12-9-2006

Clustering Multiple Contextually Related Heterogeneous Datasets

Mahmood Hossain

CLUSTERING MULTIPLE CONTEXTUALLY RELATED HETEROGENEOUS DATASETS

By

Mahmood Hossain

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulf llment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science
in the Department of Computer Science and Engineering

Mississippi State, Mississippi

December 2006

CLUSTERING MULTIPLE CONTEXTUALLY RELATED

HETEROGENEOUS DATASETS

By

Mahmood Hossain

Approved:

Susan M. Bridges Professor of Computer Science and Engineering (Major Professor)	Julia E. Hodges Professor of Computer Science and Engineering, and Department Head (Committee Member)
Eric A. Hansen Associate Professor of Computer Science and Engineering (Committee Member)	David A. Dampier Associate Professor of Computer Science and Engineering (Committee Member)
Edward B. Allen Associate Professor of Computer Science and Engineering, and Graduate Coordinator (Committee Member)	Kirk H. Schulz Dean of Bagley College of Engineering

Name: Mahmood Hossain

Date of Degree: December 8, 2006

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Susan M. Bridges

Title of Study: CLUSTERING MULTIPLE CONTEXTUALLY RELATED HETEROGENEOUS DATASETS

Pages in Study: 149

Candidate for Degree of Doctor of Philosophy

Traditional clustering is typically based on a single feature set. In some domains, several feature sets may be available to represent the same objects, but it may not be easy to compute a useful and effective integrated feature set. We hypothesize that clustering individual datasets and then combining them using a suitable ensemble algorithm will yield better quality clusters compared to the individual clustering or clustering based on an integrated feature set.

We present two classes of algorithms to address the problem of combining the results of clustering obtained from multiple related datasets where the datasets represent identical or overlapping sets of objects but use different feature sets. One class of algorithms was developed for combining hierarchical clustering generated from multiple datasets and another class of algorithms was developed for combining partitional clustering generated from multiple datasets. The first class of algorithms, called EPaCH, are based on graph-

theoretic principles and use the association strengths of objects in the individual cluster hierarchies. The second class of algorithms, called CEMENT, use an EM (Expectation Maximization) approach to progressively ref ne the individual clusterings until the mutual entropy between them converges toward a maximum.

We have applied our methods to the problem of clustering a document collection consisting of journal abstracts from ten different Library of Congress categories. After several natural language preprocessing steps, both syntactic and semantic feature sets were extracted. We present empirical results that include the comparison of our algorithms with several baseline clustering schemes using different cluster validation indices. We also present the results of one-tailed paired *T*-tests performed on cluster qualities. Our methods are shown to yield higher quality clusters than the baseline clustering schemes that include the clustering based on individual feature sets and clustering based on concatenated feature sets. When the sets of objects represented in two datasets are overlapping but not identical, our algorithms outperform all baseline methods for all indices.

ACKNOWLEDGMENTS

Praise be to God, Lord of the Worlds, The Benef cent, the Merciful. I express my sincere gratitude to everyone who helped bring this dissertation to completion. First and foremost, I acknowledge Dr. Susan Bridges, my major professor. I am deeply indebted to her for her continuous guidance, encouragement, and patience throughout my Ph.D. program. I also express my appreciation to other members of my committee, Dr. Julia Hodges, Dr. Eric Hansen, Dr. David Dampier, and Dr. Ed Allen for their suggestions, support, and availability. Special thanks to Dr. Yong Wang for sharing his document pre-processing programs without which it would not have been possible to complete the work by this time. I am grateful to my beloved wife Mimi for her endless emotional support and my mother for giving me the inspiration throughout my life. Finally, I appreciate my wonderful kids Raimah and Rameen who bring joy to my life.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Clustering is a well-studied data mining problem that has found applications in many areas. Cluster analysis is the process of categorizing data into subsets that have meaning in the context of a particular problem. There are different clustering algorithms each of which may or may not be suited to a particular application. These algorithms are based on discrete descriptions of the data and are designed to capture complex relationships between data objects. The objective in any clustering application is to maximize the inter-cluster differences and intra-cluster similarities.

1.1 Motivation

The focus of this dissertation is a set of new algorithms for clustering multiple heterogeneous datasets where the datasets are related but may represent different objects, the datasets may represent different types of objects, and the attributes of the object sets may differ signif cantly. For example, clustering can be used to partition a document collection into well-separated groups based on different types of information derived using different preprocessing techniques. Noun-phrase extraction allows us to generate a clustering of documents based on the syntactic information contained in the data while sense disambiguation allows us to generate a clustering of documents based on semantic information.

It is likely that the results of clustering the documents based on these two types of information will produce different clusterings of the data. Improved clustering would be expected if both types of information are used.

As another example, consider the biology domain where scientists are interested in identifying groups of genes that are expressed in similar patterns and thus may be co-regulated. Expression of genes can be measured using mRNA levels (gene expression data) or using protein expression levels. The methods differ in sensitivity and some genes may be transcribed to mRNA that are not translated to proteins. In addition, changes in protein levels are sometimes controlled using mechanisms other than gene expression. To further complicate matters, both gene expression and protein expression datasets are noisy. A unif ed clustering of these datasets can compensate for the noise and has the potential to reveal new biological insight.

The traditional clustering paradigm pertains to a single dataset. In some problem domains, multiple heterogeneous datasets may be available that can provide complementary information. Clustering these datasets may reveal interesting relationships between objects. Fern and Brodley [19], Hu and Yoo [28], and Strehl and Ghosh [53] have developed *ensemble* methods to combine the results from different clusterings. These algorithms were designed to use a single dataset that is clustered using different algorithms or multiple runs of the same algorithm with different parameters or initial settings. In general, these algorithms are not applicable for combining clustering results generated from differ-

ent datasets. In addition, they typically work only with partitional clustering and cannot be used with popular hierarchical clustering approaches.

Recently, there has been some research in clustering multiple heterogeneous datasets. The problem of clustering heterogeneous data can be solved in two general ways: (1) clustering multiple datasets using an integrated feature space and (2) clustering the datasets individually and then combining the clusters based on some mapping or correlation. Methods employing the first approach (e.g., Dagan, Marx, and Shamir [12], Pavlidis et al. [43]) typically require the heterogeneous data sets to be integrated into a unified feature space. This can cause substantial sparseness in the integrated feature space. Also, there can be significant differences in dimensionality of the datasets or the feature vectors for the different datasets can be of different types (e.g., one having continuous attributes, one having categorical attributes, etc.). The feature space from one dataset might provide more information than the feature space from another dataset and thus the latter may contribute noise and the overall cluster quality can deteriorate. On the other hand, methods that cluster the data sets independently and combine the clusters (e.g., Marx et. al. [39]) may require some kind of similarity computation between pairs of heterogeneous data elements. But, due to the heterogeneity of data across multiple datasets, computing such similarity may not be trivial.

Even though there has been some research in clustering multiple datasets, this new paradigm of clustering is still in its infancy. There is no concrete framework that can be tailored to a specific application. We argue that such a clustering framework should be

based on a mapping between the objects of two datasets. This mapping can be of different types depending on how an object from one dataset is mapped to zero or more objects in another dataset and vice-versa. This will allow us to take care of not only multiple datasets consisting of different aspects of the same objects, but also multiple datasets consisting of different object sets.

1.2 Statement of Hypothesis

For clustering tasks involving multiple contextually related datasets, we introduce an approach that clusters the datasets individually using hierarchical or partitional clustering and combines the resulting sets of clusters using a mapping framework between the datasets. We consider two datasets to provide *complementary information* when a mapping can be established between objects of the datasets and each feature set has relevance in the context of the same clustering task. The hypothesis of this research is that *when multiple datasets provide complementary information, this ensemble approach can yield improved clustering performance over clustering individual data sets. In cases where the data sets differ in the number of objects they contain or the aspects of the objects represented in the feature sets, this approach can produce better results than clustering based on a unif ed feature space*.

1.3 Contributions

This dissertation describes a set of algorithms that have been developed for combining the clustering results obtained from multiple contextually related datasets. In particular, we have developed two families of algorithms. The first family is designed for use with hierarchical clustering and the second one for use with partitional clustering.

Hierarchical clustering [25] is a powerful clustering method that generates a set of nested clusters in the form of a tree. Previous ensemble approaches [19, 28, 53] have been applied to combine partitional clustering results. One focus of this dissertation is the design of an ensemble approach for combining multiple hierarchical clusterings. Tree combination methods [7, 22, 24, 38, 44, 48, 49] have been studied primarily in the context of phylogenetic trees where multiple trees are combined into a single representative tree. A phylogenetic tree is used to represent the evolutionary relationships between different organisms. We demonstrate in this dissertation that phylogenetic tree combination methods can be used for heterogeneous data clustering, but they have their limitations.

A clustering application involving multiple datasets may demand a partitional clustering or a hierarchical clustering as the output. Phylogenetic tree combination methods combine two or more individual cluster hierarchies to form another cluster hierarchy. Partitional clusters extracted by cutting the resulting cluster hierarchy may not be of satisfactory quality. It is important, therefore, to explore methods for producing both hierarchical and partitional clusterings from the combination of two hierarchical clusterings. Graph-based methods can play an important role since the cluster membership of data points can

be captured in the adjacency relationship of a graph. Also, the phylogenetic methods consider only the taxonomic structure of the original hierarchies and ignore the intra-cluster similarity in the hierarchies. Incorporating the similarity measures associated with the internal nodes of the cluster hierarchies can improve the process of combining them.

When two contextually related datasets are clustered individually, little may be known about their respective distributions. Expectation Maximization (EM) algorithms are very well known for their ability to estimate the parameters of unknown distributions. The EM algorithm can be used to combine two individual clusterings obtained from two contextually related datasets.

The contributions of this research are the following:

A new ensemble-based approach for combining the results of hierarchical clustering of multiple heterogeneous datasets.

- A general mapping technique between hierarchical clusterings.
- Application of supertree and consensus tree methods from phylogenetics to combine clusters from related heterogeneous datasets.
- A new family of graph-theoretic algorithms called EPaCH for combining hierarchical clusters to produce partitional clusters.

A new family of EM-type algorithms called CEMENT for combining the results of partitional clustering of multiple heterogeneous datasets.

An improved method for computing the DB-index [13] for evaluating cluster quality.

Our test domain consists of a large document collection. We have applied our methods to document clustering where the same document set is preprocessed using different mechanisms to create different features [60]. We identified several baselines to compare the performance of our algorithms using a number of different cluster validation indices.

1.4 Organization

This dissertation is organized as follows. In this chapter, we have introduced the research problem, provided the hypothesis, and described the contributions of the dissertation. In Chapter II, we provide a review of the background literature related to the current work. In Chapter III, we describe the problem of clustering heterogeneous datasets, develop the mathematical notation used to describe the problem and algorithms, and present a mapping technique for combining multiple cluster hierarchies resulting from multiple contextually related datasets. In Chapter IV, we present a graph-theoretic algorithm EPaCH for generating partitional clusters from multiple hierarchical clusters and also present a variation, EPaCHW, that considers the intra-cluster similarity measures. In Chapter V, we describe our datasets, present the baselines, and the experimental results for the algorithms presented in Chapter IV. In Chapter VI, we present two EM-type algorithms (CEMENT_1 and CEMENT_2) for combining multiple partitional clusterings and the evaluation results for these algorithms. Finally, we summarize the algorithms developed and their significance and discuss future extensions and possible applications of the research in Chapter VII.

CHAPTER II

LITERATURE REVIEW

Clustering is an unsupervised data mining task used to partition a set of objects into meaningful groups (clusters) such that the objects in a group are similar to each other and dissimilar from objects in another group. Clustering algorithms can be categorized along several dimensions, e.g., hierarchical and partitional, hard and soft, disjunctive and non-disjunctive, deterministic and stochastic, etc. In this chapter, we present an overview of the clustering literature that is relevant to our work.

2.1 Clustering Algorithms

2.1.1 K-means Clustering

K-means clustering [37] is a simple, but popular, form of cluster analysis. It is a partitional clustering method that divides the instances into *k* clusters. Each instance is initially randomly assigned to one of *k* clusters. The mean of each cluster is then computed and each instance is reassigned to the cluster having the closest mean. This two-step process is repeated until the newly computed means in one iteration become identical to the means from the previous iteration or until another stopping criteria has been reached (i.e., maximum number of iterations). The K-median clustering algorithm is a variation of the K-means method that uses the median of each cluster rather than the mean. [25, 46]

K-means clustering is sensitive to the initial selection of clusters. To find an optimal clustering, the algorithm is typically repeated a number of times and the solution with the smallest sum of intra-cluster distances is selected. The K-means algorithm is based on the assumption that each cluster comes from a normal density distribution. The algorithm encounters problems when clusters are derived from distributions that are far from normal, are of differing sizes, or that contain outliers.

The bisecting K-means algorithm [51] is a variant of K-means that can produce either a partitional or a hierarchical clustering by recursively applying the basic K-means method. It starts by considering the whole dataset to be one cluster. At each step, one cluster (often the largest) is selected and divided into two sub-clusters using the basic K-means algorithm. This process continues until the desired number of clusters or some other specified stopping condition is reached.

2.1.2 Hierarchical Clustering

Hierarchical clustering is a powerful and useful method. The basic idea is to compute a dendrogram (Figure 2.1) that assembles a set of instances into a tree. It can be performed in two ways: bottom-up (agglomerative) and top-down (divisive) [25]. The bottom-up approach works by placing each instance in its own cluster and then merging these atomic clusters into progressively larger clusters until all the instances are in a single cluster or some termination point is reached. The top-down approach works by placing all instances

in one cluster and then subdividing this initial cluster into progressively smaller clusters until each instance forms a cluster on its own or some termination point is reached.

