contain the strongest quantitatively connecting patterns are Human_dev_anat_abtract, Human_dev_anat_staged and Brenda.

Table 41: Quantitative Connection Patterns

Ontology 1	Ontology 2	Cp1	Cp2
Human_dev_anat_abstract	Human_dev_anat_staged	0.051816801	0.103584007
Po_anatomy	Zea_mays_anatomy	0.034859457	0.079037801
Adult_mouse_anatomy	Brenda	0.017509850	0.070480748
Flybase_vocab	Plant_environment	0.016062465	0.066852368
Brenda	Po_anatomy	0.006391173	0.036148766
Human_dev_anat_abstract	Zebrafish_anatomy	0.004833003	0.035294118
Brenda	Cell	0.004047477	0.028422877
Brenda	Human_dev_anat_abstract	0.004040174	0.032816773
Brenda	Zebrafish_anatomy	0.003760804	0.031130530
Adult_mouse_anatomy	Zebrafish_anatomy	0.003140380	0.027737578
Adult_mouse_anatomy	Human_dev_anat_staged	0.002772477	0.023163161
Mao	Psi_mi	0.002011567	0.022857143
Brenda	Human_dev_anat_staged	0.001136602	0.013924902

Table 42: Semantic Connection Patterns

ID	Ontology 1	Ontology 2	Overlapped Concepts	Patterns	Score	Std
P1	Mesh	Fly Development	Drosophila	[0.9, 0.2]	4.5	0
P2	Loggerhed Nesting	Event	Event	[0.4, 0.1]	4.0	0
P3	Mao	Go Daily Termdb	Cellular component	[1.0, 0.2]	3.08	7.66
			Molecular function	[1.0, 0.2]		
			Biological process	[1.0, 0.1]		
			Phosphorylation	[1.0, 0.8]		
P4	Attribute and Value	Rex	Coordination	[1.0, 0.9]	3.06	2.75
			Process	[1.0, 0.2]		
P5	MeSH	Plasmodium Life Cycle	Parasite	[1.0, 0.2]	2.00	3.46
			Sporozoite	[1.0, 1.0]		
			Zygote	[1.0, 1.0]		
			Oocyst	[1.0, 1.0]		
P6	Psi Mi	Sequence	Sequence variant Mutation	[1.0, 0.5]	2.00	0
P7	Sequence	Molecule Role	Gap	[1.0, 0.8]	1.82	1.62
			Protein	[1.0, 0.3]		
P8	Go Daily Termdb	Cell	Xanthophore	[1.0, 1.0]	1.82	8.22
			Cell	[1.0, 0.1]		
P9	Event	Mammalian Phenotype	Cell death Necrosis	[0.9, 0.9]	1.81	1.86
			Tumorigenesis	[1.0, 0.3]		
			Diarrhea	[1.0, 0.9]		
P10	Flybase Vocab	Rex	Reduction	[1.0, 0.5]	1.71	0.4
			Detachment	[1.0, 0.7]		

Table 42 shows some semantic connection patterns identified among these OBO ontologies. The pattern contains some connection pattern instances as [level value in Ontology 1,level value in Ontology 2]. For example [0.9, 0.2] means that the CCP in $O1$ is 0.9 and CCP in $O2$ is 0.2. This pattern implies that the concept appears close to the leaf node in $O1$ while it appears close to the root node in $O2$. By definition, it is a connecting pattern between $O1$ and $O2$. Figure 30 shows the plot of the semantic pattern distribution of the 10 patterns listed in Table 42.

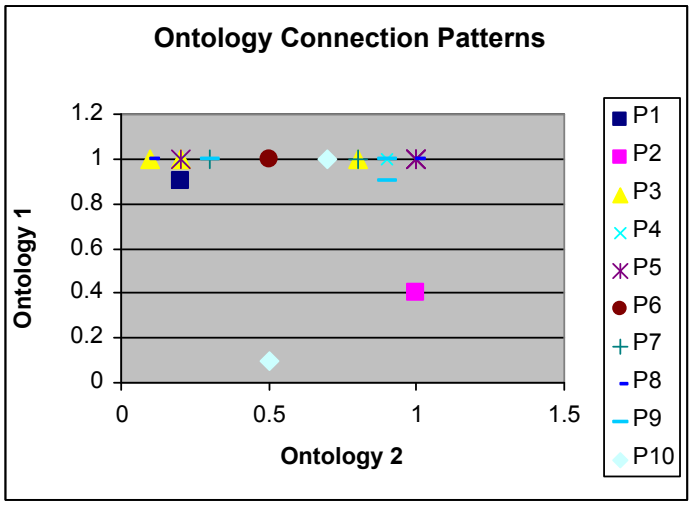

Figure 30: Semantic Connection Patterns

7.4.3 Ontology Clustering

Following the method described in 6.4, we clustered the 40 OBO ontologies using MCL. As shown in Table 43, the clustering experiments resulted in seven clusters for each of the two formulas, when degree of concept overlap was chosen to be the metric of similarity. Both approach I (probability-based) and approach II (area-based), are largely consistent in clustering the OBO ontologies into seven clusters; a few differences are observed. The following ontologies fall into different clusters depending on choice of approach: Dictyostelium Discoideum Anatomy (O7), Fungal Anatomy (O14), Fly Development (O40), Rex (O34) and Plasmodium_life Cycle (O30). Three of the ontologies Emap (O9), Evidence_code (O11), and Image (O18) were found to be singletons, i.e., in clusters by themselves.

As shown Table in 44, more substantial differences between the two approaches I and III were observed where the area-based similarity was based on common edges

(parentchild term pair) in III. While the Concept-based metric resulted in seven clusters, the edge-based one resulted in six clusters. They showed different results. Specifically, the edge-based clustering showed different results for the following ontologies: Arabidopsis Development (O2), Attribute and Value (O3), Dictyostelium Discoideum Anatomy (O7), Disease Ontology (O8), Loggerhead Nesting (O19), Mosquito Anatomy (O25), Pathway (O27), Plant Trait (O29), Plasmodium Life Cycle (O30), Po Temporal (O32), Psi Mi (O33), Temporal Gramene (O36), Worm Development (O37), Zea Mays Anatomy (O38) and Fly Development (O40). The clustering graphs shown in Figures 31-33 are generated using the Pajek [23] that is the program for the large network analysis.

Table 43: Ontology Clustering Based on Shared Concepts

ID	Ontology clustering using Approach I	Ontology clustering using Approach II
CC1	Adult_mouse_anatomy (O1), Brenda (O4), Chebi (O6), **Dictyostelium_discoideum_anatomy (O7),** Fly_anatomy (O12), **Fungal_anatomy (O14),** Human_dev_anat_abstract (O16), Human_dev_anat_staged (O17), Medaka_anatomy_development (O22), Mesh (O23), Molecule_role (O24), Mosquito_anatomy (O25), **Plasmodium_life_cycle (O30),** Zebrafish_anatomy (O39), **Fly_development (O40)**	Adult_mouse_anatomy (O1), Brenda (O4), Chebi (O6), Fly_anatomy (O12), Human_dev_anat_abstract (O16), Human_dev_anat_staged (O17), Medaka_anatomy_development (O22), Mesh (O23), Molecule_role (O24), Mosquito_anatomy (O25), Zebrafish_anatomy (O39)
CC2	Attribute_and_value (O3), Flybase_vocab (O13), Loggerhead_nesting (O19), Plant_environment (O28), Plant_trait (O29)	Attribute_and_value (O3), Flybase_vocab (O13), Loggerhead_nesting (O19), Plant_environment (O28), Plant_trait (O29), **Rex (O34)**
CC3	Cell (O5), Po_anatomy (O31), Worm_development (O37), Zea_mays_anatomy (O38)	Cell (O5), **Dictyostelium_discoideum_anatomy (O7),** **Fungal_anatomy (O14),** Go_anatomy (O31), Worm_development (O37), Zea_mays_anatomy (O38), **Fly_development (O40)**
CC4	Event (O10), Go (O15), Pathway (O27), **Rex (O34)**	Event (O10), Go (O15), Pathway (O27)
CC5	Mao (O21), Psi_mi (O33), Sequence (O35)	Mao (O21), Psi_mi (O33), Sequence (O35)
CC6	Disease_ontology (O8), Mammalian_phenotype (O20), Mouse_pathology (O26)	Disease_ontology (O8), Mammalian_phenotype (O20), Mouse_pathology (O26)
CC7	Arabidopsis_development (O2), Po_temporal (O32), Temporal_gramene (O36)	Arabidopsis_development (O2), **Plasmodium_life_cycle (O30),** Po_temporal (O32), Temporal_gramene (O36)
Singletons	Emap (O9), Evidence_code (O11), Image (O18)	Emap (O9), Evidence_code (O11), Image (O18)

Table 44: Comparison of Ontology Clustering Based on Shared Concepts and Links

ID	Approach I	Approah III	
CC1	Adult_mouse_anatomy (O1), Brenda (O4), Chebi (O6), **Dictyostelium_discoideum_anatomy (O7),** Fly_anatomy (O12), Fungal_anatomy (O14), Human_dev_anat_abstract (O16), Human_dev_anat_staged (O17), Medaka_anatomy_development (O22), Mesh (O23), Molecule_role (O24), **Mosquito_anatomy (O25),** **Plasmodium_life_cycle (O30),** Zebrafish_anatomy (O39), **Fly_development (O40)**	RC1	Adult_mouse_anatomy (O1), Brenda (O4), Chebi (O6), Fly_anatomy (O12), Fungal_anatomy (O14), Human_dev_anat_abstract (O16), Human_dev_anat_staged (O17), Medaka_anatomy_development (O22), Mesh (O23), Molecule_role (O24), Zebrafish_anatomy (O39)
CC2	Attribute_and_value (O3), **Flybase_vocab (O13),** **Loggerhead_nesting (O19),** Plant_environment (O28), **Plant_trait (O29)**		Flybase_vocab (O13), Plant_environment (O28)
CC3	Cell (O5), Po_anatomy (O31), **Worm_development (O37),** **Zea_mays_anatomy (O38)**	RC3	Cell (O5), Po_anatomy (O31)
CC4	Event (O10), Go (O15), **Pathway (O27),** Rex (O34)	RC4	Event (O10), Go (O15), Rex (O34)
CC5	Mao (O21), **Psi_mi (O33),** Sequence (O35)	RC5	Mao (O21), Sequence (O35)
CC6	**Disease_ontology (O8),** Mammalian_phenotype (O20), Mouse_pathology (O26)	RC6	Mammalian_phenotype (O20), Mouse_pathology (O26)
CC7	Arabidopsis_development (O2), Po_temporal (O32), Temporal_gramene (O36)	Other	Arabidopsis_development (O2), Attribute_and_value (O3), Dictyostelium_discoideum_anatomy (O7), Disease_ontology (O8), Emap(O9), Evidence_code(O11), Image (O18), Loggerhead_nesting (O19), Mosquito_anatomy (O25), Pathway (O27), Plant_trait (O29), Plasmodium_life_cycle (O30), Po_temporal (O32), Psi_mi (O33), Temporal_gramene (O36), Worm_development (O37), Zea_mays_anatomy (O38), Fly_development (O40)
Other	Emap (O9), Evidence_code (O11), Image (O18)		