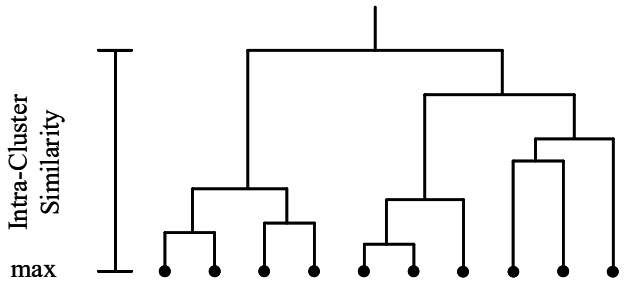


Figure 2.1 Clusters as internal nodes in a dendrogram

The agglomerative algorithm starts by computing a similarity matrix for all pairs of instances. Then the two closest remaining instances/clusters are joined to create a new node of the tree. The length of the branch is based on the similarity between the joined items. The two joined items are replaced by the new node and the similarity matrix is updated. The process is repeated n-1 times until there remains only one cluster of size n [46].

There are three different ways of updating the similarity matrix: *single-link*, *complete-link*, and *average-link* [17]. Single-link is based on the maximum similarity between the instances of two clusters, complete-link is based on the minimum similarity between the instances of two clusters, and average-link is based on the average similarity between the instances of two clusters.

The strength of hierarchical clustering is that it does not require a parameter for the number of clusters and any desired number of clusters can be obtained by cutting the dendrogram at the proper level. It is also less susceptible to noise and outliers. In addition, the structure of the dendrogram may correspond to meaningful taxonomies in some domains.

2.1.3 Graph Theoretic Clustering

The traditional clustering algorithms described above may have an unacceptable computational cost in very high dimensional data spaces. Graph theoretic algorithms are based on graph theory and can overcome the problem with high dimensionality by exploiting the simplif ed structure of a graph. These algorithms also do not depend upon the geometric structure of the clusters. There are many variations in the family of graph theoretic clustering algorithms including minimal spanning tree (MST) based methods, clique based methods, multilevel partitioning methods, hypergraph based methods, etc.

In general, the idea is to produce a graph where vertices correspond to data points and an edge corresponds to the link between two similar data points [62]. Then an algorithm is used to partition the graph into highly connected components based on some constraints and each component refers to a cluster. Hartuv and Shamir [26] presented an algorithm that converts the dataset into a similarity graph and then removes min cut edges (minimum number of edges that disconnects a given connected graph) recursively from any subgraph that is not highly connected (a subgraph whose edge connectivity is less than half the number of nodes).

Minimal spanning trees provide eff cient ways for clustering objects in high-dimensional spaces. A spanning tree is a subgraph of a graph that has the same set of vertices and does not have a cycle and a minimal spanning tree is a spanning tree with the minimum total edge-weight. Prim's algorithm and Kruskal's algorithm [11] are two widely used methods for computing minimal spanning trees in a graph. Prim's algorithm builds upon a single partial minimum spanning tree which is an arbitrary vertex at the beginning. The algorithm repeatedly adds an edge connecting the vertex nearest to but not already in the current partial minimum spanning tree. Kruskal's algorithm maintains a set of partial minimum spanning trees (the set is called a forest) and repeatedly adds the shortest edge in the graph whose vertices are in different partial minimum spanning trees.

In MST based clustering methods, each cluster corresponds to one subtree of the MST. Xu, Olman, and Xu [65] presented MST based algorithms for clustering microarray data. One algorithm constructs a weighted graph from the high-dimensional dataset and creates a MST. Then it removes the $k \quad 1$ largest edges, where k is the number of clusters. The removal of an edge creates one new component and all the resulting components correspond to the k clusters. Each of these clusters has a path that connects all vertices within a cluster with a minimum cost. This simple algorithm is expected to work well if the inter-cluster edge-distances outweigh the intra-cluster edge-distances.

Clustering can also be viewed as f nding large cliques (a *clique* is a complete subgraph of a graph) in a graph. This is essentially identifying clusters of related objects using complete-link hierarchical clustering. In each step of such a clustering, two clusters with

the smallest maximum pairwise distance are merged. Ben-Dor, Shamir, and Yakhini [5] used cliques for clustering biological data. The input data is preprocessed to form an undirected graph and then a stochastic bisection algorithm is used to compute cliques.

Karypis and Kumar [30] presented a multilevel graph partitioning algorithm that works by recursively bisecting the graph until the desired number of subgraphs is reached. At each level of recursion, a given graph (or subgraph) is transformed into a sequence of successively coarser graphs by repeatedly merging pairs of connected nodes. A minimum edge-cut bisection of the coarsest graph is then computed so that each subgraph contains approximately half of the vertex weight of the original graph. This bisection is successively refined (by moving vertices among the partitions) back to the next level of finer graph in order to obtain a bisection of the original graph. Karypis and Kumar [32] modified this recursive bisecting algorithm into a k-way partitioning algorithm where the coarsest graph is directly partitioned into k subgraphs and these subgraphs are successively projected back to the original graph. This k-way partitioning results in a speedup by a factor of $O(\log_2 k)$.

Clustering problems have also been solved using hypergraph based approaches. A hypergraph is a graph where an edge can connect more than two vertices, as opposed to regular graphs, where an edge connects only two vertices. One of the straightforward solutions to hypergraph clustering is to convert the hypergraph to a graph where each hyperedge is replaced by a set of regular edges (in a clique) and then to apply the regular multilevel graph partitioning algorithm [2]. Karypis and Kumar [31] extended their multilevel graph

partitioning algorithm [32] to make it work for hypergraph partitioning where the goal is to minimize the number of hyperedges connecting vertices from different components of the hypergraph.

2.1.4 EM Clustering

Expectation Maximization (EM) is an approach for learning probabilistic models in problems that involve hidden variables [15]. The EM algorithm works by approximating a probability function and is typically used to compute maximum likelihood estimates from incomplete samples. When applied to clustering, EM views the dataset as incomplete and iteratively recomputes the hidden parameters until a desired convergence value is achieved. An EM clustering starts by initializing the parameter vector, e.g., the mean for each class. In the E-step, it computes the probability of an object belonging to a class. In the M-step, the parameters for each class are adjusted. These EM steps are repeated until convergence. As the algorithm progresses through different iterations of the E-step, it does not actually assign an object to a cluster. But after the algorithm converges, based on the largest probability, each object is actually assigned to a cluster. The k-means algorithm is a special form EM clustering when the objects have a spherical Gaussian distribution [9]. Fraley and Raftery [20] presented a comprehensive EM-based clustering framework that uses a mean vector and a covariance matrix as the parameters to be estimated. An agglomerative hierarchical clustering is used to generate the initial partition and compute the initial parameters.

Two major drawbacks of EM clustering are its sensitivity to the selection of initial parameters and the slow convergence rate. Different approaches have been proposed to overcome these. Ordonez, Omiecinski, and Ezquerra [42] presented an efficient EM clustering algorithm that performs the initialization based on the global mean and the global covariance. These values are computed in one extra pass. Caffo, Jank, Jones [8] presented a fast EM clustering algorithm that uses only a subset of the entire dataset instead of making a pass over all the objects. The sample size is increased gradually with the iterations so that accuracy is not sacrificed. Verbeek, Nunnink, and Vlassis [56] presented an accelerated version of the EM clustering, where instead of computing the conditional expectation of the missing data in the E-step for each object, it is computed for a group of objects.

2.1.5 Semi-Supervised Clustering

In general, semi-supervised learning deals with learning from independently sampled labeled and unlabeled data. Prior knowledge plays a very important role in a semi-supervised learning method. Even though clustering is traditionally perceived to be an unsupervised process, in a semi-supervised clustering framework, the performance of an unsupervised clustering algorithm can be improved with some supervision in the form of some labeled data or constraints. Semi-supervised clustering involves two general approaches: constraint-based and distance-based [4].

In constraint-based approaches, the clustering algorithm itself is modified so that user-provided labels or constraints can be used to obtain better results. Wagstaff et al. [59]

presented a constrained K-means algorithm that uses a set of "pairwise constraints" to specify whether two data points should be in the same class (*must-linked*) or in different classes (*cannot-linked*). Basu, Banerjee, and Mooney [3] use a set of initial seed clusters generated from labeled data to perform subsequent iterations of the K-means algorithm. In distance-based approaches, the distance measure used by the particular clustering algorithm is trained to satisfy the labels or constraints in the supervised data. Skarmeta, Bensaid, and Tazi [50] applied a semi-supervised framework for text categorization where labeled data is used to tune the distance matrix and guide hierarchical clustering. Klein et al. [33] used a shortest path algorithm to modify the distance metric based on pairwise constraints and applied a complete-link hierarchical algorithm on the modif ed metric. Xing et al. [64] presented an iterative convergence algorithm that learns a distance metric based on similar and dissimilar examples provided by the user. Bilenko, Basu, and Mooney [6] provided a K-Means-based semi-supervised clustering algorithm that integrates both metric learning and pairwise constraints.

2.2 Similarity/Distance Measures

Most clustering algorithms use some notion of similarity (or distance) to determine the closeness of objects. The choice of a particular measure may depend on the particular application domain and the type of the objects. Nevertheless, the performance of a clustering algorithm for a particular domain depends on the selection of the particular measure. The objects in a dataset are usually represented as a set of features. The features can be

of different types, e.g., continuous, categorical, nominal, binary etc. A desired property of a similarity (or distance) measure is to be a *metric*. If the distance between two objects is normalized between zero and one, then it is common to compute the distance between two objects and then compute the similarity by subtracting the distance from one and vice-versa. A distance function $d(a\ b)$ between two objects a and b is called a distance metric, if it satisf es the following properties:

it is non-negative, i.e., $d(a\ b)$ 0

it is ref exive, i.e., $d(a\ a) = 0$

it is symmetric, i.e., $d(a\ b) = d(b\ a)$, and

it follows triangular inequality, i.e., $d(i\ j)$ $d(i\ k) + d(k\ j)$

A popular distance measure that is used in k-means type partitional clustering is the *Minkowski* distance that is def ned by [25]:

$$d(a\ b) = \sqrt[n]{\sum_{i=1}^{m} a_i \quad b_i\ ^n}$$

where m is the number of features. Two special cases of the *Minkowski* distance that are commonly used are the *Euclidean* distance ($n = 2$) and the *Manhattan* distance $n = 1$. Both are metrics and work well when the data objects are compact and clusters are well-separated, but allow the larger-scaled features to dominate others. Normalization of the features is used to overcome this problem.

Pearson's coeff cient is a measure that is based on the correlation between the objects of two datasets. It is commonly used for clustering gene expression data. Its main draw-

back lies in its inability to compute the magnitude of the difference between two objects. Also, it is not a metric. Similarity is computed using Pearson's coeff cient by [18]:

$$s(a\ \ b) = \frac{1}{m} \sum_{i=1}^{m} \frac{a_i \quad \overline{a}}{a} \quad \frac{b_i \quad \overline{b}}{b}$$

where $\overline{a}$ is the mean of a and $\ _a$ is the standard deviation of a given by:

$$_a = \sqrt{\sum_{i=1}^{m} \frac{(a_i \quad \overline{a})^2}{m}}$$

Another important similarity metric frequently used for document clustering is *cosine* similarity. The cosine measure is based on the notion of two vectors in an m-dimensional space and is represented by the normalized dot product of the two vectors which is essentially the cosine value of the angle between them. Mathematically, cosine similarity can be written as: [60],

$$s(a\ \ b) = \frac{a \quad b}{a \quad b} = \frac{\sum_{i=1}^{m} a_i b_i}{\sqrt{\sum_{i=1}^{m} a_i^2}\ \sqrt{\sum_{i=1}^{m} b_i^2}}$$

A common similarity measure that is used for binary features is the *Jaccard* coeff - cient. It is computed as the ratio of the number of attributes that are present in both the objects and the number of attributes that are present in at least one of the two objects [25].

2.3 Cluster Ensemble

Though different clustering algorithms are available, they can produce different clustering results due to factors such as noise in the data, inconsistency among the algorithms,

data distribution, pre-processing procedures, size of data, etc. The difficulty of identifying one particular technique as the best has motivated research into combining different clustering results.

Fred and Jain [21] formulated the problem of combining different clustering results as "evidence accumulation" where each clustering result is viewed as independent evidence of data organization. Individual cluster sets are generated by different runs of the K-means algorithm using random initializations. Then each clustering is mapped into a coassociation matrix using a voting mechanism. The coassociation between two objects is calculated by the frequency of their cooccurrences in the same cluster. The final clusters are then obtained by applying a minimum spanning tree based clustering algorithm on this matrix, cutting weak links at a threshold. This essentially carries out a single link method by merging each pair of objects or the clusters they used to belong to if they pass the threshold test.

Almost concurrently with this work, Strehl and Ghosh [53] introduced the notion of a cluster ensemble whose goal is to combine the output of multiple clustering algorithms to obtain better quality and more robust clustering results. Their framework is based on transforming the set of clusterings into a hypergraph where an edge can connect more than two vertices as opposed to regular graphs. The hypergraph represents each object as a vertex and each cluster as a hyperedge. A concatenated hypergraph matrix is formed where each column represents a hyperedge (essentially a 0/1 vector based on whether an object belongs to the cluster). Three techniques are then used to generate the ensemble. With

the first one, called Cluster-based Similarity Partitioning Algorithm (CSPA), a similarity matrix is constructed by defining the similarity between two objects as the fraction of clustering in which they belong to the same cluster. This similarity matrix is used to construct a weighted similarity graph, where the vertices correspond to the data points and an edge represents a high similarity (with respect to a threshold) between two data points. Then a graph partitioning algorithm is used to extract clusters from the resulting graph. With the second approach, called HyperGraph Partitioning Algorithm (HGPA), the hypergraph is partitioned by cutting a minimal number of hyper-edges using a suitable hypergraph partitioning method. With the third approach, called Meta-Clustering Algorithm (MCLA), a meta-cluster is formed by clustering the hyperedges and collapsing the related hyperedges. A metagraph is constructed for this where each hyperedge becomes a vertex and the Jaccard coefficient is used to calculate the similarity between two hyperedges. Each object is assigned to the collapsed hyperedge in which it participates most strongly.