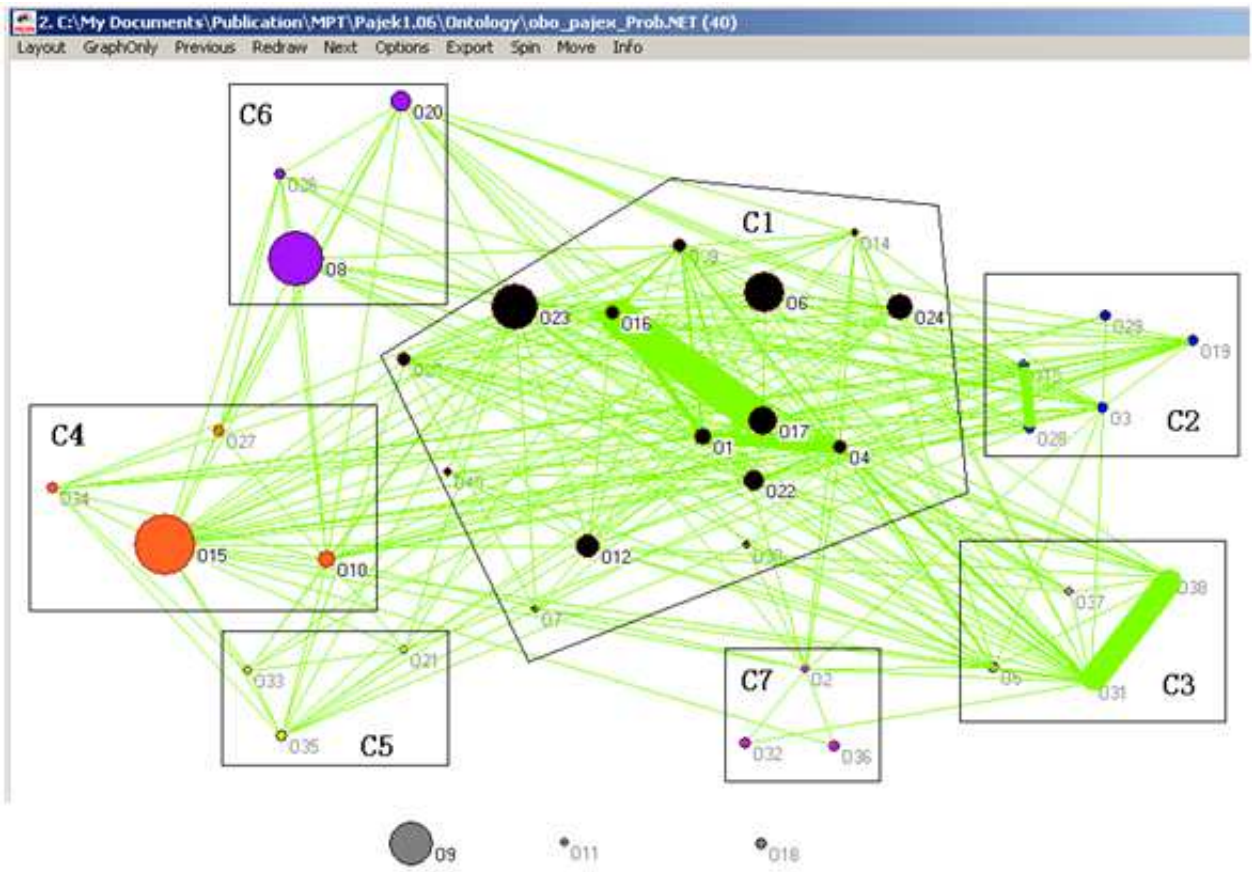

Figure 31: Ontology Clustering Result of Approach I

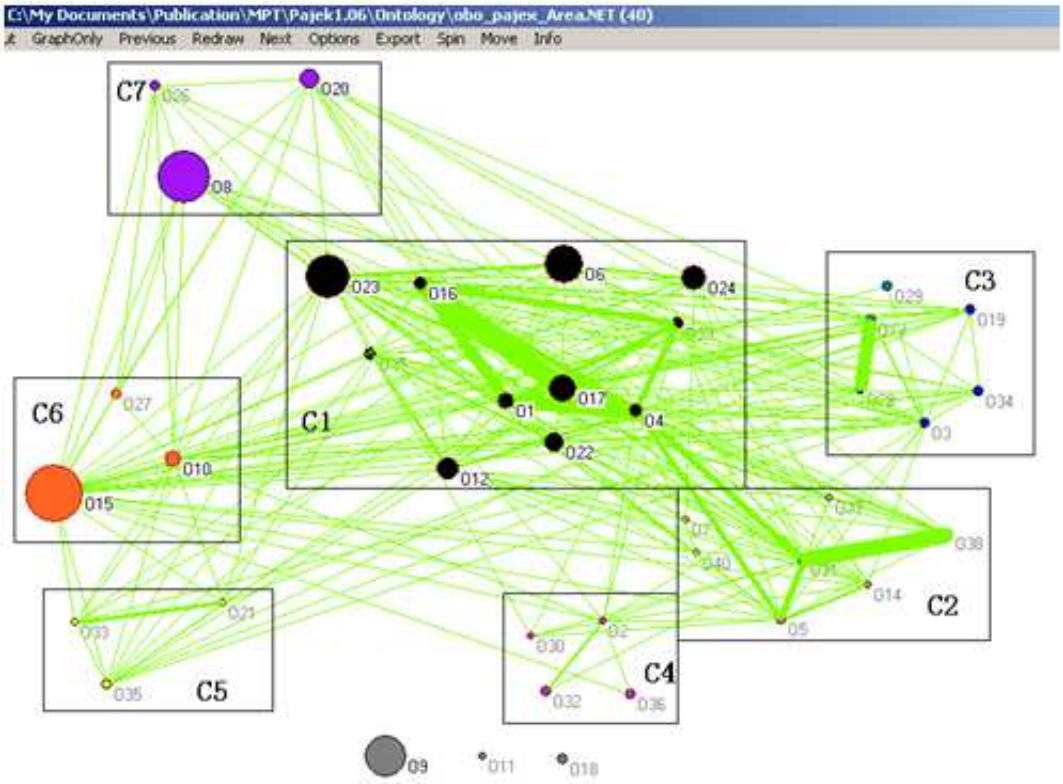

Figure 32: Ontology Clustering Result of Approach II

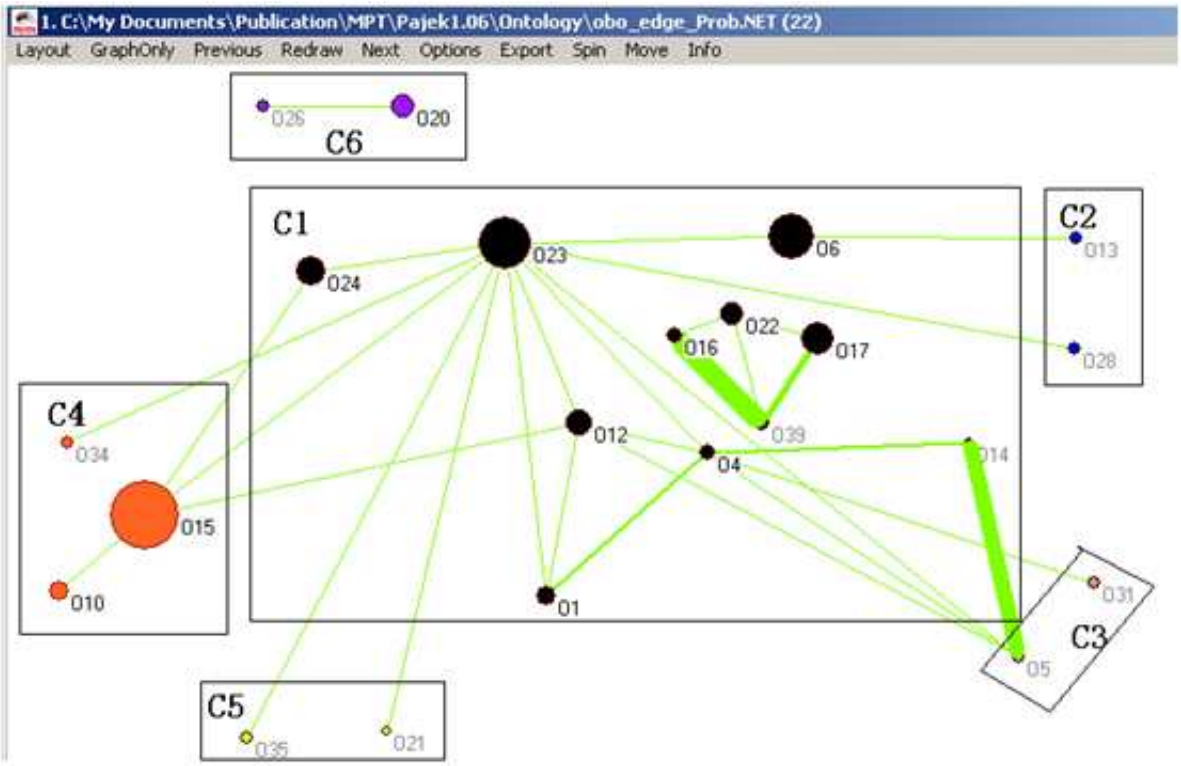

Figure 33: Ontology Clustering Result of Approach III

7.4.4 InterOBO Prototype Development

We have implemented a prototype of InterOBO to establish proof of concept for the proposed model for analyzing and clustering ontologies. The InterOBO prototype has been implemented using Java, Java 2 Platform Standard Edition (J2SE platform) 5.0 and SuSe Linux on an AMD Opteron dual CPU machine with 2.4 GHz CPU, 4 Gb memory, and a 120 Gb hard disk. The backend database is MySQL version 5.0. InterOBO maintains a representation of the OBO ontologies. In order to browse and search the OBO ontology analysis and clustering information, InterOBO provides query interfaces

(shown in Figure 34):

- Query on a specific concept: for a given concept, this provides the description of the concept, synonyms, information on ontologies that contain the concept.
- Query on the overlapping relationships between ontologies: for a given set of ontologies, try to find overlapping relationships such as shared concepts, shared links, shared properties.
- Query on the shared concepts and links through the overlapped ontologies: for a given ontology, try to find any links to other ontologies and concepts or properties involved in the connections.

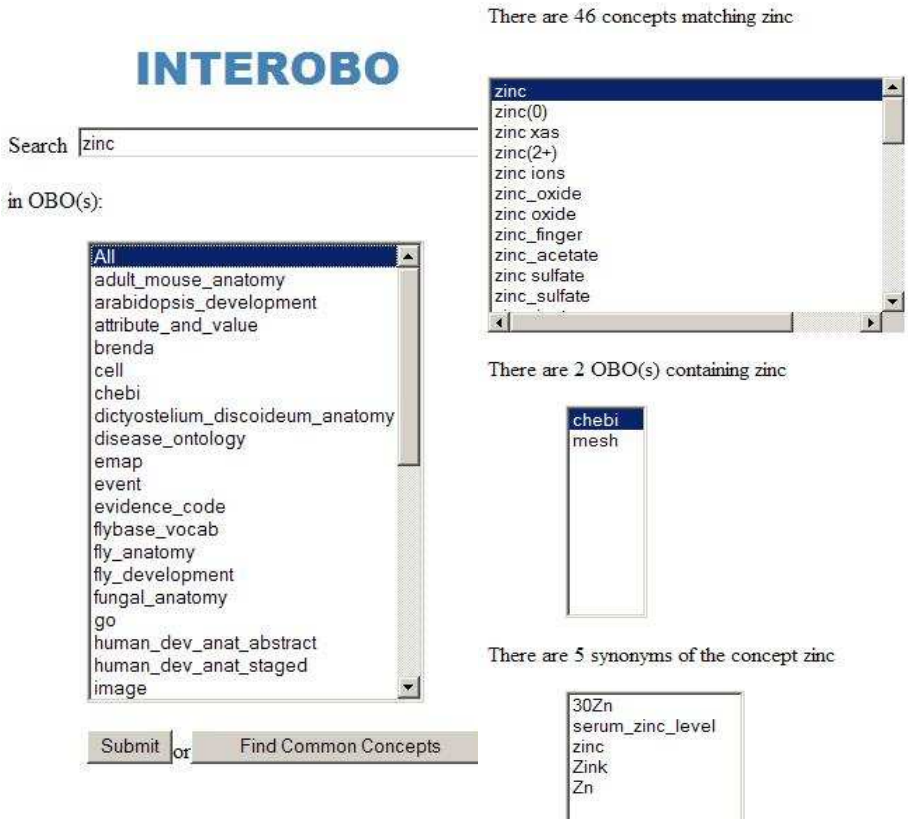

Figure 34: InterOBO Query Interfaces

CHAPTER 8

SUMMARY AND FUTURE WORK

8.1 Citonomy

8.1.1 Summary

In this dissertation, a framework, called Citonomy, was presented to utilize the semantic information, especially the citation semantics in scientific documents, to improve the quality of document clustering. The CSE (Citation Semantics Extraction) model which involves reference clustering and labeling was explained. Two approaches – CS-VS (combining Citation Semantics and Vector Space measure) and CS2CS (from Citation Semantics to Cluster Semantics) were discussed and evaluated. Our experimental results showed that both could improve the quality of document clustering over traditional document clustering algorithms such as K-Means and K-Medoids. Furthermore, CS2CS as a linear (or nearly linear with splitting and merging) clustering algorithm, is also faster than many traditional document clustering algorithms. A brief comparison between CS-VS and CS2CS is shown in Chapter 3. For convenience, we copy that table here again (Table 45).

In CS-VS, when calculating similarity of two documents, we use both the similarity between vectors of two documents and the similarity between the citation semantics of these documents. That is, we calculate these two kinds of similarity separately, then combine them together through either harmonic mean or simple addition. Then use this

Table 45: Comparison Between Approaches of Citonomy: CS-VS and CS2CS

	CS-VS	**CS2CS**
Highlight	Similarity between Citation Semantics	3-Level Feature Selection
Model of Documents	VSM + Citation Semantics + Title + Keywords + Co-citation	Feature Vector (formed from VSM + Citation Semantics + Title + Keywords)
Similarity measure	Combined VSM similarity and semantics similarity	Similarity between feature vectors
Document Clustering	K-Medoids clustering, static, the number of clusters is predefined	CS2CS linear clustering, dynamic, the number of clusters changes, real time clustering
Use of training set	Use evolution strategy on training set to get weights in combining similarities	Get initial cluster feature vectors from training set
Accuracy compared to traditional K-Medoids and K-Means clustering	Improved more than 5% on average	Improved more than 10% on average
Runtime complexity in terms of the number of documents n	$O(n^2)$	$O(n)$ or $O(n\log n)$ with splitting and merging

measure to do K-Medoids clustering. Note, we also consider the similarity between titles and take into account the information of co-citation. Because of the process of computing the extra similarities, especially the similarity between citation semantics, CS-VS is a little slower than K-Medoids without using these similarities, but they have same runtime complexity in terms of the number of documents.

In CS2CS, a 3-level feature selection with a 2-dimensional normalization is introduced to utilize citation semantics in document clustering. That is, we form feature vectors for single documents and clusters by selecting features for reference clusters (level 1), single documents (level 2), and document clusters (level 3). Then we do document clustering by finding the similarities between document feature vectors and cluster feature vectors. Since the runtime of CS2CS clustering is linear in terms of the number of documents, it is much faster than K-Medoids clustering. If we do splitting and merging in CS2CS clustering whenever the total number of documents is doubled, its runtime complexity would be $O(nlogn)$ which is still faster than CS-VS. And with splitting and

merging, CS2CS can determine the number of clusters dynamically, do realtime clustering over evolving dataset of documents. Moreover, since the 3-level feature selection process effectively selects important terms and removes noise, the quality of the resulted clusters is much higher than that resulted from traditional document clustering algorithms and CS-VS. It even performed better than the traditional algorithms without using the semantics information of documents. In other words, CS2CS is not limited to scientific documents.

We also investigated the use of ontologies in document clustering and CS2CS based fuzzy clustering. The experimental results on both proposed solutions were also promising.

8.1.2 Future Work

Citonomy is used to explore the idea that by correctly utilizing the hidden information in documents, one can improve the quality of document clustering. Our experiments on scientific documents verified our assumption and approaches. The same idea could also be applied to online documents where not only the titles, references, and keywords could be utilized, but the hyper-links that serve for the similar purpose as references, could also be utilized as well. For example, in wikipedia (www.wikipedia.org), the users can create articles and save them to predefined categories. However, choosing the category is subjective and mistake is unavoidable. If CS2CS could be used to find the best matches for the users, the system could prompt the users to choose more appropriate categories.

Similarly, the idea of CS2CS can also be used in scientific document search engines. One can form a feature vector from the user query sentence, and compare it to the feature vectors of existing categories. Since it avoids searching for all the documents, the response to query would be faster.

We discussed fuzzy clustering in this dissertation and presented the algorithm using similar process of CS2CS. We also showed some experimental results. However, more work need to do to fully investigate the advantages and overall performance of using CS2CS to do fuzzy clustering. A hard part of research on fuzzy clustering is the evaluation. It is hard to find collections which have be fuzzy-clustered and hence, it is difficult to (automatically) evaluate the quality of the results of the fuzzy clustering.

8.2 InterOBO Summary and Future Work

Ideally, one would like to relate all ontologies in a domain of discourse to a central reference ontology. The latter refers to an upper level ontology that would serve as a semantic anchor for all ontologies in a domain. However, even if there was general agreement on what would constitute a central reference ontology (”ontology of ontologies”), the cost and constraint of relating current and future ontologies to a reference ontology renders such an approach impractical. The pragmatic alternative is to maintain pairwise mappings between ontologies. While this may lack the semantic clarity of having an overarching upper level ontology, it is a feasible approach. Sub-domain-specific ontologies may be developed by different teams of domain experts in parallel. As the workload

is distributed, this keeps the task of creating ontologies on pace with the growth of knowledge. The disadvantage is that, in principle, the mapping of a new ontology (or new concepts) to m existing ontologies requires m comparisons. However, the actual work of maintenance can be reduced if the new ontology is added to a pre-existing network of ontologies. Higher the degree of redundancy or overlap among existing ontologies, the lower the amount of work required to incorporate the new ontology.

The main motivation in creating a mapping between various ontologies is to facilitate searches of annotated data. Given a query for a data item (sequence, structure or some other biological item), the retrieved data D_i might be explicitly annotated with a term T_i from ontology O_i. However, if there exists a mapping from term T_i to term T_j in ontology O_j, then some D_j annotated with term T_j may also be relevant to the query. Similarly, searches for ontology term T_i can be extended to all synonymous T_j and the associated annotated data retrieved. This would facilitate virtual integration of search space without the need to create a centralized data warehouse of the entire set of annotated data. The clustering of ontologies can be useful as a guide to the extent to which a given search should be broadened. A cluster boundary can serve as a pragmatic search space delimiter for maximizing recall with minimal loss of precision. Given a search that maps explicitly to an ontology within a cluster, it makes intuitive sense to extend it to other ontologies within the same cluster. In terms of parallel implementations, exhaustive searches could be implemented by maintaining separate indices for each cluster on physically distinct nodes. This would prevent duplication of searches and also allow the maintenance of efficient indices of minimal size.

We have presented a scheme for extrapolating concept and edge level synonym matches to mapping at the level of ontologies, and applied MCL to the OBO ontologies to obtain ontology clusters. The future work would be to apply this framework to other domains where there are multiple ontologies available and to transform the InterOBO prototype into a real world application.

APPENDIX

Table A.1: Words of the Cluster Feature Vector of Cluster Blood Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
leaflet	0.0590			
scri	0.0590			
ffh	0.0590			
rot	0.0590	C22.394	Foot Rot	Animal Diseases
atrosepticum	0.0590			
reca	0.0590			
stably	0.0590			
gyra	0.0590			
bestkeeper	0.0590			
topa	0.0590			
housekeeping	0.0590	N02.278.354.422.412	Housekeeping, Hospital	Health Care Facilities, Manpower, and Services
toth	0.0590			
tsx	0.0590			
pectobacterium	0.0590	B03.440.450.425.585	Pectobacterium	Bacteria
glna	0.0590			
nsv	0.0581			
melo	0.0581	B01.650.388.100.300.188.444	Cucumis melo	Eukaryota
mnsv	0.0581			
eif	0.0581	D08.811.913.696.620.682.700.300	eIF-2 Kinase	Enzymes and Coenzymes
aranda	0.0581			
melon	0.0581			
moriones	0.0581			
cvyv	0.0581			
zeyheri	0.0581			
cucurbit	0.0581			
nieto	0.0581			
ecotiling	0.0581			
atfkbp	0.0572			
frb	0.0572			
scfkbp	0.0572			

Table A.2: Words of the Cluster Feature Vector of Cluster Blood Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
suramin	0.0710	D02.455.426.559.847.638.555.750	Suramin	Organic Chemicals
tdc	0.0710			
roseus	0.0710			
phosphotyrosine	0.0710	D12.125.072.050.875.750	Phosphotyrosine	Amino Acids, Peptides, and Proteins
catharanthine	0.0710			
egta	0.0710			
mbpk	0.0710			
cdpk	0.0710			
atfkbp	0.0698			
raptor	0.0698	B01.050.150.900.248.815	Raptors	Eukaryota
frb	0.0698			
polysome	0.0698			
scfkbp	0.0698			
ternary	0.0698	D12.776.260.665.600	Ternary Complex Factors	Amino Acids, Peptides, and Proteins
attor	0.0698			
fkbp	0.0698			
cyclodextrin	0.0685	D04.345.103	Cyclodextrins	Polycyclic Compounds
taxane	0.0685			
guanidine	0.0685	D02.078.370	Guanidines	Organic Chemicals
hypergravity	0.0685	G01.595.060.535.369.300	Hypergravity	Physical Phenomena
taxol	0.0685			
gravity	0.0685	E07.440	Gravity Suits	Equipment and Supplies
urea	0.0685	C10.228.140.163.100.937	Urea Cycle Disorders, Inborn	Nervous System Diseases
baccatin	0.0685			
guanidino	0.0685			
durzan	0.0685			
ventimiglia	0.0685			
citrulline	0.0685	D12.125.095.226	Citrulline	Amino Acids, Peptides, and Proteins
busulfan	0.0673	D02.033.455.125.125	Busulfan	Organic Chemicals
aplasia	0.0673	C15.378.071.750	Red-Cell Aplasia, Pure	Hemic and Lymphatic Diseases