Hu and Yoo [28] used an additive ensemble approach that is based on generating different clustering results using various clustering algorithms on the same dataset. The clustering result from each algorithm is used to construct a distance matrix using probability density functions. The individual distance matrices are added to build a master distance matrix. This master distance matrix induces a weighted similarity graph and a graph partitioning algorithm is used to partition the resulting graph into a set of clusters.

Fern and Brodley [19] used bipartite graph partitioning to solve the cluster ensemble problem. They represented both data objects and clusters as vertices of the same graph.

This procedure does not encounter the problem of ignoring the cluster similarity while computing the object similarity and vice versa, and both types of similarity calculation are performed collectively. They used different runs of K-means on randomly sampled (both vertically and horizontally) subsets of the original dataset to generate a set of individual clusterings. When this set of clusterings is transformed into a graph, two vertices are made adjacent only if one is an object and the other is a cluster that the object belongs to. If a new clustering is added to the ensemble, a new set of cluster vertices are added to the graph and each of them is connected to the constituent objects. The resulting graph is bipartite since cluster vertices are connected only to object vertices and vice versa. Then this bipartite graph is partitioned using a graph partitioning method where the objects and the clusters are partitioned simultaneously and the resulting partition of the objects is output as the f nal clustering.

2.4 Combining Phylogenetic Trees

A phylogenetic tree depicts the evolutionary relationships between different species where each leaf represents a species, each internal node represents the most recent common ancestor of its descendants, and the root represents the common ancestor of all species (http://en.wikipedia.org). If different datasets are used to build phylogenetic trees for the same set of species, the resulting trees may show dissimilar evolutionary history. These different views can be resolved by using a tree combination method that takes a collection of phylogenetic trees and outputs a single representative tree.

Margush and McMorris [38] introduced the notion of "consensus tree" to combine phylogenetic trees. There are different variations of the consensus tree methods [7]. The strict consensus tree contains exactly those subtrees common to all the trees in the collection. The majority rule tree contains exactly those subtrees that appear in more than half of the input trees. The greedy consensus contains subtrees ordered by decreasing frequency of appearance in the input trees.

Gusfield [24] presented a tree refinement approach for combining phylogenetic trees that can contain more than one object at each leaf. A phylogenetic tree is said to refine another if the second one can be obtained by contracting a number of edges of the first one, i.e., the first one will contain the evolutionary history contained in the second plus some additional information. Two trees are said to be compatible if an *agreement tree* can be found that refines both. The algorithm inspects each tree with respect to the other and modifies each using the additional information found on the other. If the leaf containing an object in one tree has more objects than the leaf containing the same object in the other tree, then it replaces that leaf by a subtree from the other that consists of the same objects. This is done for both the trees and the two modified trees are compared bottom-up to check whether each possible pair of corresponding nodes in the two trees contains the same objects in the leaves below.

The above methods may result in unresolved trees when two trees consist of overlapping sets of objects. Another class of methods called "supertree" methods has been developed that can work with such trees. In phylogenetics, a supertree is defined as a rooted

evolutionary tree that is assembled from smaller trees that share some but not necessarily all leaf nodes in common. A supertree will consist of all the objects in the individual input trees. The most straightforward approach is to combine the matrix representation of data used to construct each input tree into a single supermatrix and construct the supertree from the resulting matrix using a maximum parsimony algorithm [48].

Gordon [22] presented a pruned tree based approach that works by first pruning objects from the trees and then grafting the pruned objects back to create a new consensus tree. A common pruned tree is defined as a tree that can be derived by pruning edges from two trees so that the reduced trees consist of the same possible subtrees. For example, the common pruned tree for the trees $(((a\ b)\ (c\ d))\ e)$ and $(((a\ d)\ (b\ c))\ e)$ will be $((a\ c)\ e)$. Gordon uses each subset of the common pruned tree to generate supersequences with respect to both the trees and those supersequences are used to generate a supertree. For example, the supertree for the above example will be $((a\ b\ c\ d)\ e)$. This algorithm runs in polynomial time, but does not generate a supertree with incompatible trees and depends on the order of input trees.

Ragan [44] presented a technique called "Matrix Representation using Parsimony" which is by far the most popular supertree algorithm used by the biologists. The internal nodes of each tree are labeled with distinct characters and all the input trees are converted into a composite binary encoded matrix based on those internal node labels. The supertree is constructed from the resulting matrix using a maximum parsimony algorithm.

Semple and Steel [49] presented a graph-based algorithm for combining the input trees into a supertree even if they are not compatible. It converts the input trees into a composite weighted graph where each leaf node is represented by a vertex and the weight of each edge is equal to number of occurrences of the corresponding pair of leaf nodes in some proper clusters. The resulting graph is then modified by collapsing the edges that have the same weight as the number of source trees, and the modified graph is partitioned by cutting the edges that results in the minimum cut-edge weights. This process is recursively applied until each lowest level subtree contains fewer than three leaves.

2.5 Clustering Heterogeneous Datasets

The traditional clustering paradigm pertains to a single dataset. Even with the ensemble approach, different clusters are generated with multiple runs of the same algorithm or different algorithms, but with the same dataset. Recently, there has been some research in clustering multiple heterogeneous datasets where the datasets are related but contain information about different types of objects and the attributes of the different object sets differs significantly.

Marx et al. [39] presented a variant of traditional clustering, called *coupled clustering*, for revealing analogies between clusters from two distinct datasets. They formed a one-to-one correspondence between two cluster sets so that each cluster is matched with a counterpart in the other data set. Each pair of matched clusters is joined to form a coupled cluster. This requires that the similarity values between heterogeneous pairs of

data elements be computed. They applied coupled clustering to a collection of text documents from two distinct domains that are characterized by their own terminology and key-concepts.

Dagan, Marx, and Shamir [12] presented a framework for clustering multiple datasets (related but distinct domains) such that each cluster includes elements from multiple datasets and can capture a common theme. This approach is based on word co-occurrence statistics within the analyzed datasets and essentially constructs a common feature set consisting of frequent keywords from the datasets. A data instance is assigned to a cluster probabilistically using an annealing like process. They applied their approach to three sets of religion-related keywords from three different religions to reveal common themes like scriptures, rituals, etc. based on respective keyword clusters.

Pavlidis et al. [43] presented a support vector machine (SVM) based approach to clustering objects using heterogeneous datasets. The SVM computes separate kernels for each data type and combines them to construct an explicitly heterogeneous kernel function. McClean, Scotney, and Robinson [40] presented a Hidden Markov Model (HMM) based framework for clustering heterogeneous gene sequences. They cluster similar sequences and then use the mappings between the states of each sequence in a cluster and the states of an underlying hidden process to learn the probabilistic description of an HMM for each cluster.

Dayanik and Manning [14] presented a clustering approach that is based on a relational representation of heterogeneous biological data, i.e., multiple datasets are related

by key attributes. The data points in one dataset contain references to the data points in the other datasets. This mapping allows explicit links to be established between heterogeneous data points. These types of links are used to connect heterogeneous data objects in a weighted undirected graph. A multilevel graph partitioning algorithm is used to generate clusters from this graph. This approach was applied to biological data where scientif c abstracts contain links to protein sequences and 3D protein structures and the 3D protein structures are cross-referenced to the primary citations in the abstracts.

CHAPTER III

CLUSTERING HETEROGENEOUS DATASETS

The traditional clustering paradigm pertains to a single dataset. The overall objective of the dissertation is to provide a framework and algorithms for clustering multiple heterogeneous datasets where the datasets are related but contain information about different types of objects. The attributes of the different datasets can also differ signif cantly. In this chapter we develop the mathematical notation for describing our clustering framework, present the problem description, and also present a general approach for dealing with clustering of heterogeneous datasets. Although the notation and approach will be described in terms of combining clustering results from two datasets, both can be extended to deal with more than two datasets.

3.1 Problem Description

We begin by def ning the problem of clustering heterogeneous datasets in general. Let $F_x = \begin{matrix} f_1^x & f_2^x & \cdots & f_{m_x}^x \end{matrix}$ and $F_y = \begin{matrix} f_1^y & f_2^y & \cdots & f_{m_y}^y \end{matrix}$ be two sets of features. Values for each feature will come from a specif ed domain. Let $\begin{matrix} x \\ j \end{matrix}$ be the domain of f_j^x. Then, we have two datasets $D_x \quad \begin{matrix} x & x & & x \\ 1 & 2 & \cdots & m_x \end{matrix}$ and $D_y \quad \begin{matrix} y & y & & y \\ 1 & 2 & \cdots & m_y \end{matrix}$. Let o_j^x be an object belonging to D_x and o_j^y be an object belonging to D_y. Then, $D_x = \begin{matrix} o_1^x & o_2^x & \cdots & o_{n_x}^x \end{matrix}$ and $D_y = \begin{matrix} o_1^y & o_2^y & \cdots & o_{n_y}^y \end{matrix}$, where each dataset D_i consists of n_i objects.

Heterogeneity between two or more datasets can occur at the feature level, at the object level, or both. At the feature level, there can be three cases. The datasets may have the same features, i.e., $F_x = F_y$, some common features, i.e., $(F_x \quad F_y) \quad (F_x \quad F_y)$, or no common feature, i.e., $(F_x \quad F_y) = \quad$. The same situation can occur at the object level, i.e., $D_x = D_y$, $(D_x \quad D_y) \quad (D_x \quad D_y)$, or $D_x \quad D_y = \quad$. The ensemble algorithms found in the literature [19, 53, 28] primarily deal with the situation when $F_x = F_y$ and $D_x = D_y$. In this dissertation, we will develop algorithms to deal with the situation when $F_x = F_y$ and $D_x \quad D_y$.

Given D_x and D_y, we can create a unif ed dataset $D_{xy} = D_x \quad D_y$. If we need to build a non-overlapping set of partitional clusters from D_{xy}, then we can think of an equivalence relation $\mathcal{R}$ on D_{xy}, where each relation pair represents two objects belonging to the same cluster. The equivalence classes of D_{xy} with respect to $\mathcal{R}$ will form a partition of D_{xy}. If the equivalence class of o_j^x is E_j^x, then the equivalence classes of D_x are $E_x = \quad E_1^x \quad E_2^x \qquad E_{n_x}^x$ and the equivalence classes of D_y are $E_y = \quad E_1^y \quad E_2^y \qquad E_{n_y}^y$. Let us def ne a heterogeneous cluster H_j to be a set of objects consisting of objects from both D_x and D_y. Then each equivalence class described above will represent a heterogeneous cluster H_j. Given all these def nitions, our clustering problem is to f nd a set of heterogeneous clusters $\mathcal{H}$, where $\mathcal{H} = \quad H_1 \quad H_2 \qquad H_k \quad = E_x \quad E_y$.

The problem of clustering heterogeneous data can be solved in two general ways: (1) clustering the multiple datasets using a unif ed feature space and (2) clustering the datasets individually and then combining the resulting clusters based on some mapping. The f rst

approach will require us to integrate the heterogeneous data into a unif ed feature space and then to perform the clustering on the integrated dataset. Essentially, the problem of clustering heterogeneous datasets then becomes a function, $D_x \quad D_y \qquad H_1 \; H_2 \qquad H_k$. The second approach will require us to generate different sets of clusters from individual datasets. Then these can be combined using a meta-clustering scheme. For example, we can apply a hierarchical clustering algorithm to cluster the individual datasets and then combine the corresponding dendrograms. If $\mathcal{C}_x = \; C_1^x \; C_2^x \qquad C_p^x$ and $\mathcal{C}_y = \; C_1^y \; C_2^y \qquad C_q^y$ are the sets of clusters computed from the individual datasets D_x and D_y, then the problem of clustering heterogeneous data becomes a function, $\mathcal{C}_x \quad \mathcal{C}_y \; H_1 \; H_2 \qquad H_k$.

3.2 Clustering Heterogeneous Datasets with Hierarchical Clustering

Hierarchical clustering is a powerful clustering method that computes a dendrogram to organize a set of objects into a tree. It is very important to have effective and eff cient clustering algorithms that can provide intuitive browsing capabilities by organizing large datasets into a set of meaningful clusters. Hierarchical clustering plays an important role in providing a view of the data at different levels of granularity. The resulting hierarchies allow for meaningful visualization and interactive exploration of a large set of objects.

3.2.1 Hierarchical Clustering

Hierarchical clustering is performed in a series of steps. The agglomerative version starts with a number of singleton clusters and then proceeds by combining these singleton

clusters into successively larger sub-clusters. The divisive version starts with a single cluster containing all the objects and then proceeds by partitioning this all-inclusive cluster into successively smaller sub-clusters. The goal of any hierarchical algorithm to produce a dendrogram as shown in Figure 2.1. Each leaf of the dendrogram represents a singleton cluster and each internal node represents a sub-cluster. Since agglomerative methods are more commonly used and we have used agglomerative methods in our work, we will provide an overview of the basic agglomerative algorithm.

An agglomerative algorithm is based on a "metric" that is used to compute a proximity matrix that initially contains the pair-wise similarities for all the data objects. This proximity matrix is used to decide which two of the closest remaining clusters will be combined at each step of the algorithm. At each step, the pair of clusters with the maximum similarity (minimum distance) are combined to create a new cluster. Then the algorithm computes the similarity between this new cluster to all other clusters and updates the matrix by adding a new row and column corresponding to the new cluster and deleting the rows and columns corresponding to the combined clusters. This process continues until the similarity matrix is reduced to a single element. Figure 3.1 presents the agglomerative clustering algorithm.