Table A.3: Words of the Cluster Feature Vector of Cluster BrainFunc Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
panl	0.0520			
cnlt	0.0520			
agronomic	0.0520			
issrb	0.0520			
murri	0.0520			
trotter	0.0520			
pilosa	0.0520			
masl	0.0520			
kaye	0.0520			
tefera	0.0520			
crush	0.0520	C21.866.797.240	Crush Syndrome	Disorders of Environmental Origin
rufipogon	0.0520			
dzbs	0.0520			
dzls	0.0520			
ril	0.0520			
ethiopia	0.0520	Z01.058.290.120.310	Ethiopia	Geographic Locations
rpr	0.0520			
pswt	0.0520			
issr	0.0520			
issra	0.0520			
lodg	0.0520			
dia	0.0520			
eragrostis	0.0520	B01.650.388.100.822.355	Eragrostis	Eukaryota
agro	0.0520			
ninter	0.0520			
rehearse	0.0511			
ietswaart	0.0511			
meinzer	0.0507			
konstanz	0.0507			
neologism	0.0507			

Table A.4: Words of the Cluster Feature Vector of of Cluster BrainFunc Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
cnlt	0.0620			
pswt	0.0620			
lodg	0.0620			
murri	0.0620			
pedl	0.0620			
ril	0.0620			
pilosa	0.0620			
ethiopia	0.0620	Z01.058.290.120.310	Ethiopia	Geographic Locations
eragrostis	0.0620	B01.650.388.100.822.355	Eragrostis	Eukaryota
agro	0.0620			
crush	0.0620	C21.866.797.240	Crush Syndrome	Disorders of Environmental Origin
pwt	0.0620			
rpr	0.0620			
issra	0.0620			
issrb	0.0620			
agronomic	0.0620			
fss	0.0606			
daphn	0.0606	B01.650.388.100.932.500	Daphne	Eukaryota
rao	0.0606			
vas	0.0606			
mfi	0.0606			
analogue	0.0606			
neologism	0.0601			
precentral	0.0601			
paraphasia	0.0601			
intergenerational	0.0591	F01.829.263.370.110	Intergenerational Relations	Behavior and Behavior Mechanisms
kindred	0.0591			
spinocerebellar	0.0591	A08.612.220.725	Spinocerebellar Tracts	Nervous System
farrer	0.0591			
poorkaj	0.0591			

Table A.5: Words of the Cluster Feature Vector of Cluster Cardiovasc Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
unguided	0.0655			
yepes	0.0648			
neuroserpin	0.0648			
som	0.0648			
precondition	0.0641	E02.592	Ischemic Preconditioning	Therapeutics
hyperglycemia	0.0641	C18.452.394.952	Hyperglycemia	Nutritional and Metabolic Diseases
fagan	0.0641			
mcao	0.0641			
ergul	0.0641			
tortuosity	0.0641			
grosset	0.0634			
pdq	0.0634			
pulsatile	0.0634	G01.595.560.620	Pulsatile Flow	Physical Phenomena
antiparkinson	0.0634	D27.505.954.427.090.050	Antiparkinson Agents	Chemical Actions and Uses
mannac	0.0627			
sialylated	0.0627			
acetylmannosamine	0.0627			
ncam	0.0627			
hibm	0.0627			
acetylglucosamine	0.0627	D03.383.742.686.850.600.677.120	Uridine Diphosphate N-Acetylglucosamine	Heterocyclic Compounds
Quadriceps	0.0627	A02.633.567.850	Quadriceps Muscle	Musculoskeletal System
epimerase	0.0627	D08.811.399.894	Racemases and Epimerases	Enzymes and Coenzymes
sialic	0.0627	C10.228.140.163.100.435.810	Sialic Acid Storage Disease	Nervous System Diseases
gne	0.0627			
dystroglycan	0.0627	D12.776.210.500.410.500	Dystroglycans	Amino Acids, Peptides, and Proteins
oman	0.0620	Z01.252.245.500.600	Oman	Geographic Locations
omani	0.0620			
pandian	0.0620			
shafaee	0.0620			
sultan	0.0620			

Table A.6: Words of the Cluster Feature Vector of Cluster Cardiovasc Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
leap	0.0746			
ltp	0.0746			
neuroserpin	0.0732			
capsule	0.0725	A02.835.583.443	Joint Capsule	Musculoskeletal System
doctor	0.0718			
ssmc	0.0718			
cbt	0.0718			
apt	0.0718			
ergul	0.0711			
tortuosity	0.0711			
mcao	0.0711			
precondition	0.0711	E02.592	Ischemic Preconditioning	Therapeutics
dysarthria	0.0711	C10.597.606.150.500.800.150.200	Dysarthria	Nervous System Diseases
pdq	0.0704			
pulsatile	0.0704	G01.595.560.620	Pulsatile Flow	Physical Phenomena
grosset	0.0704			
pill	0.0704			
antiparkinson	0.0704	D27.505.954.427.090.050	Antiparkinson Agents	Chemical Actions and Uses
beyond	0.0697			
bogoslovsky	0.0697			
salvage	0.0697	E02.186.800	Salvage Therapy	Therapeutics
penumbra	0.0697			
oman	0.0690	Z01.252.245.500.600	Oman	Geographic Locations
omani	0.0690			
sultan	0.0690			
warn	0.0690	F01.145.209.259.800.200	Duty to Warn	Behavior and Behavior Mechanisms
margarita	0.0684			
nedices	0.0684			
pop	0.0684			
pamplona	0.0684			

Table A.7: Words of the Cluster Feature Vector of Cluster Cough Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
pungency	0.0677			
gustatory	0.0677	C10.177.825	Sweating, Gustatory	Nervous System Diseases
chemesthesis	0.0677			
pepper	0.0677	J02.500.250.725.500	Black Pepper	Food and Beverages
tohoku	0.0677			
yazawa	0.0677			
capsinoid	0.0677			
geriat	0.0677			
capsiate	0.0677			
pungent	0.0677			
codeine	0.0660	D03.132.577.249.547.547.149	Codeine	Heterocyclic Compounds
takahama	0.0660			
citric	0.0660	D02.241.081.901.434.249	Citric Acid	Organic Chemicals
kamei	0.0660			
narcotic	0.0660	D27.505.696.277.600	Narcotics	Chemical Actions and Uses
tractus	0.0660			
opiate	0.0660	D03.132.577	Opiate Alkaloids	Heterocyclic Compounds
cholinergic	0.0660	A08.663.542.234	Cholinergic Fibers	Nervous System
snore	0.0643	C23.888.852.779.850	Snoring	Pathological Conditions, Signs and Symptoms
apnoea	0.0643			
surinder	0.0643			
strachan	0.0625			
indoor	0.0625	N06.850.460.100.080	Air Pollution, Indoor	Environment and Public Health
kloft	0.0625			
charit	0.0625			
groneberg	0.0625			
dinh	0.0625			
fischer	0.0625			
audience	0.0608			
broadcast	0.0608			

Table A.8: Words of the Cluster Feature Vector of Cluster Cough Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
pepper	0.0734	J02.500.250.725.500	Black Pepper	Food and Beverages
tohoku	0.0734			
chemesthesis	0.0734			
capsiate	0.0734			
gustatory	0.0734	C10.177.825	Sweating, Gustatory	Nervous System Diseases
capsinoid	0.0734			
pungent	0.0734			
cholinergic	0.0713	A08.663.542.234	Cholinergic Fibers	Nervous System
takahama	0.0713			
citric	0.0713	D02.241.081.901.434.249	Citric Acid	Organic Chemicals
codeine	0.0713	D03.132.577.249.547.547.149	Codeine	Heterocyclic Compounds
narcotic	0.0713	D27.505.696.277.600	Narcotics	Chemical Actions and Uses
opiate	0.0713	D03.132.577	Opiate Alkaloids	Heterocyclic Compounds
snore	0.0692	C23.888.852.779.850	Snoring	Pathological Conditions, Signs and Symptoms
apnoea	0.0692			
lethargy	0.0692	C10.597.606.441	Lethargy	Nervous System Diseases
ther	0.0671			
indoor	0.0671	N06.850.460.100.080	Air Pollution, Indoor	Environment and Public Health
pulm	0.0671			
strachan	0.0671			
groneberg	0.0671			
pupt	0.0671			
cook	0.0671	J01.494.300	Cooking and Eating Utensils	Technology, Industry, and Agriculture
radio	0.0650	D01.496.448.496.665	Serum Albumin, Radio-Iodinated	Inorganic Chemicals
broadcast	0.0650			
manometry	0.0629	E05.559	Manometry	Investigative Techniques
huisman	0.0608			
antitussive	0.0605	D27.505.954.427.153	Antitussive Agents	Chemical Actions and Uses
beraprost	0.0587			
mite	0.0587	B01.050.500.131.166.132.419	Mites	Eukaryota