Different methods are used to compute the similarity between two clusters in step 10. Three important methods are *single-link*, *complete-link*, and *average-link*. In the single-link method, the similarity of two clusters is computed as the similarity between the closest pair of objects in the two clusters. In the complete-link method, the similarity between

```
Input:  A set of objects  o_1 o_2    o_n
Output:  A set of nested clustering C =  C_1 C_2     C_n 1
 1.  for i   1 to n do
 2.     C_i    o_i
 3.  for i   1 to n do
 4.     for j   1 to n do
 5.        s_ij    Similarity(C_i C_j)
 6.  for i   1 to n   1 do
 7.     C_n+i    argmax_s_pq C_p   C_q
 8.     C_i   C_n+i
 9.     for j   1 to n + i do
10.        s_j(n+i)    s_(n+i)j    Similarity(C_j C_n+i)
11.        s_jp    s_pj    s_jq    s_qj    0
```

Figure 3.1 The agglomerative clustering algorithm

two clusters is computed based on the similarity between the two farthest objects. In the average-link method, the similarity between two clusters is computed based on the average distance between all pair of objects. Mathematically, the similarity between two clusters C_i and C_j can be written as:

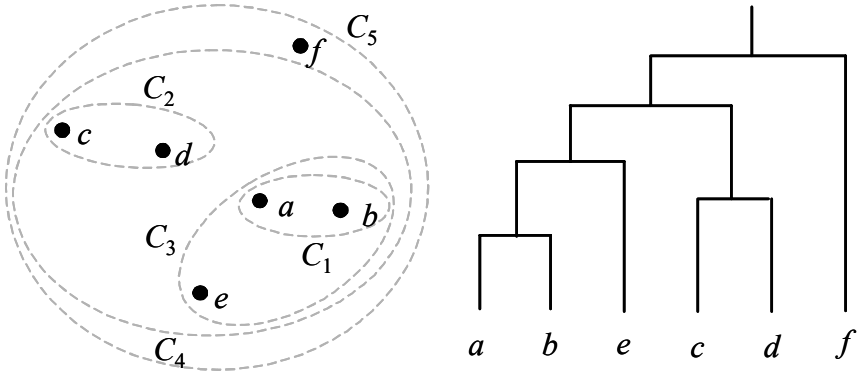
$$Similarity(C_i\ C_j) = \begin{cases} \max_{o_p\ C_i\ o_q\ C_j}(s_{pq}) & \text{Single Link} \\ \min_{o_p\ C_i\ o_q\ C_j}(s_{pq}) & \text{Complete Link} \\ \frac{1}{C_i\ C_j} \quad o_p\ C_i \quad o_q\ C_j}(s_{pq}) & \text{Average Link} \end{cases}$$

The single-link method is sensitive to noise and outliers whereas the complete-link method tends to break large clusters. The average-link is a compromise between the two. Since we have used the average link similarity in our work, let us explain that with an example. Figure 3.2(a) shows a set of objects $a\ b\ c\ d\ e\ f$ in a two-dimensional space. Each object is initialized as a singleton cluster. At this point, the maximum similarity is between a and b, and a and b will be combined into C_1. Now, the maximum similarity is between c and d, and c and d will be combined into C_2. In the next step, the maximum similarity is between C_1 and e (based on average similarity between them), and they will be combined into C_3. This process will continue until one big cluster is produced. The corresponding dendrogram is shown in Figure 3.2(b). In step 10 of the agglomerative algorithm (Figure 3.1), it is not necessary to perform $O(n^2)$ computations to compute the similarity between a new cluster and all existing clusters. With the average-link method, the similarity between an exiting cluster C_j and a new cluster $C_k = C_p \quad C_q$ is computed as [18]:

$$\frac{n_p s_{jp} + n_q s_{jp}}{n_p n_q}$$

where n_p and n_q are the number of objects in C_p and C_q respectively.

The time complexity of the single-link and complete-link methods are $O(n^2)$ and the average-link method is $O(n^2 \lg n)$. Since, the algorithm must maintain a similarity matrix, the space complexity is $O(n^2)$.



(a) Nested clustering (b) The dendrogram

Figure 3.2 The average-link agglomerative clustering process

3.2.2 Hierarchical Clustering on Heterogeneous Datasets

As mentioned in the previous section, one way to cluster heterogeneous datasets is to integrate them into a unif ed feature space and then perform the clustering [12, 16, 41, 43, 55]. This approach has some drawbacks. If the feature sets are complementary in nature, then the ability of the feature sets to contribute different information may be lost. The unif ed feature space may be biased towards the feature set consisting of more features.

Let us explain this with an example. Figure 3.3(a) shows a dataset D_1 that has six objects and three features. Figure 3.3(b) shows another dataset D_2 that has the same six objects but seven features that are different from features in the f rst dataset. If we use the normalized Euclidean distance measure [61], we will obtain the dendrograms shown in Figures 3.4(a) and 3.4(a) respectively from D_1 and D_2. It can be seen that the f rst dendrogram provides clustering information that is somewhat different from the second dendrogram. If we combine the two datasets by concatenating the feature sets, we will

obtain the dataset D_3 as shown in Figure 3.3(c). Hierarchical clustering on D_3 will yield the same dendrogram as the one obtained from D_2. So, the information provided by D_1, i.e., the ability of the dendrogram generated from D_1 to interpolate, is lost here. The reason is straightforward, the influence of D_1 on the concatenated feature set, and hence on the similarity (distance) matrix is overpowered by the influence of D_2.

	f_1	f_2	f_3
o_1	0.8	0.5	0.9
o_2	0.85	0.45	0.95
o_3	0.5	0.8	0.6
o_4	0.35	0.85	0.55
o_5	0.7	0.6	0.8
o_6	0.6	0.7	0.7

(a) D_1 - three features

	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
o_1	0.8	0.7	0.9	0.6	0.8	0.7	0.8
o_2	0.75	0.65	0.85	0.55	0.75	0.65	0.75
o_3	0.9	0.8	0.75	0.7	0.65	0.8	0.65
o_4	0.6	0.5	0.6	0.4	0.5	0.55	0.45
o_5	0.45	0.35	0.45	0.25	0.35	0.4	0.3
o_6	0.55	0.4	0.5	0.35	0.45	0.45	0.35

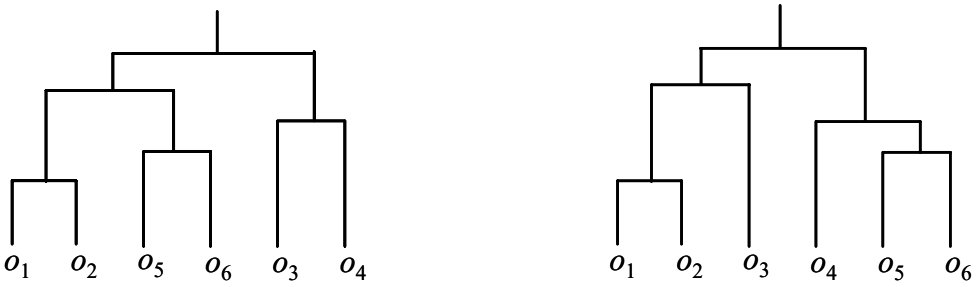
(b) D_2 - seven features

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
o_1	0.8	0.5	0.9	0.8	0.7	0.9	0.6	0.8	0.7	0.8
o_2	0.85	0.45	0.95	0.75	0.65	0.85	0.55	0.75	0.65	0.75
o_3	0.5	0.8	0.6	0.9	0.8	0.75	0.7	0.65	0.8	0.65
o_4	0.35	0.85	0.55	0.6	0.5	0.6	0.4	0.5	0.55	0.45
o_5	0.7	0.6	0.8	0.45	0.35	0.45	0.25	0.35	0.4	0.3
o_6	0.6	0.7	0.7	0.55	0.4	0.5	0.35	0.45	0.45	0.35

(c) D_3 - concatenated features

Figure 3.3 Datasets with identical objects but different feature sets

The above problem can be more complicated if the two datasets have some features in common. Also, a unified feature space may cause substantial sparseness. This is true in particular when two datasets do not represent exactly the same set of objects but have



(a) (b)

Figure 3.4 Dendrograms generated by the datasets of Figure 3.3 - Figure 3.3(a) generates 3.4(a) and Figures 3.3(b) and 3.3(c) generate 3.4(b).

some objects in common. If the individual feature sets consist of entirely different types, e.g., one is nominal and the other is continuous, it may be difficult to generate a unified feature space.

Our research hypothesis is based on the other approach, i.e., to perform separate clustering of the individual datasets based on their respective feature sets and then combine these clustering results into a final set of clusters. This approach makes use of complementary information in each dataset and allows individual clusterings to be mutually informative.

In this chapter, we provide a general approach to cluster heterogeneous datasets. This approach is based on the hierarchical clustering of the individual datasets and is expected to explore the contributing effect of heterogeneous datasets. Our approach is presented for two datasets and assumes the notion of a mapping between the two datasets. In the next section, we present the notion of mapping at an abstraction level that can be tailored to specific applications.

3.3 Mapping between Heterogeneous Datasets

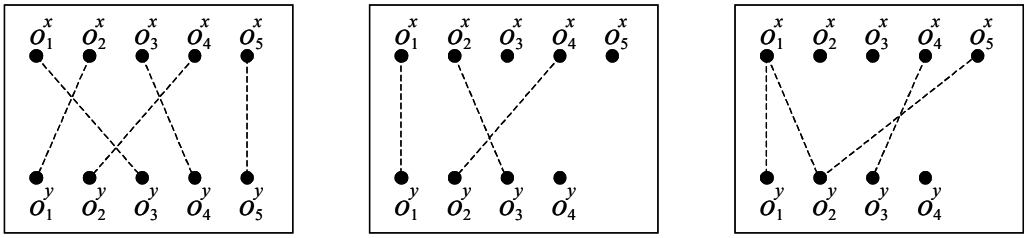
It is very important to establish a mapping between the objects of two datasets when the datasets are heterogeneous in nature. In this section, we use a bipartite graph to represent the mapping between the objects of two datasets. We first define "mapping" and then examine the different types of mapping that reflect the restrictions on how objects may be related across heterogeneous datasets.

We define *mapping* as the number of possible objects in one dataset that may be related to a single object in another datasets. A mapping represents a constraint on the way heterogeneous objects are related. We represent the mapping between two datasets D_x and D_y using a bipartite graph $\mathcal{G} = (D_x\ D_y\ \mathcal{E})$ such that $\mathcal{E} =$ $a\ b$ a $D_x\ b$ D_y . D_x D_y is the set of vertices and $\mathcal{E}$ is the set of edges of $\mathcal{G}$. We classify the mapping between two heterogenous datasets based on three different situations and we name those corresponding to the type of the bipartite graph. The three types of mapping are *perfect*, *bi-regular*, and *irregular*.

A mapping is *perfect* when a perfect matching exists in $\mathcal{G}$. In this case, there is a one-to-one correspondence between the objects of the two datasets and $D_x = D_y$. This situation is depicted in Figure 3.5(a). Each dataset has five objects, $o_1^x\ o_2^x\ o_3^x\ o_4^x\ o_5^x$ and $o_1^y\ o_2^y\ o_3^y\ o_4^y\ o_5^y$ respectively. The object pairs that are mapped to each other are $o_1^x\ o_3^y$, $o_2^x\ o_1^y$, $o_3^x\ o_4^y$, $o_4^x\ o_2^y$, and $o_5^x\ o_5^y$. It can be seen that each object is mapped to exactly one object in the other dataset.

A mapping is *bi-regular* if every vertex in $\mathcal{G}$ has a degree of 0 or 1. In this case, a data object in one dataset is mapped to one or no object in the other dataset and the number of objects in the datasets may not be the same. This is shown in Figure 3.5(b). Dataset D_x has five objects and D_y has four objects, $o_1^x\ o_2^x\ o_3^x\ o_4^x\ o_5^x$ and $o_1^y\ o_2^y\ o_3^y\ o_4^y$ respectively. The object pairs that are mapped to each other are $o_1^x\ o_1^y$, $o_2^x\ o_3^y$, and $o_4^x\ o_2^y$. The data objects $o_3^x\ o_5^x$ and o_4^y are not mapped to objects in the other dataset.

A mapping is *irregular* if every vertex in $\mathcal{G}$ has a degree of 0 or more. In this case, there is an $m\ n$ mapping between the objects of the two datasets, where $m\ n\ 0$. Again, the datasets may not have the same number of objects. This is shown in Figure 3.5(c). The data object o_1^x is mapped to two different objects o_1^y and o_2^y, o_2^y is mapped to two objects o_1^x and o_5^x, and o_4^x is mapped to one object o_3^y. On the other hand, o_2^x, o_3^x, and o_4^y are not mapped to anything.



(a) A perfect mapping (b) A bi-regular mapping (c) An irregular mapping

Figure 3.5 Different types of mapping between the objects of two datasets

3.4 Combining Heterogeneous Cluster Hierarchies

Now that we have def ned the abstraction of mapping, we will describe how this mapping can be used to combine heterogeneous cluster hierarchies. We begin by introducing mathematical notation related to dendrograms and then explain with examples how two dendrograms can be ref ned for heterogeneous clustering.

3.4.1 Preliminary Def nitions

Let us assume that we have two datasets $D_x = \ o_1^x \ o_2^x \quad o_{n_x}^x \ $ and $D_y = \ o_1^y \ o_2^y \quad o_{n_y}^y \ $. Let $\mathcal{D}_i$ be the dendrogram generated by hierarchically clustering D_i. $\mathcal{D}_i$ consists of a set of internal nodes and a set of external nodes (or *leaves*). We assume that $\mathcal{D}_i$ has the properties of a binary rooted tree, i.e., it has a distinct root and each internal node has exactly two descendants.

We also assume that the leaves of $\mathcal{D}_i$ are distinct. Let $\mathcal{L}_i$ be the leaves of $\mathcal{D}_i$. Intuitively, we can say that the clustering is a mapping $D_i \quad \mathcal{D}_i$ where $D_i = \mathcal{L}_i$. Let the set of internal nodes be $\mathcal{C}_i = \ C_1^i \ C_2^i \quad C_{n_i \ 1}^i \ $. We def ne $\ _l(N_p)$ and $\ _r(N_p)$ to be the left and right descendants of the node N_p. Alternatively, we also say $N_p \ . \ N_q$ if N_q descends directly from N_p. Conversely, we def ne $\ (N_p)$ to be the immediate ancestor of N_p. Also, we def ne $\ (N_p)$ to be the sibling of N_p so that $\ (N_p) = \ (\ (N_p))$. Each C_j^i

represents a cluster that comprises all the external nodes that descend from C_j^i. We def ne C_j^i recursively by,

$$C_j^i = \begin{array}{llll} C_p^i \quad C_q^i & \text{where } 1 \quad p \ q < j & & \text{if } {}_l(C_j^i) \quad \mathcal{C}_i \text{ and } {}_r(C_j^i) \quad \mathcal{C}_i \\ o_p^i \quad C_q^i & \text{where } 1 \quad p \quad n_i \text{ and } 1 \quad q < j & \text{if } {}_l(C_j^i) \quad D_i \text{ and } {}_r(C_j^i) \quad \mathcal{C}_i \\ C_p^i \quad o_q^i & \text{where } 1 \quad p < j \text{ and } 1 \quad q \quad n_i & \text{if } {}_l(C_j^i) \quad \mathcal{C}_i \text{ and } {}_r(C_j^i) \quad D_i \\ o_p^i \quad o_q^i & \text{where } 1 \quad p \ q \quad n_i & \text{if } {}_l(C_j^i) \quad D_i \text{ and } {}_r(C_j^i) \quad D_i \end{array}$$

Having def ned $\mathcal{L}_i$ and $\mathcal{C}_i$, we can def ne a dendrogram as $\mathcal{D}_i = (\mathcal{N}_i \ \mathcal{E}_i)$, where $\mathcal{N}_i$ is the set of nodes given by $\mathcal{N}_i = \mathcal{L}_i \quad \mathcal{C}_i$ and $\mathcal{E}_i$ is the set of edges given by $\mathcal{E}_i = \quad (a \ b) \ a \ b$ $\mathcal{N}_i \quad a . b$. It should be noted that $\mathcal{L}_i = n_i$, $\mathcal{C}_i = n_i \ 1$, $\mathcal{N}_i = 2n_i \ 1$, and $\mathcal{E}_i = 2n_i \ 2$.

The key feature of a dendrogram is that each leaf represents a singleton cluster and each internal node represents a cluster that is formed by merging the clusters that appear in its descendants with the root representing a cluster containing all the objects. For example, as shown in Figure 3.6, the dendrogram represents a hierarchical clustering on the object set $a \ b \ c \ d \ e \ f \ g$. It consists of the leaves $\mathcal{L} = \ a \ b \ c \ d \ e \ f \ g$ and the internal nodes $\mathcal{C} = \ C_1 \ C_2 \quad C_6$. The internal node C_1 has descendants e and f both of which are external nodes and thus represents the cluster $e \ f$. The internal node C_4 has descendants C_1 and g and thus represents the cluster $e \ f \ g$. The internal node C_6 has descendants C_4 and C_5 and thus represents the cluster $a \ b \ c \ d \ e \ f \ g$.

We assume that each dendrogram $\mathcal{D}_i$ has a similarity matrix $[s_{pq}^i]$ associated with it, where $1 \quad p \ q \quad 2n_i \quad 1$. Each element of the matrix s_{pq}^i represents the distance between two nodes N_p^i and N_q^i, where

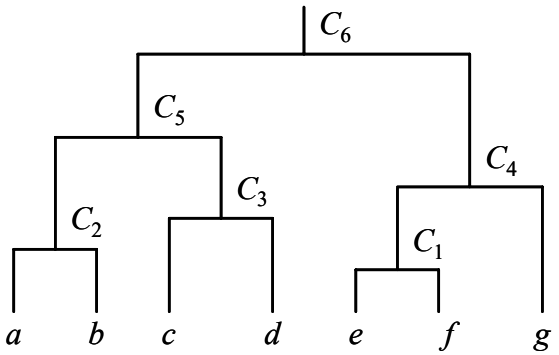


Figure 3.6 Clusters as internal nodes in a dendrogram

$$N_p^i = \begin{cases} o_p^i & \text{if } 1 \quad p \quad n_i \\ C_{p \;\; n_i}^i & \text{if } n_i < p \quad 2n_i \quad 1 \end{cases}$$

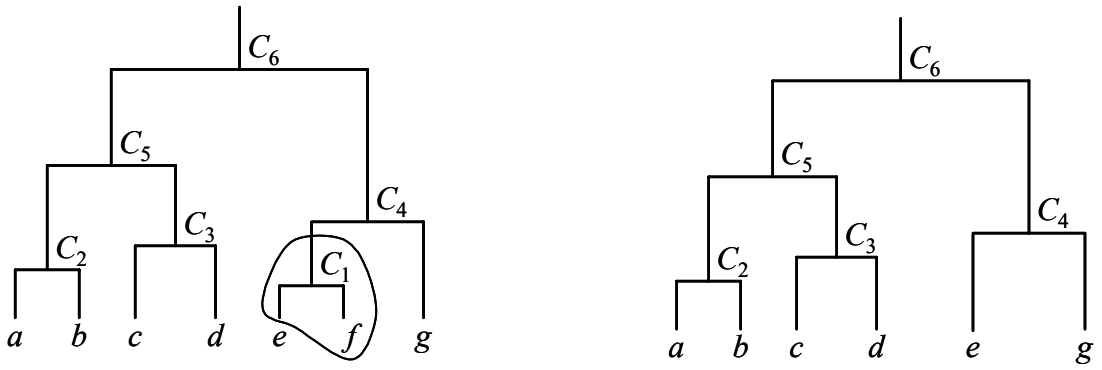
We also assume that each internal node C_j^i has an intra-cluster similarity $\;{}_j^i$ associated with it. $\;{}_j^i$ represents the similarity between the two descendants of C_j^i so that $\;{}_j^i = s_{pq}^i$, where $\;{}_l(C_j^i) = N_p^i$ and $\;{}_r(C_j^i) = N_q^i$.

With the above def nitions, we say that $\mathcal{C}_i$ is an ordered set given by $C_j^i \;\; {}_j^i \;\; {}_1 \;\; {}_j^i \;\; {}_{j+1}^i$, i.e., the internal nodes (the clusters) are numbered based on the descending order of the corresponding similarity values. For example, as shown in Figure 3.6, C_1 has the highest similarity associated with it and C_6 has the lowest. We also impose a numbering order on $\mathcal{N}_i = \;\; N_1^i \;\; N_2^i \;\;\;\;\; N_{2n_i \;\; 1}^i$, where $N_j^i = o_j^i$ for $1 \quad j \quad n_i$ and $N_j^i = C_j^i$ for $n_i < j \quad 2n_i \quad 1$.

Next, we present how two dendrograms can be ref ned by being mapped to each other. Mapping a dendrogram to another dendrogram can be accomplished by a process of *removing* and *expanding* leaves.

3.4.2 Removing Leaves of a Dendrogram

One operation used when mapping one dendrogram onto another is removal of one (or more) leaves of one dendrogram. After removal of a leaf, the edges connecting the leaf and its sibling to its immediate ancestor are removed and the immediate ancestor is replaced by the sibling of the removed leaf. Formally, we say that *removing* a leaf l from a dendrogram $\mathcal{D}$ is the function $_{l} : \mathcal{D}$ $\mathcal{D}$, where $\mathcal{D} = (\mathcal{N}\ \mathcal{E})$, $\mathcal{D} = (\mathcal{N}\ \mathcal{E})$, $\mathcal{N} = \mathcal{L}$ $\mathcal{C}$, $\mathcal{L} = \mathcal{L}$ l , $\mathcal{C} = \mathcal{C}$ (l) , $\mathcal{E} = \mathcal{E}$ $(a\ b)$ $a = (l)$ $b = (l)$ $(\ (\ (l))\ \ (l))$.



(a) Original dendrogram

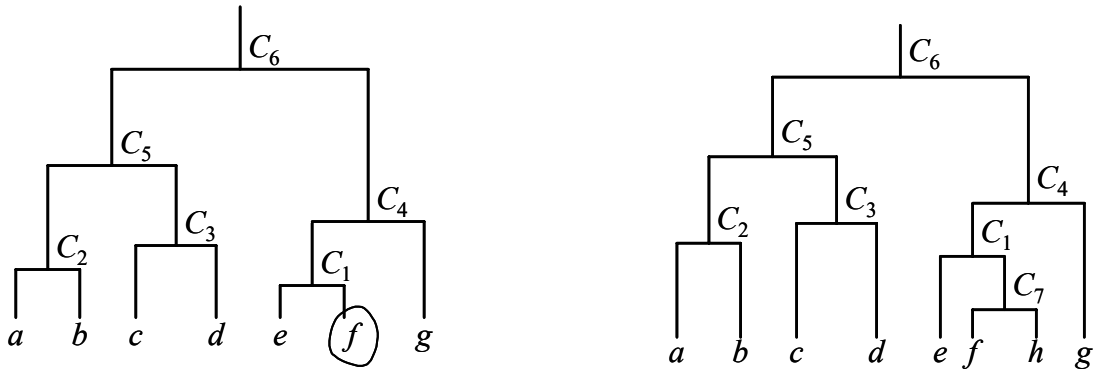
(b) Dendrogram after removing f

Figure 3.7 Removing a leaf from a dendrogram

Let us explain this with an example as shown in Figure 3.7. The dendrogram in Figure 3.7(a) consists of seven objects $a\ b\ c\ d\ e\ f\ g$. If the leaf node f is removed from the dendrogram, the edge connecting f and its the immediate ancestor C_1 and the edge connecting f's sibling e and C_1 are removed. Also, C_1 is replaced by the e. This results in the dendrogram shown in Figure 3.7(b).

3.4.3 Expanding Leaves of a Dendrogram

Another operation used for mapping one dendrogram onto another is expansion of one (or more) of its leaves. Consequently, the original leaf node is replaced by a new internal node and two leaves are added as descendants of this new internal node resulting in three new edges. One of the leaves is the original leaf and the other is a new leaf node. The edge connecting the original leaf and its original immediate ancestor is also removed. Formally, we say that *expanding* a leaf l of a dendrogram $\mathcal{D}$ with a new leaf l is the function $_{l\,l'} : \mathcal{D} \quad \mathcal{D}$, where $\mathcal{D} = (\mathcal{N}\ \mathcal{E})$ and $\mathcal{D} = (\mathcal{N}\ \mathcal{E})$. If C_p is the original ancestor of l and C_q is the new ancestor of l, then $C_q = \ l\ l\ , \mathcal{C} = \mathcal{C} \quad C_q$, $\mathcal{N} = \mathcal{L} \quad \mathcal{C}, \mathcal{L} = \mathcal{L} \quad l\ , \mathcal{E} = \mathcal{E} \quad (C_q\ l)\ (C_q\ l\ )\ (C_p\ C_q) \quad (C_p\ l)$.



(a) Original dendrogram (b) Dendrogram after expanding f with h

Figure 3.8 Expanding a leaf from a dendrogram

Let us explain this with an example as shown in Figure 3.8. The dendrogram in Figure 3.8(a) consists of seven objects $a\ b\ c\ d\ e\ f\ g$. Assume that we are expanding the node

f using h. Consequently, a new internal node $C_7 = f\ h$ is added. Three new edges are also added; an edge connecting f and C_7, an edge connecting h and C_7, and an edge connecting C_1 and C_7. Also, the edge connecting C_1 and f is removed. This results in the dendrogram shown in Figure 3.7(b).

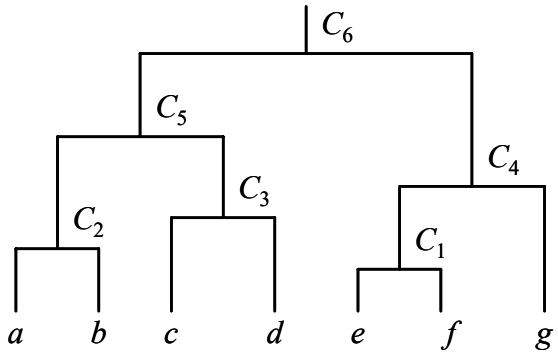
3.4.4 Cover of a Dendrogram

Let Q $\mathcal{L}$ be a set of leaf nodes of $\mathcal{D}$ and $_Q(\mathcal{D})$ be the dendrogram obtained by removing the leaves of $\mathcal{D}$ that are not in Q, i.e., $_Q(\mathcal{D}) = \ _{l_m}(\ \ _{l_2}(\ _{l_1}(\mathcal{D}))\ \ )$, where $\mathcal{L}\ \ Q = \ l_1\ l_2\ \ \ l_m$. We say that a dendrogram $\mathcal{D}$ *covers* another dendrogram $\mathcal{D}$ if $\mathcal{D}$ can be reduced to $\mathcal{D}$ by removing the leaves that are not in $\mathcal{D}$. Formally we say that $\mathcal{D}\ \ \mathcal{D}$ if $_{\mathcal{L}'}(\mathcal{D}) = \mathcal{D}$.

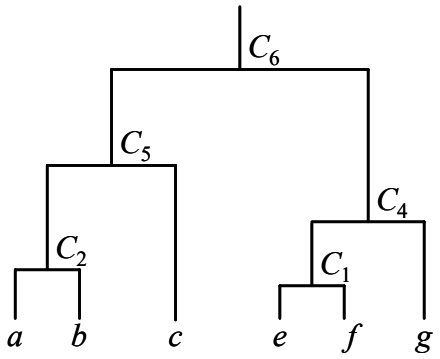
Let us consider the dendrograms shown in Figure 3.9. The dendrogram $\mathcal{D}_1$ can be reduced to $\mathcal{D}_2$ by *removing* d and d is the only node that is in $\mathcal{D}_1$ but not in $\mathcal{D}_2$. Hence, $\mathcal{D}_1$ covers $\mathcal{D}_2$. Similarly, the dendrogram $\mathcal{D}_1$ can be reduced to $\mathcal{D}_3$ by *removing* f and $\mathcal{D}_1$ covers $\mathcal{D}_3$.