Table A.9: Words of the Cluster Feature Vector of Cluster EndocrDisord Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
horstman	0.0640			
fvii	0.0640			
miami	0.0640			
minagar	0.0640			
acl	0.0640			
phosphatidylserine	0.0640	D10.570.755.375.760.400.971	Phosphatidylserines	Lipids
apla	0.0640			
jimenez	0.0640			
gpi	0.0640			
cardiolipin	0.0640	D10.570.755.375.760.400.885.185	Cardiolipins	Lipids
ahn	0.0640			
bidot	0.0640			
wmw	0.0626			
horiuchi	0.0626			
rage	0.0626	F01.470.093.640	Rage	Behavior and Behavior Mechanisms
carboxymethyl	0.0626			
optima	0.0626			
camcog	0.0626			
epicentre	0.0612			
immunopositive	0.0612			
gfap	0.0612			
timp	0.0612			
oval	0.0612	A07.541.459.500	Foramen Ovale	Cardiovascular System
jnnp	0.0599			
chabardes	0.0599			
vesper	0.0599			
subthalamic	0.0599	A08.186.211.730.317.800.800	Subthalamic Nucleus	Nervous System
pollak	0.0599			
pallidal	0.0599			
stereotact	0.0599			

Table A.10: Words of the Cluster Feature Vector of Cluster EndocrDisord Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
hexose	0.0740	D08.811.913.696.445.850	UDPglucose-Hexose-1-Phosphate Uridylyltransferase	Enzymes and Coenzymes
radiolabel	0.0740			
moiety	0.0740			
path	0.0740			
apoplasm	0.0740			
sugarcane	0.0740			
recover	0.0740			
sorghum	0.0740	B01.650.388.100.822.894	Sorghum	Eukaryota
japonicum	0.0727	B01.050.500.500.736.715.770.680.570	Schistosoma japonicum	Eukaryota
meliloti	0.0727	B03.440.400.425.700.887.500	Sinorhizobium meliloti	Bacteria
indol	0.0727	D03.132.436	Indole Alkaloids	Heterocyclic Compounds
vulgaris	0.0727	B01.040.080.469.400	Chlorella vulgaris	Eukaryota
overproduce	0.0727			
rhizobia	0.0727			
rhizobium	0.0727	B03.440.400.425.700.800	Rhizobium	Bacteria
indeterminate	0.0727			
pin	0.0727	E06.292	Dental Pins	Dentistry
iaamtms	0.0727			
operon	0.0727	G05.360.340.024.686	Operon	Genetic Phenomena
rhp	0.0713			
arid	0.0713			
nine	0.0713			
g_iv	0.0713			
isf	0.0713			
baydar	0.0713			
esselink	0.0713			
g_i	0.0713			
damascena	0.0713	B01.650.388.100.838.518.500	Nigella damascena	Eukaryota
vosman	0.0713			
damask	0.0713			

Table A.11: Words of the Cluster Feature Vector of Cluster Neurol Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
ambul	0.0561	E02.831.335	Early Ambulation	Therapeutics
leap	0.0561			
ltp	0.0561			
overground	0.0561			
homocarnosine	0.0556			
carnosine	0.0556	D12.644.400.100	Carnosine	Amino Acids, Peptides, and Proteins
balion	0.0556			
carnosinase	0.0556			
tatsch	0.0550			
pirker	0.0550			
oertel	0.0550			
ibzm	0.0550			
normalcy	0.0550			
radiotracer	0.0550			
booij	0.0550			
lokkegaard	0.0550			
schwarz	0.0550			
asenbaum	0.0550			
tracer	0.0550	D01.496.749.731	Radioactive Tracers	Inorganic Chemicals
hed	0.0544			
migraineurs	0.0544			
tth	0.0544			
westgaard	0.0544			
uir	0.0544			
leistad	0.0544			
treadmill	0.0541			
immuno	0.0539			
sudanese	0.0539			
kuwaiti	0.0539			
whoqol	0.0539			

Table A.12: Words of the Cluster Feature Vector of Cluster Neurol Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
carnosine	0.0672	D12.644.400.100	Carnosine	Amino Acids, Peptides, and Proteins
carnosinase	0.0672			
pirker	0.0665			
schwarz	0.0665			
nucl	0.0665			
ibzm	0.0665			
radiotracer	0.0665			
tracer	0.0665	D01.496.749.731	Radioactive Tracers	Inorganic Chemicals
migraineurs	0.0658			
tth	0.0658			
westgaard	0.0658			
kuwaiti	0.0651			
whoqol	0.0651			
facet	0.0651			
bref	0.0651			
spiritual	0.0651	E02.190.901	Spiritual Therapies	Therapeutics
cit	0.0650			
vlaar	0.0650			
worsen	0.0644			
meaningful	0.0644			
cholinesterase	0.0644	D08.811.277.352.100.170	Cholinesterases	Enzymes and Coenzymes
cibic	0.0644			
donepezil	0.0644			
smell	0.0636	F02.830.816.643	Smell	Psychological Phenomena and Processes
becker	0.0636			
maastricht	0.0636			
weber	0.0636	C04.557.645.375.850	Sturge-Weber Syndrome	Neoplasms
azm	0.0636			
overground	0.0629			
hars	0.0629			

Table A.13: Words of the Cluster Feature Vector of Cluster Plant Mapped to MeSH Terms

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
iridaceae	0.0401	B01.650.388.100.549	Iridaceae	Eukaryota
agostino	0.0401			
sacl	0.0401			
camara	0.0401			
crocus	0.0401	B01.650.388.100.549.500	Crocus	Eukaryota
glucosyltransferase	0.0401	D08.811.913.400.450.460	Glucosyltransferases	Enzymes and Coenzymes
saffron	0.0401			
spice	0.0401	J02.500.250.725	Spices	Food and Beverages
crocetin	0.0401			
panax	0.0398	B01.650.388.100.087.500	Panax	Eukaryota
subgenus	0.0398			
constraint	0.0398			
nonphotosynthetic	0.0398			
ipomoea	0.0398	B01.650.388.100.238.500	Ipomoea	Eukaryota
convolvulaceae	0.0398	B01.650.388.100.238	Convolvulaceae	Eukaryota
obtusiflora	0.0398			
pseudogene	0.0398	G05.360.340.024.340.700	Pseudogenes	Genetic Phenomena
ndh	0.0398			
exaltata	0.0398			
memelink	0.0396			
egta	0.0396			
catharanthine	0.0396			
mbpk	0.0396			
cdpk	0.0396			
tdc	0.0396			
suramin	0.0396	D02.455.426.559.847.638.555.750	Suramin	Organic Chemicals
hple	0.0394			
hplf	0.0394			
nile	0.0394	B04.820.250.350.300.950	West Nile virus	Viruses
aldehyde	0.0394	D02.047	Aldehydes	Organic Chemicals

Table A.14: Words of the Cluster Feature Vector of Cluster Plant Mapped to MeSH Terms (MeSH Considered in Forming Document Feature Vectors)

Label	Weight	MeSH ID	MeSH Term	Root MeSH Term
spice	0.0469	J02.500.250.725	Spices	Food and Beverages
saffron	0.0469			
iridaceae	0.0469	B01.650.388.100.549	Iridaceae	Eukaryota
sacl	0.0469			
crocus	0.0469	B01.650.388.100.549.500	Crocus	Eukaryota
glucosyltransferase	0.0469	D08.811.913.400.450.460	Glucosyltransferases	Enzymes and Coenzymes
sativus	0.0469	B01.650.388.100.300.188.666	Cucumis sativus	Eukaryota
obtusiflora	0.0466			
autotroph	0.0466	G02.111.087.070	Autotrophic Processes	Chemical Phenomena
nonphotosynthetic	0.0466			
constraint	0.0466			
ipomoea	0.0466	B01.650.388.100.238.500	Ipomoea	Eukaryota
convolvulaceae	0.0466	B01.650.388.100.238	Convolvulaceae	Eukaryota
panax	0.0466	B01.650.388.100.087.500	Panax	Eukaryota
ndh	0.0466			
exaltata	0.0466			
hpl	0.0464			
localise	0.0464			
rfp	0.0464			
hple	0.0464			
hplf	0.0464			
nile	0.0464	B04.820.250.350.300.950	West Nile virus	Viruses
detergent	0.0464	D27.720.877.265	Detergents	Chemical Actions and Uses
hydroperoxide	0.0464			
aldehyde	0.0464	D02.047	Aldehydes	Organic Chemicals
micelle	0.0464	D05.374	Micelles	Macromolecular Substances
rpp	0.0461			
tir	0.0461			
mpss	0.0461			
poptrarf	0.0458			

REFERENCE LIST

[1] Blaise Cronin's home page. http://www.slis.indiana.edu/faculty/cronin/. Accessed on May 10, 2010.