3.4.5 Combining Dendrograms

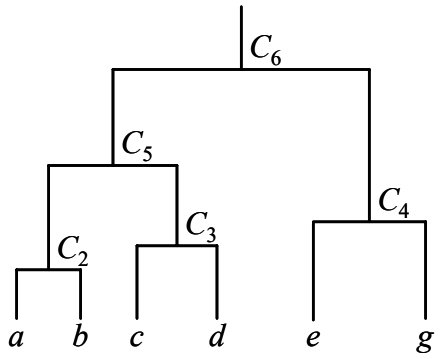
Now we outline an approach for combining two given dendrograms. As introduced earlier in this chapter, we can have three types of mapping between the objects of two heterogeneous datasets. Using the mapping, we will compute a ref nement of each dendrogram and then will apply a suitable ensemble algorithm. We def ne the *relative ref nement* of a dendrogram as ref ning it with respect to another dendrogram by *expanding* each of its



(a) $\mathcal{D}_1$



(b) $\mathcal{D}_2$ - covered by $\mathcal{D}_1$

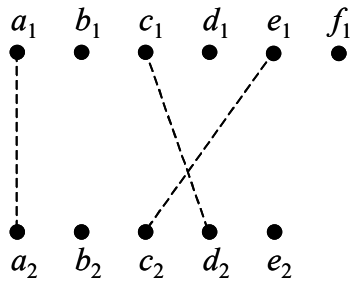


(c) $\mathcal{D}_3$ - covered by $\mathcal{D}_1$

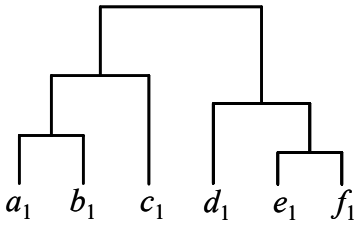
Figure 3.9 The cover of a dendrogram

leaves using the notion of mapping described in section 3.3. If we have two dendrograms $\mathcal{D}_x$ and $\mathcal{D}_y$, we will compute the refinement of $\mathcal{D}_x$ relative to $\mathcal{D}_y$ and the refinement of $\mathcal{D}_y$ relative to $\mathcal{D}_x$. The refinement of a dendrogram will satisfy the cover property, i.e., the refinement of a dendrogram will cover the dendrogram itself. Once the two dendrograms are refined, a suitable tree combining algorithm (e.g., a supertree method) can be applied on the two refined dendrograms.

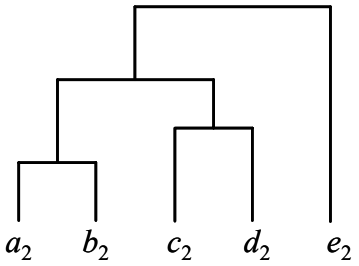
Let us explain this with an example as shown in Figure 3.10. We consider two datasets $a_1\ b_1\ c_1\ d_1\ e_1\ f_1$ and $a_2\ b_2\ c_2\ d_2\ e_2$. Figure 3.10(a) shows an assumed mapping between them. Figures 3.10(b) and 3.10(c) show the dendrograms generated from the respective datasets. Using the mapping of Figure 3.10(a), the dendrogram of Figure 3.10(b) is refined to the one shown in Figure 3.10(d) and the dendrogram of Figure 3.10(c) is refined to the one shown in Figure 3.10(e).



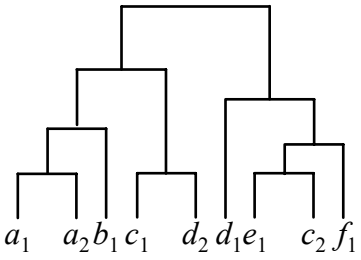
(a) A mapping between two datasets $D_1 = \ a_1\ b_1\ c_1\ d_1\ e_1\ f_1\ $ and $D_2 = \ a_2\ b_2\ c_2\ d_2\ e_2$



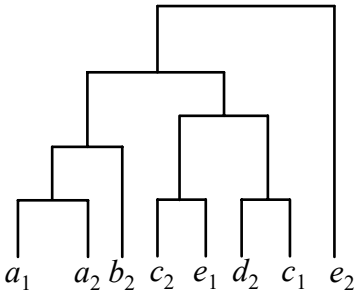
(b) The dendrogram $\mathcal{D}_1$ generated from D_1



(c) The dendrogram $\mathcal{D}_2$ generated from D_2



(d) The ref nement of $\mathcal{D}_1$



(e) The ref nement of $\mathcal{D}_2$

Figure 3.10 Computing the ref nement of two dendrograms using a given mapping

CHAPTER IV

ENSEMBLE ALGORITHMS FOR HIERARCHICAL CLUSTERING

The goal of this dissertation is to develop a framework and algorithms for discovery of a single set of clusters from multiple related heterogeneous datasets. Clustering of multiple related heterogeneous datasets can take different forms depending upon the physical characteristics of data and the mapping cardinality among datasets. To test our hypotheses, we will develop algorithms that can handle multiple datasets in two situations: (1) two datasets with non-identical features that represent the same objects, i.e., there is a perfect mapping between the objects and (2) two datasets with non-identical features and overlapping sets of objects, i.e., there is a bi-regular mapping between the objects.

In this chapter, we will present algorithms for combining multiple hierarchical clusterings generated from heterogeneous datasets. First we describe the phylogenetic tree combination methods that we have used in our experiments. Then we present an algorithm for combining multiple hierarchical clusterings into a single partitional clustering. Finally, we describe a variation of this algorithm that can more effectively combine multiple cluster hierarchies. Even though we have used two cluster hierarchies in our examples, the methods can handle multiple hierarchies.

4.1 Phylogenetic Tree Combination Methods

In addition to developing new algorithms for combining cluster hierarchies, we want to investigate the applicability of phylogenetic tree combination methods in combining multiple hierarchical clusterings. A phylogenetic tree represents the evolutionary history of some objects where the root is the ancestral object. Phylogenetic trees built for the same organisms but from different datasets may depict differing evolutionary history. These different or conf icting hypotheses about phylogenetic relationships for the same organisms can be resolved by combining the individual evolutionary trees into a single representative tree. Two major approaches for tree combination have been developed in the f eld of phylogenetic inference: consensus tree methods and supertree methods.

4.1.1 Consensus Tree

A consensus tree method [38] tries to retain the branching information from the individual trees as much as possible. There are two major types of consensus trees: *strict* and *majority rule* [7]. The strict consensus tree contains only the subtrees that are common to all the input trees. If two subtrees contain the same objects but different taxonomic relationship, then those are collapsed to appear at the same level. For example, the strict consensus tree will be $((a\ b\ c)\ d\ e)$ for three given trees $((((a\ c)\ b)\ d)\ e)$, $(((a\ b)\ c)\ (d\ e))$, and $((a\ (b\ c))\ (d\ e))$ since only the subsets $a\ b\ c$ and $a\ b\ c\ d\ e$ appear in all the trees.

The majority rule tree contains exactly those subtrees that appear in more than half of the input trees. Thus every subtree of the strict consensus tree will also be a subtree of the majority rule tree. With only two input trees, the majority consensus tree is identical to the strict tree. For example, the majority rule tree is $((a\ b\ c)\ (d\ e))$ for three given trees $((((a\ c)\ b)\ d)\ e)$, $(((a\ b)\ c)\ (d\ e))$, and $((a\ (b\ c))\ (d\ e))$ since the subsets $a\ b\ c\ d\ e$, $a\ b\ c$ and $d\ e$ appear in two or more trees out of three.

The well known phylogenetic package PHYLIP[1] uses the *greedy* consensus tree algorithm that allows additional subtrees to be included in the majority rule tree. It orders all the subtrees based on their frequency in the input trees and then inserts them in order of frequency. A tie between two subtrees having the same frequency is broken arbitrarily. If a subtree is not compatible with the already existing subtrees, then it is not included. For example, in a collection of three trees $((((a\ c)\ b)\ d)\ e)$, $(((a\ b)\ c)\ (d\ e))$, and $((a\ (b\ c))\ (d\ e))$, $a\ b\ c\ d\ e$ and $a\ b\ c$ appear three times, $d\ e$ appear two times, and $a\ b\ c\ d$, $a\ b$, $b\ c$, and $a\ c$ appear only once. The resulting greedy consensus tree will be $((a\ (c\ b))\ (d\ e))$.

4.1.2 Supertree

The consensus tree methods cannot be used with input trees where the trees consist of some common objects, but not all. This kind of trees can be combined with another important class of methods called “supertree” methods. A supertree is def ned as a tree that

[1] `http://evolution.genetics.washington.edu/phylip.html`

is built from input trees containing overlapping sets of objects. Some common approaches for building a supertree are the supermatrix approach [48], matrix representation using parsimony [44], pruned tree approach [22], etc.

Semple and Steel [49] presented a graph-based supertree algorithm that works even with incompatible trees and can scale up to large datasets. It constructs a graph where two nodes (objects) are connected if they are in a proper cluster in at least one of the input trees. The edges having weight equal to the number of input trees are then collapsed and the modif ed graph is partitioned using the minimum weight cut edges. The set of input trees is then induced by each resulting component and the algorithm is called recursively with each of these sets of subtrees until each set of subtrees contains fewer than three leaves. A bottom-up approach is used to build the supertree by adding the trees at the end of each recursive call to a new common root.

For example, let us consider two trees $((((a\ c)\ b)\ d)\ e)$ and $(((a\ b)\ c)\ (d\ e))$. A graph will be constructed having the edges $a\ b$, $a\ c$, $a\ d$, $b\ c$, $b\ d$, $c\ d$, and $d\ e$. Only the edges $a\ b$ and $a\ c$ will have edge-weights of 2 since those appear in both the clustering and they will be collapsed. This will result in the edges $a\ b$, $a\ c$, $a\ d$, $b\ c$, $b\ d$, and $c\ d$ being replaced by a single edge $abc\ d$ having an edge-weight of 1. Then the minimum weight cut set will be computed from the modif ed graph producing three components having the vertices $a\ b\ c$, d , and e respectively. The vertices $a\ b\ c$ will induce the subtrees $((a\ b)\ c)$ and $((a\ c)\ b)$. The above process will be recursively applied to these subtrees resulting in three components

having the vertices a , b , and c respectively. The bottom-up supertree building process will first generate $(a\ b\ c)$ and finally $((a\ b\ c)\ d\ e)$.

4.2 Generating Partitional Clusters from Multiple Cluster Hierarchies

In a clustering application involving multiple related heterogeneous datasets, it may be desirable to generate a partitional clustering as a result of combining the results of the individual hierarchical clusterings. For example, it may not be possible or feasible to access the original dataset after hierarchical clustering is performed. A phylogenetic tree combination method can be used to combine the resulting dendrograms to generate a single representative dendrogram [27] and then a bottom-up or top-down strategy can be used to cut the output dendrogram and extract the desired number of partitional clusters.

However, this approach has some shortcomings. The phylogenetic tree combination methods focus on taxonomic structures of the individual trees to determine the taxonomic relationship of the objects in the final tree and they do not focus on a quantitative measure of how closely the data objects are related. In addition, some of these algorithms, e.g., the consensus tree method, cannot deal with datasets that have some common objects but not all.

Also, it may be difficult to generate the desired number of partitional clusters from the dendrogram generated by the supertree or consensus tree methods. This is because the output dendrogram will typically have clusters (internal nodes) consisting of more than two sub-clusters (descendants). The consensus tree method may create a lot of singleton

clusters with objects that cannot be resolved. The supertree method merges more than two sub-clusters when there is lack of compatibility between the sub-clusters. For example, consider the dendrograms shown in Figure 4.1(a) and 4.1(b). The supertree algorithm will generate the dendrogram shown in Figure 4.1(c). It is not possible to extract two partitional clusters from the dendrogram in Figure 4.1(c). One solution can be four clusters, $a\ b\ c$, e, $d\ g\ h$, and f. Another solution can be one big cluster, $a\ b\ c\ d\ e\ f\ g\ h$.

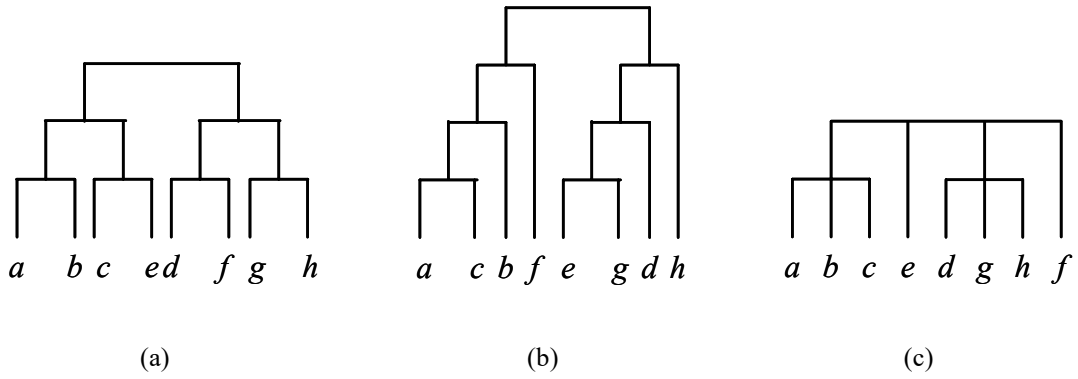


Figure 4.1 Problem with the dendrogram generated by the supertree algorithm

It is important, therefore, to develop methods for producing partitional clusters as dendrograms are combined. In this section, we present a graph-based method EPaCH (Ensemble method for generating Partitional clusters from multiple Cluster Hierarchies) for combining multiple cluster hierarchies to yield a set of partitional clusters. Instead of extracting partitional clusters from a combined dendrogram, this method directly generates partitional clusters from two or more dendrograms by capturing the cluster membership of data objects from multiple dendrograms into the adjacency relationship of a single graph.

4.2.1 Algorithm Overview

In a hierarchical clustering, each node in the tree corresponds to a cluster. If one focuses on two data objects, both will be members of the cluster represented by the root, but the objects may also both be elements of other sub-clusters. In a dendrogram, all the proper sub-clusters (clusters other than the root) become part of successively coarser sub-clusters as one travels from the leaves (representing singleton clusters) toward the root.

EPaCH works by first generating a graph using the strength of association of each pair of objects in the dendrograms. The purpose is to bring together the objects that are strongly associated with each other into the form of a subgraph. The EPaCH algorithm is based on the assumption that *the larger the number of common sub-clusters two data objects belong to, the stronger their association*. The strength of association is represented as weighted edges between objects in a graph.

For example, let us consider the dendrogram of Figure 4.2. There are six objects, i.e., $a\ b\ c\ d\ e$ and f in the dendrogram. There are five sub-clusters, i.e., $a\ b$, $a\ b\ c$, $a\ b\ c\ d$, $a\ b\ c\ d\ e\ f$, and $e\ f$. According to our assumption, a and b are expected to be more closely associated than a and c or c and d or e and f.

Definition 1 Let $\mathcal{C} = C_1\ C_2 \quad C_{n\ 1}$ be the set of nested clusters represented by the dendrogram. Let $A_i = C_p\ C_i \quad C_p$ be the ancestral set of C_i, i.e., the set of sub-clusters that contain C_i. We define d_i to be the depth of a sub-cluster C_i such that $d_i = A_i$. The depth of the root is 0, i.e., $d_{n\ 1} = 0$.

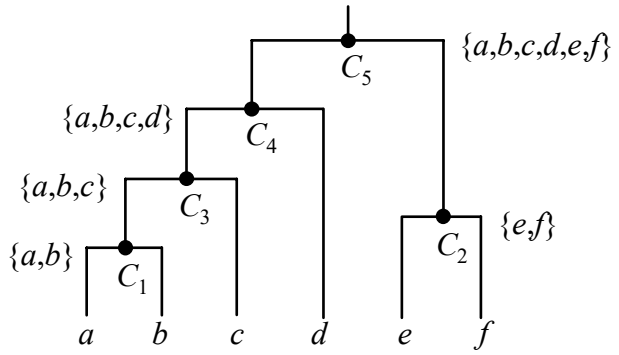


Figure 4.2 Clusters as internal nodes in a dendrogram

Given the def nition of d_i, we can restate our assumption by saying that two objects that belong to a sub-cluster which is at a lower level (greater depth) in the tree are more closely associated than those that are both members of sub-clusters at a higher level (lesser depth). For example, let us consider the dendrogram of Figure 4.2. According to our restated assumption, a and b are expected to be more closely associated than e and f.

Lemma IV.1 Let a, b, c, and d be four objects and C_i and C_j be the nearest common ancestors of $a\ b$ and $c\ d$ respectively. Then, $d_i > d_j$ $_{ab} > {}_{cd}$, where $_{ab}$ is the strength of association between a and b.

Consequently, a sub-cluster that contains two data objects and is at a lower level of the dendrogram will contribute more to the strength of association of the two objects than a sub-cluster that contains the same two objects and is at a higher level of the dendrogram. For example, in Figure 4.2, the sub-cluster C_1, i.e., $a\ b$ will contribute more to the strength of association of a and b compared to the sub-cluster C_3, i.e., $a\ b\ c$ and sub-cluster C_4, i.e., $a\ b\ c\ d$.

Based on this idea, we calculate the strength of association for each pair of data objects from each dendrogram and then add the individual strengths of association to calculate the combined strength. We assume that the strength of association can be considered to be a measure of proximity of the two objects.

Def nition 2 Let a and b be two data objects and $\ ^{i}_{ab}$ be the strength of association of a and b in dendrogram $\mathcal{D}_i$. We def ne $\ ^{i}_{ab}$ as the sum of the normalized depths of the proper clusters that both a and b belong to. If d^i_{max} is the maximum depth of a proper cluster in a dendrogram $\mathcal{D}_i$ and d^i_j is the depth of the cluster C^i_j, then $\ ^{i}_{ab}$ is def ned as:

$$^{i}_{ab} = \sum_{a\ b\ C^i_j} d^i_j \ \ d^i_{max}$$

Lemma IV.2 If $\ ^{i}_{ab}$ is the strength of association of a and b in dendrogram $\mathcal{D}_i$, then there is a non-zero lower bound and a linear upper bound on $\ ^{i}_{ab}$.

Proof 1 The strength of association of two objects could be zero if the only sub-cluster they belong to is the root of the dendrogram which is not a proper cluster. Since $\ ^{i}_{ab}$ only considers the proper clusters, it cannot be zero. Two objects can have the minimum possible strength of association when the only sub-cluster they belong to is at a depth of one and the corresponding strength of association is $1 \ \ d^i_{max}$. Two objects can have the

maximum possible strength of association when the lowest sub-cluster they belong to is at a depth of d^i_{max} and the corresponding strength of association is:

$$\begin{aligned}\sum_{j=1}^{d^i_{max}} \frac{j}{d^i_{max}} &= \frac{1}{d^i_{max}} \sum_{j=1}^{d^i_{max}} j \\ &= \frac{1}{d^i_{max}} \frac{d^i_{max}(d^i_{max}+1)}{2} \\ &= \frac{d^i_{max}+1}{2}\end{aligned}$$

Thus, the lower bound and the upper bound can be written as,

$$\frac{1}{d^i_{max}} \quad {}^{i}_{ab} \quad \frac{d^i_{max}+1}{2}$$

We normalize the contribution of each sub-cluster to the strength of association since we want to ensure that the strength of association of two objects for each dendrogram is appropriately weighted. For example, let us assume that two objects a and b that are contained in two dendrograms $\mathcal{D}_1$ and $\mathcal{D}_2$. The lowest sub-cluster that a and b belong to in both $\mathcal{D}_1$ and $\mathcal{D}_2$ has a depth of 2. Also, the maximum depth of $\mathcal{D}_1$ and $\mathcal{D}_2$ are 2 and 3 respectively. If the contribution of each sub-cluster to the strength of association is not normalized, then the strength of association of a and b will be $2 + 2 = 4$. This is not appropriately weighted since $\mathcal{D}_1$ is expected to make more contributions to the strength of association of a and b compared to $\mathcal{D}_2$. But if we normalize the contribution of each sub-cluster as we def ned, we do not encounter this problem.

The strengths of association are used to construct an undirected weighted graph where the vertices correspond to data objects and two vertices are connected by an edge having weight equal to the strength of association of the two corresponding data objects. We call

this the *cluster association graph*. Once the cluster association graph has been constructed, a graph partitioning algorithm is used to extract clusters from the graph. We have used the k-way graph partitioning algorithm given by Karypis and Kumar [32].

4.2.2 An Illustrative Example

Let us consider the dendrogram $\mathcal{D}_1$ of Figure 4.3 that has a maximum depth of 3. The data objects a and b are in the proper clusters $a\ b$, $a\ b\ c$, and $a\ b\ c\ d$. The depths of these proper clusters are 3, 2, and 1 respectively. Consequently, their contributions to the strength of association of a and b are $\frac{3}{3}$, $\frac{2}{3}$, and $\frac{1}{3}$ respectively. The individual contributions are used to calculate the overall strength of association for a and b, i.e., $\overset{1}{}_{ab}$ $= \frac{3}{3} + \frac{2}{3} + \frac{1}{3} = 2$. Table 4.1 shows the strength of association for each possible pair of objects in both dendrograms of Figure 4.3.

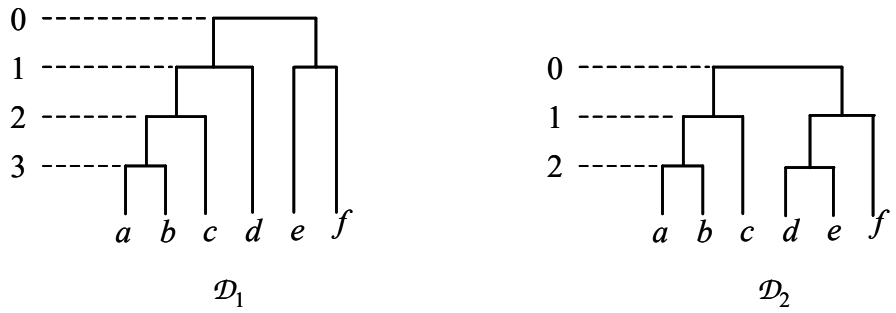


Figure 4.3 The depths of sub-clusters in dendrograms

The combined strength of association of each pair of data objects in both dendrograms is used to construct the cluster association graph shown in Figure 4.4. Each node corre-

Table 4.1 Strengths of association for different pairs of objects in Fig. 4.3

	$\mathcal{D}_1$	$\mathcal{D}_2$	Combined
ab	$\frac{3}{3}+\frac{2}{3}+\frac{1}{3}=2$	$\frac{2}{2}+\frac{1}{2}=1\ 5$	3.5
ac	$\frac{2}{3}+\frac{1}{3}=1$	$\frac{1}{2}=0\ 5$	1.5
ad	$\frac{1}{3}=0\ 33$	0	0.33
ae	0	0	0
af	0	0	0
bc	$\frac{2}{3}+\frac{1}{3}=1$	$\frac{1}{2}=0\ 5$	1.5
bd	$\frac{1}{3}=0\ 33$	0	0.33
be	0	0	0
bf	0	0	0
cd	$\frac{1}{3}=0\ 33$	0	0.33
ce	0	0	0
cf	0	0	0
de	0	$\frac{2}{2}+\frac{1}{2}=1\ 5$	1.5
df	0	$\frac{1}{2}=0\ 5$	0.5
ef	$\frac{1}{3}=0\ 33$	$\frac{1}{2}=0\ 5$	0.83

sponds to a data object and the combined strength of association of each pair is used as the weight of the edge between the corresponding nodes. For example, $\overset{1}{}_{ab} = 2$ and $\overset{2}{}_{ab} = 1\ 5$. Consequently, a and b will be connected by an edge having a weight of 3.5. Let us consider another pair of objects a and f. It can be seen from Table 4.1 that $\overset{1}{}_{af} = \overset{2}{}_{af} = 0$. Consequently, a and f will not be connected since the edge weight is 0.

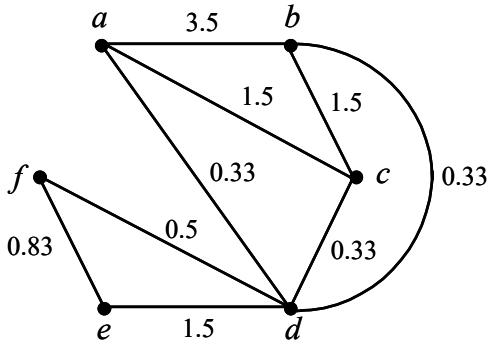


Figure 4.4 Cluster association graph

For graph partitioning, we have used the METIS[2] software. METIS has two graph partitioning programs, `pmetis` (based on a two-way partitioning algorithm [30]) and `kmetis` (based on a k-way partitioning algorithm [32]). The METIS programs require the edge weights to be integer values. If we simply convert the edge weights in Figure 4.4 to integers, it will result in information loss. Rather, we decided to normalize the combined strength of association of each pair of objects. We have proved previously that the strength of association of two objects in a particular dendrogram is bounded on the

[2] `http://www-users.cs.umn.edu/~karypis/metis/metis/index.html`

lower side by the reciprocal of the maximum dendrogram depth. Hence, we normalize $_{ab}$ by the harmonic sum of the maximum dendrogram depths, i.e.,

$$_{ab} = \quad _{ab} \qquad \frac{1}{d^i_{max}}$$
$$\mathcal{D}_i$$

This normalization results in a minimum edge weight of one. If we apply this normalization, then the cluster association graph shown in Figure 4.4 will become the one shown in 4.5. Note that the edges $a\ d$, $b\ d$, and $c\ d$ have been dropped in the process. If the desired number of clusters is two, then `kmetis` will give us the clusters $a\ b\ c$ and $d\ e\ f$ as shown with the dotted ellipses in 4.5.

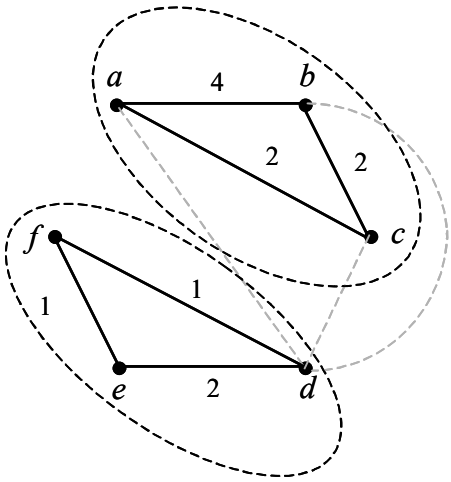


Figure 4.5 Normalized cluster association graph

4.2.3 The EPaCH Algorithm

The input to the EPaCH algorithm is a set of m dendrograms $\mathcal{D}_1\ \mathcal{D}_2\ \ \ \mathcal{D}_m$ and the output of the algorithm is a set of k disjoint partitional clusters $C_{p1}\ C_{p2}\ \ \ C_{pk}$. We will use $\mathcal{L}$ in our algorithm to represent the overall set of objects where $\mathcal{L} = \ \ _{i=1}^{m} \mathcal{L}_i$.

Def nition 3 Let $G = (V\ E)$ be a weighted undirected graph where V is the set of vertices and E is the set of undirected edges. Let $w : E\ \ \ \mathbf{R}^+$ be the weight function associated with the graph where each weight is a positive real number, so that for $e\ \ \ E$, $w(e)\ \ \ \mathbf{R}^+$.

The algorithm is described in Figure 4.6. The algorithm f rst computes the strength of association for each pair of objects (steps 2-11). Then it computes the cluster association graph (steps 13-20). Finally, in step 22, it invokes a graph partitioning algorithm to generate the clusters. Each connected component of the graph will represent a partitional cluster.