[2] Chemical Entities of Biological Interest. http://www.ebi.ac.uk/chebi/. Accessed on May 10, 2010.

[3] CiteSeer. http://citeseer.ist.psu.edu/. Accessed on May 10, 2010.

[4] CiteSeerX. http://citeseerx.ist.psu.edu/. Accessed on May 10, 2010.

[5] DAML+OIL. http://www.daml.org/2001/03/daml+oil-index. Accessed on September 15, 2010.

[6] Dublin Core. http://dublincore.org/. Accessed on September 15, 2010.

[7] Eugene Garfield's home page. http://www.garfield.library.upenn.edu/. Accessed on May 10, 2010.

[8] FaCT (Fast Classification of Terminologies) System. http://www.cs.man.ac.uk/~horrocks/FaCT/. Accessed on May 10, 2010.

[9] Facts of Shepherd's Citations. http://www.usps.com/judicial/1974deci/1-88.htm. Accessed on May 10, 2010.

[10] Foundational Model of Anatomy. http://sig.biostr.washington.edu/projects/fm/AboutFM.html. Accessed on May 10, 2010.

[11] Friend of a Friend. http://ebiquity.umbc.edu/resource/html/id/82/. Accessed on September 15, 2010.

[12] GenBank. http://www.ncbi.nlm.nih.gov/genbank/. Accessed on May 10, 2010.

[13] The Gene Ontology. http://www.geneontology.org/. Accessed on May 10, 2010.

[14] Google Scholar. http://scholar.google.com/. Accessed on May 10, 2010.

[15] Henk Moed's home page. http://www.cwts.nl/hm/. Accessed on May 10, 2010.

[16] Journal Citation Reports. http://www.thomsonreuters.com/products_services/scientific/Journal_Citation_Reports. Accessed on May 10, 2010.

[17] Journals currently indexed in MEDLINE. http://www.nlm.nih.gov/tsd/serials/lji.html. Accessed on May 10, 2010.

[18] Medical Subject Headings. http://www.nlm.nih.gov/mesh/. Accessed on May 10, 2010.

[19] MEDLINE vs. PubMed. http://www.nlm.nih.gov/pubs/factsheets/dif_med_pub.html. Accessed on May 10, 2010.

[20] Nature Physics Portal. http://www.nature.com/physics/archive/index.html. Accessed on May 10, 2010.

[21] The Open Biological and Biomedical Ontologies. http://www.obofoundry.org/. Accessed on May 10, 2010.

[22] OWL Web Ontology Language. http://www.w3.org/TR/owl-guide/. Accessed on September 15, 2010.

[23] Pajek. http://vlado.fmf.uni-lj.si/pub/networks/pajek/. Accessed on May 10, 2010.

[24] PubMed. http://www.ncbi.nlm.nih.gov/pubmed/. Accessed on May 10, 2010.

[25] PubMed Central. http://www.pubmedcentral.nih.gov/about/ftp.html. Accessed on May 10, 2010.

[26] Racer (Renamed Abox and Concept Expression Reasoner). http://www.sts.tu-harburg.de/~r.f.moeller/racer/. Accessed on May 10, 2010.

[27] RDF Schema. http://www.w3.org/TR/rdf-schema/. Accessed on September 15, 2010.

[28] Research Papers in Economics. http://repec.org/. Accessed on May 10, 2010.

[29] Resource Description Framework. http://www.w3.org/TR/PR-rdf-syntax/. Accessed on September 15, 2010.

[30] Scopus. http://www.info.scopus.com/. Accessed on May 10, 2010.

[31] Shepherd's Citations Service Information. http://law.lexisnexis.com/shepards. Accessed on May 10, 2010.

[32] SNOMED CT(Systematized Nomenclature of Medicine-Clinical Terms). http://www.ihtsdo.org/snomed-ct/. Accessed on May 10, 2010.

[33] Thomas Scientific. http://www.thomsonreuters.com/business_units/scientific/. Accessed on May 10, 2010.

[34] Web of Science. http://thomsonreuters.com/products_services/scientific/Web_of_Science. Accessed on May 10, 2010.

[35] Beil, F., Ester, M., and Xu, X. Frequent term-based text clustering. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (2002).

[36] Bolelli, L., Ertekin, S., and Giles, C. *Knowledge Discovery in Databases: PKDD 2006*. Springer, 2006, ch. Clustering Scientific Literature Using Sparse Citation Graph Analysis.

[37] Cameron, R. A Universal Citation Database as a Catalyst for Reform in Scholarly Communication. First Monday 2,4 (1997), 1396.

[38] Chen, C. CiteSpace II: Detecting and Visualizing Emerging Trends and Transient Patterns in Scientific Literature. Journal of the American Society for Information Science and Technology 5,7 (2006), 359–377.

[39] Chen, C., Song, I., Yuan, X., and Zhang, J. The Thematic and Citation Landscape of Data and Knowledge Engineering (1985-2007). Data & Knowledge Engineering 67 (2008), 234–259.

[40] Chen, C., Song, I., and Zhu, W. Trends in conceptual modeling: Citation analysis of the ER conference papers (1979-2005). In Proceedings of the 11th International Conference on the International Society for Scientometrics and Informatrics (2007), pp. 189–200.

[41] Cronin, B. The Need for a Theory of Citing. Journal of Documentation 37 (1981), 16–24.

[42] Cronin, B., and K., O. Citation-based Auditing of Academic Performance. Journal of the American Society for Information Science 45,2 (1994), 61–72.

[43] Cronin, B., and Snyder, H. Comparative Citation Rankings of Authors in Monographic and Journal Literature: A Study of Sociology. Journal of Documentation 53,3 (1997), 263–273.

[44] Dash, M., Choi, K., Scheuermann, P., and Liu, H. Feature Selection for Clustering - A Filter Solution. In Second IEEE International Conference on Data Mining (ICDM'02) (2002).

[45] Day, M., Tsai, T., Sung, C., Lee, C., Wu, S., Ong, C., and Hsu, W. A Knowledge-based Approach to Citation Extraction. In Information Reuse and Integration, 2005 IEEE International Conference on (2005), pp. 189–200.

[46] Dinakarpandian, D., Tong, T., and Lee, Y. A pragmatic approach to mapping the open biomedical ontologies. International Journal of Bioinformatics Research and Applications 3,3 (2007), 341–365.

[47] Dongen, S. Graph Clustering by flow simulation. `http://www.micans.org/mcl/`, 2000. Dissertation.

[48] Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., and Harshman, R. Using latent semantic analysis to improve access to textual information. In

Proceedings of the SIGCHI conference on Human factors in computing systems (1988).

[49] Forgy, E. Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classification. Biometrics 21 (1965), 768–780.

[50] Forsythe, G. E., Malcolm, M. A., and B., M. C. *Computer methods for mathematical computation*. Prentice Hall, 1977, ch. Least squares and the singular value decomposition.

[51] Garfield, E. Citation Indexes for Science. Science, New Series 122,3159 (1955), 108–111.

[52] Garfield, E. Long-Term Vs. Short-Term Journal Impact: Does It Matter? The Physiologist 41,3 (1998), 113–115.

[53] Garfield, E., and Sher, I. New Factors in the Evaluation of Scientific Literature Through Citation Indexing. American Documentation 14,3 (2007), 195–201.

[54] Garfield, E., Sher, I., and Torpie, R. The Use of Citation Data in Writing the History of Science. Institute for Scientific Information, 1964.

[55] Giles, C., Bollacker, K., and Lawrence, S. CiteSeer: An automatic citation indexing system. In Digital Libraries 98 - The Third ACM Conference on Digital Libraries (1998), pp. 89–98.

[56] Gruber, T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies 43,4-5 (1995), 907–928.

[57] Han, H., Giles, C., Zha, H., Li, C., and Tsioutsiouliklis, K. Two Supervised Learning Approach for Name Disambiguation in Author Citations. In Proceedings of the 2004 joint conference on digital libraries (2004).

[58] Han, H., Zha, H., and Giles, C. Name Disambiguation in Author Citations using a K-Way Spectral Clustering Method. In Proceedings of the 2005 joint conference on digital libraries (2005).

[59] Hartigan, J. A. Clustering Algorithms. Wiley, 1975.

[60] Hertz, J., Krogh, A., and Palmer, R. Introduction to the Theory of Neurol Computation. Addison-Wesley Longman, 1991.

[61] J., M. An Examination of Citation Index. Aslib Proceedings 17,6 (1965), 184–196.

[62] Jacso, P. As we may search - Comparison of major features of the Web of Science, Scopus, and Google Scholar citation-based and citation-enhanced databases. Current Science 89,9 (2005).

[63] Jain, A. K., and Dubes, R. C. Algorithms for Clustering Data. Prentice-Hall, Inc., 1988.

[64] Kaufman, L., and Rousseeuw, P. J. Finding Groups in Data. An Introduction to Cluster Analysis. John Wiley & Sons, 1990.

[65] King, B. Step-wise Clustering Procedure. Journal of the American Statistical Association 62,317 (1967), 86–101.

[66] Kohonen, T. Self-Organization and Associative Memory. Springer-Verlag, 1989.

[67] Kostoff, R. The Use and Misuse of Citation Analysis in Research Evaluation. Scientometrics 43,1 (1998), 27–43.

[68] Krichel, T. Working towards an Open Library for Economics: The RePEc project. `http://openlib.org/home/krichel/myers.html`, 2000.

[69] Lambrix, P. Towards a semantic web for bioinformatics using ontology-based annotation. In 14th IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (2005), pp. 3–7.

[70] Larsen, B., and Aone, C. Fast and effective text mining using linear-time document clustering. In Conference on Knowledge Discovery and Data Mining (1999).

[71] Lawrence, S., Giles, C., and Bollacker, K. Digital libraries and autonomous citation indexing. IEEE Computer 32,6 (1999), 67–71.

[72] Liu, T., Liu, S., Chen, Z., and Ma, W. An Evaluation on Feature Selection for Text Clustering. In Proceedings of the Twentieth International Conference on Machine

Learing (ICML-2003) (2003).

[73] MacRoberts, M., and MacRoberts, B. Problems of Citation Analysis. Scientometrics 36,3 (1996), 435–444.

[74] Moed, H. Bibliometric Measurement of Research Performance and Price's Theory of Differences among the Sciences. Scientometrics 15,5-6 (1989), 473–483.

[75] Moed, H. Bibliometric Indicators Reflect Publication and Management Strategies. Scientometrics 47,2 (2000), 323–326.

[76] Moed, H., Burger, W., Frankford, J., and Van Raan, A. The Application of Bibliometrics Indicators: Important Field and Time Dependent Factors to Be Considered. Scientometrics 8,3-4 (1985), 177–203.

[77] Moed, H., Burger, W., Frankford, J., and Van Raan, A. The Use of Bibliometrics Data for the Measurement of University Research Performance. Research Policy 14,3 (1985), 131–149.

[78] Moed, H., Luwel, M., and Nederhof, A. Towards Research Performance in the Humanities. Library Trends 50,3 (2002), 498–520.

[79] Moed, H., Van Leeuwen, T., and Reeduk, J. Towards Appropriate Indicators of Journal Impact. Scientometrics 46,3 (1999), 575–589.

[80] Moravcsik, M., and Murugesan, P. Some Results on the Function and Quality of Citations. Social study of science 5 (1975), 88–91.

[81] Nakov, P., Schwarts, A., and Hearst, M. Citances: Citation Sentences for Semantic Analysis of Bioscience Text. In Proceedings of the SIGIR'04 workshop on Search and Discovery in Bioinformatics (2004).

[82] Noruzi, A. Google Scholar: The New Generation of Citation Indexes. Libri 55 (1975), 170–180.

[83] Person, O. The Intellectual Base and Research Front of JASIS 1986-1990. Journal of the American Society for Information Science and technology 45,1 (1994), 31–38.

[84] Price, D. Networks of Scientific Papers. Science 149,3683 (1965), 510–515.

[85] Rechenberg, I. Cybernetic solution path of an experimental problem. In Royal Aircraft Establishment (1965).

[86] Rector, A., Bechhofer, S., Goble, C., Horrocks, I., Nowlan, W., and Solomon, W. The GRAIL concept modelling language for medical terminology. Artificial Intelligence in Medicine 9 (1997), 139–171.

[87] Redner, S. How Popular Is Your Paper? An Empirical Study of the Citation Distribution. The European Physical Journal B 4,2 (1998), 131–134.

[88] Rijsbergen, C., Robertson, S., and Porter, M. New models in probabilistic information retrieval. `http://opensigle.inist.fr/handle/10068/550946`, 1980. Research and Development Department, British Library: London.

[89] Rijsbergen, C. J. V. Information Retrieval. ButterWorths, London, 1979.

[90] Salton, G., Wong, A., and Yang, C. A vector space model for automatic indexing. Communications of the ACM 18,11 (1975), 613–620.

[91] Saracoglu, R., Tutuncu, K., and Allahverdi, N. A Fuzzy Clustering Approach for Finding Similar Documents Using a Novel Similarity Meansure. Expert Systems with Applications 33 (2007), 600–605.

[92] Schuyler, P. L., Hole, W. T., Tuttle, M. S., and Sherertz, D. D. The UMLS Metathesaurus: Representing different views of biomedical concepts. Bull Med Libr Assoc, 81,2 (1993), 217–222.

[93] Schwefel, H. P. Numerical optimization of computer models. John Wiley & Sons, 1981.

[94] Schwefel, H. P. Evolution and optimum seeking: The Sixth Generation. John Wiley & Sons, 1995.

[95] Sneath, P., and Sokal, R. Numerical Taxonomy: the Principles and Practice of Numerical Classification. Freeman, 1973.

[96] Steinbach, M., Karypis, G., and Kumar, V. A comparison of document clustering

techniques. In KDD Workshop on Text Mining (2000).

[97] Takasu, A. Bibliographic Attribute Extraction from Erroneous References Based on a Statistical Model. In Proceedings of the 2003 joint conference on digital libraries (2003).

[98] Teufel, S., Siddharthan, A., and Tidhar, D. Automatic Classification of Citation Function. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Proceeding (2006), pp. 103–110.

[99] Tong, T., Dinakarpandian, D., and Lee, Y. Literature clustering using citation semantics. In 42nd Hawaii International Conference on System Sciences (2009).

[100] Yang, Y. Noise reduction in a statistical approach to text categorization. In Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (1995), pp. 256–263.

[101] Yoo, I., and Hu, X. A Comprehensive Comparison Study of Document Clustering for a Biomedical Digital Library MEDLINE. In Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries (2006), pp. 220–229.

[102] Zahn, C. T. Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transaction on Computers C-20,1 (1971), 68–86.

[103] Zhang, B., Chen, Y., Fan, W., Fox, E. A., Goncalves, M., Cristo, M., and Calado, P. Intelligent Fusion of Structural and Citation-Based evidence for Text Classification. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (2005).

VITA

Tuanjie Tong graduated in April 1996 with a Master's degree in Automatic Control from Beijing Institute of Technology, Beijing, China.

He then joined Beijing Advanced System Inc., a joint venture of IBM and TsingHua University. After working there for two years as a software engineer, he became the manager of the software department of Beijing Goldkey Inc. which was established by Goldkey Taiwan Inc. He worked there for another two years before attending Western Illinois University (WIU).

From WIU he received two Master's degrees in Mathematics and Computer Science. Tired of pursing Master's degrees, in June 2005, he finally joined the Interdisciplinary Ph.D. program at the University of Missouri–Kansas City (UMKC) with Computer Informatics and Computer Networking as his Coordinating discipline and Co-discipline, respectively. During his Ph.D. study, he has been honored with the School of Graduate Studies Dissertation Research Fellowship (2009-2010), the Chancellor's Doctoral Fellowship (2007-2009),the School of Computing and Engineering Dean's Doctoral Fellowship (2007), the School of Graduate Studies Dean's Doctoral Fellowship (2006), and the School of Computing and Engineering Outstanding Student Award (2007 and 2009).

Upon completion of his degree requirements, he plans on taking an position in either industry or academia where he can apply what he learned to real-world problems and continue to do research in related areas. His major research interests are knowledge management and discovery, particularly in document clustering and ontology management.

He is a student member of the honor societies Phi Kappa Phi and Upsilon Pi Epsilon, Institute of Electrical and Electronic Engineers (IEEE), and Association for Computing Machinery (ACM).