4.2.4 Complexity of EPaCH

The complexity of EPaCH is essentially determined by the complexity of the operations that compute the strength of association of each pair of objects. In the worst case, EPaCH will have to deal with a cluster hierarchy that has a maximum depth of $n\ \ \ 2$, where n is the number of objects in the hierarchy. This will happen when just one node is added to the next upper level sub-cluster as one travels from the deepest sub-cluster to the all inclusive root cluster. For example, the cluster hierarchy shown in Figure 4.7 represents the worst-case with six objects.

```
Input:  A set of m dendrograms D_1 D_2 ... D_m
Output:  A set of k disjoint partitional clusters C_p1 C_p2 ... C_pk
 1.  // compute strengths of association
 2.  for each pair of objects a b in L do
 3.      ab = 0
 4.  normalizer = 0
 5.  for each dendrogram D_i do
 6.     for each cluster C_j^i do
 7.        for each pair a b in C_j^i do
 8.           ab = ab + d_j^i / d_max^i
 9.     normalizer = normalizer + 1 / d_max^i
10.  for each pair of objects a b in L do
11.      ab = ab / normalizer
12.  // compute cluster association graph
13.  V = {}
14.  for each object a in L do
15.     V = V + a
16.  for each object a in L do
17.     for each object b in L do
18.        if a = b and ab > 0 then
19.           e = (a, b, w(e) = ab)
20.           E = E + e
21.  // partition graph
22.  C_p1 C_p2 ... C_pk = PartitionGraph(k, G)
```

Figure 4.6 The EPaCH algorithm

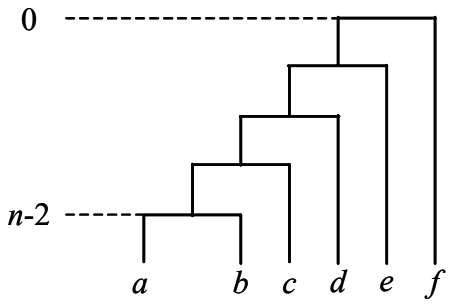


Figure 4.7 A cluster hierarchy with worst-case complexity for EPaCH

The total number of different pairs of objects in the entire hierarchy is essentially the number of 2-combinations of an n-element set, i.e., $\binom{n}{2} = n!\ (2!(n\ \ 2)!) = n(n\ \ 1)\ 2$. Intuitively, the worst-case complexity will be $O(n^3)$ since the upper bound on the number of required node-visits for a pair of objects is $n\ \ 1$. An exact analysis of the worst-case complexity follows.

In a worst-case hierarchy, the computation of the strength of association of the two objects belonging to the lowest level sub-cluster will require $n\ \ 1$ node visits, the computation of the strength of association of the two objects belonging to the sub-cluster just above the lowest level one will require $n\ \ 2$ node visits, and so on. The total number of different pairs of objets in the entire hierarchy is essentially the number of 2-combinations of an n-element set, i.e., $\binom{n}{2} = n!\ (2!(n\ \ 2)!) = n(n\ \ 1)\ 2$. Out of these different pairs, one will require $n\ \ 1$ node visits, two will require $n\ \ 2$ node visits, three will require $n\ \ 3$ node visits, and so on. For example, in the hierarchy of Figure 4.7, the computation of strength of association for only one pair of objects, i.e., $a\ b$, will require $n\ \ 1$ node

visits. The computation of strength of association for two pairs of objects, i.e., $a\ c$ and $b\ c$, will require $n \quad 2$ node visits, and so on.

In the worst-case situation, the total number of node visits can be written as:

$$
\begin{aligned}
&= \sum_{i=1}^{n\ \ 1} i(n \quad i) \\
&= \sum_{i=1}^{n\ \ 1} [(n \quad i) + (i \quad 1)(n \quad i)] \\
&= \sum_{i=1}^{n\ \ 1} (n \quad i) + \sum_{i=1}^{n\ \ 1} (i \quad 1)(n \quad i) \\
&= \sum_{i=1}^{n\ \ 1} (i) + \sum_{i=2}^{n\ \ 1} (i \quad 1)(n \quad i) \\
&= \sum_{i=1}^{n\ \ 1} (i) + \sum_{i=2}^{n\ \ 1} [(n \quad i) + (i \quad 2)(n \quad i)] \\
&= \sum_{i=1}^{n\ \ 1} (i) + \sum_{i=2}^{n\ \ 1} (n \quad i) + \sum_{i=2}^{n\ \ 1} (i \quad 2)(n \quad i) \\
&= \sum_{i=1}^{n\ \ 1} (i) + \sum_{i=1}^{n\ \ 2} (i) + \sum_{i=3}^{n\ \ 1} (i \quad 2)(n \quad i) \\
&\ \ \vdots \\
&= \sum_{i=1}^{n\ \ 1} (i) + \sum_{i=1}^{n\ \ 2} (i) + \sum_{i=1}^{n\ \ 3} (i) + \quad + \sum_{i=1}^{2} (i) + \sum_{i=1}^{1} (i) \\
&= \frac{(n \quad 1)n}{2} + \frac{(n \quad 2)(n \quad 1)}{2} + \frac{(n \quad 3)(n \quad 2)}{2} + \quad + \frac{2 \quad 3}{2} + \frac{1 \quad 2}{2} \\
&= \frac{1}{2}\ (n \quad 1)n + (n \quad 2)(n \quad 1) + (n \quad 3)(n \quad 2) + \quad + 2 \quad 3 + 1 \quad 2 \\
&= \frac{1}{2}\ 1 \quad 2 + 2 \quad 3 + \quad + (n \quad 2)(n \quad 1) + (n \quad 1)n \\
&= \frac{1}{2}\ \frac{n(n \quad 1)(n \quad 2)}{3} \\
&= \frac{n(n^2 \quad 1)}{6}
\end{aligned}
$$

$$= \frac{(n^3 \quad n)}{6}$$

$$= O(n^3)$$

If there are k hierarchies, the complexity will be $O(kn^3)$. Since $n \quad k$, we can take the complexity to be $O(n^3)$.

EPaCH will have best-case complexity when the cluster hierarchy has a maximum depth of $\lg n \quad 1$.[3] This will happen when the hierarchy is fully balanced as shown in Figure 4.8. In a best-case hierarchy, the computation of the strength of association of the two objects belonging to the lowest level sub-cluster will require $\lg n$ node visits, the computation of the strength of association of the two objects belonging to the sub-cluster just above the lowest level will require $\lg n \quad 1$ node visits, and so on. Out of the total $n(n \quad 1)\ 2$ different pairs of objects in the entire hierarchy, $n\ 2$ will require $\lg n$ node visits, n will require $\lg n \quad 1$ node visits, $2n$ will require $\lg n \quad 2$ node visits, and so on.

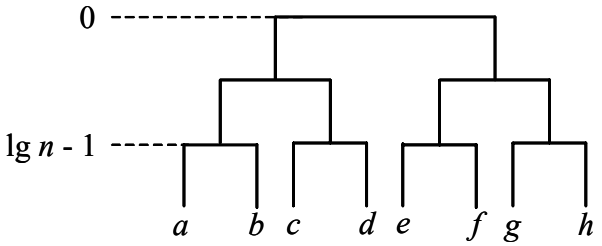


Figure 4.8 A cluster hierarchy with best-case complexity for EPaCH

[3] In this dissertation, we use lg to represent $\log_2$.

For example, in the hierarchy of Figure 4.8, the computation of strength of association for all the lowest level pairs of objects, i.e., $a\ b$, $c\ d$, $e\ f$, and $g\ h$, will require $\lg n$ node visits. The computation of strength of association for eight pairs of objects belonging to the second lowest level, i.e., $a\ c$, $a\ d$, $b\ c$, $b\ d$, $e\ g$, $e\ h$, $f\ g$, and $f\ h$, will require $\lg n \quad 1$ node visits, and so on.

In the best-case situation, the total number of node visits can be written as:

$$\begin{aligned}
& \frac{n}{2}\lg n + n(\lg n \quad 1) + 2n(\lg n \quad 2) + \quad + 2^{\lg n \ \ 2}n \ \ 1 \\
= \ & \frac{n}{2}\lg n + n(\lg n \quad 1) + 2n(\lg n \quad 2) + \quad + 2^{\lg n \ \ 2}n(\lg n \quad (\lg n \quad 1)) \\
= \ & \frac{n}{2}\lg n + n\lg n + 2n\lg n + \quad + 2^{\lg n \ \ 2}n\lg n \\
& n + 4n + \quad + 2^{\lg n \ \ 2}(\lg n \quad 1)n \\
= \ & \frac{n}{2}\lg n + n\lg n + 2n\lg n + \quad + 2^{\lg n \ \ 2}n\lg n \\
& n \ \ 1 + 4 + \quad + 2^{\lg n \ \ 2}(\lg n \quad 1)
\end{aligned}$$

Since

$$\begin{aligned}
& \frac{n}{2}\lg n + n\lg n + 2n\lg n + \quad + 2^{\lg n \ \ 2}n\lg n \\
= \ & \frac{n}{2}\lg n \ \ 1 + 2 + 2^2 + \quad + 2^{\lg n \ \ 1} \\
= \ & \frac{n}{2}\lg n \ \ 2^0 + 2^1 + 2^2 + \quad + 2^{\lg n \ \ 1} \\
= \ & \frac{n}{2}\lg n \ \ 2^{\lg n \ \ 1+1} \quad 1 \\
= \ & \frac{n}{2}\lg n \ \ 2^{\lg n} \quad 1 \\
= \ & \frac{n}{2}\lg n\,(n \quad 1)
\end{aligned}$$

and

$$
\begin{aligned}
& 1 + 4 + \quad + 2^{\lg n \ 2}(\lg n \quad 1) \\
= \ & 1 + 2 \ \ 2 + \quad + 2^{\lg n \ 2}(\lg n \quad 1) \\
= \ & 2^0 \ \ 1 + 2^1 \ \ 2 + 2^2 \ \ 3 + \quad + 2^{\lg n \ 2}(\lg n \quad 1) \\
= \ & 2[2^0 \ \ 1 + 2^1 \ \ 2 + 2^2 \ \ 3 + \quad + 2^{\lg n \ 2}(\lg n \quad 1)] \\
& [2^0 \ \ 1 + 2^1 \ \ 2 + 2^2 \ \ 3 + \quad + 2^{\lg n \ 2}(\lg n \quad 1)] \\
= \ & [2 + 2^2 \ \ 2 + 2^3 \ \ 3 + \quad + 2^{\lg n \ 2}(\lg n \quad 2) + 2^{\lg n \ 1}(\lg n \quad 1)] \\
& [2^0 \ \ 1 + 2^1 \ \ 2 + 2^2 \ \ 3 + \quad + 2^{\lg n \ 2}(\lg n \quad 1)] \\
= \ & [1 \quad 1 + 2^1 \ \ 2 \quad 2^1 + 2^2 \ \ 3 \quad 2^2 + \quad + \\
& 2^{\lg n \ 2}(\lg n \quad 1) \quad 2^{\lg n \ 2} + 2^{\lg n \ 1} \lg n \quad 2^{\lg n \ 1}] \\
& [2^0 \ \ 1 + 2^1 \ \ 2 + 2^2 \ \ 3 + \quad + 2^{\lg n \ 2}(\lg n \quad 1)] \\
= \ & 2^{\lg n \ 1} \lg n \quad [1 + 2 + 2^2 + \quad + 2^{\lg n \ 2} + 2^{\lg n \ 1}] \\
= \ & \frac{2^{\lg n}}{2} \lg n \quad [2^{\lg n \ 1+1} \quad 1] \\
= \ & \frac{n}{2} \lg n \quad 2^{\lg n} + 1 \\
= \ & \frac{n}{2} \lg n \quad n + 1
\end{aligned}
$$

the best-case complexity of EPaCH is:

$$
\begin{aligned}
& \frac{n}{2} \lg n(n \quad 1) \quad n \ \ \frac{n}{2} \lg n \quad n + 1 \\
= \ & \frac{n^2}{2} \lg n \quad \frac{n}{2} \lg n \quad \frac{n^2}{2} \lg n + n^2 \quad n \\
= \ & O(n^2)
\end{aligned}
$$

To compute the average-case complexity, we need to find an upper bound on the average depth of each cluster hierarchy. We will use a simple intuitive probabilistic model for this purpose. Let us assume that x is the deepest node in the hierarchy. To find the expected depth of x, we need to consider the number of nodes that can appear in the path from x to the root. If we count the probabilities for each one of the n nodes, the expected number of nodes in a random permutation of n nodes can be written as:

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \quad + \frac{1}{n}$$

Since each node can appear either as a left child or a right child along the path from x to the root, the expected depth of the cluster hierarchy can be written as:

$$2 \quad \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \quad + \frac{1}{n}$$

Let us assume that $n = 2^k$, where k is an integer. Then, the upper bound on average depth is:

$$\begin{aligned}
& 2 \quad \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \quad + \frac{1}{n} \\
= \; & 2 \quad \frac{1}{1} + \quad \frac{1}{2} + \frac{1}{3} \quad + \quad \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \quad + \quad + \quad \frac{1}{n} \\
& 2 \quad \frac{1}{1} + \quad \frac{1}{2} + \frac{1}{2} \quad + \quad \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} \quad + \quad + \quad \frac{1}{n} \\
= \; & 2 \quad \frac{1}{2^0} + \quad \frac{1}{2^1} + \frac{1}{2^1} \quad + \quad \frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^2} \quad + \quad + \quad \frac{1}{n} \\
= \; & 2 \sum_{i=0}^{\lg n} \sum_{j=0}^{2^i \; 1} \frac{1}{2^i} \\
= \; & 2 \sum_{i=0}^{\lg n} 1 \quad = \quad 2 \lg n
\end{aligned}$$

So, we can see the upper bound on the average depth of a cluster hierarchy is $O(\lg n)$. Intuitively, we can say the average-case complexity of EPaCH will be $O(n^2 \lg n)$ since the upper bound on the number of node-visits two objects may require is $\lg n$ and there are $n(n \quad 1)\ 2$ possible pairs of objects. We can conclude that the average-case complexity of EPaCH is no worse than the complexity of building a single cluster hierarchy using the average-link agglomerative algorithm.

4.3 EPaCHW - A Modif cation of EPaCH

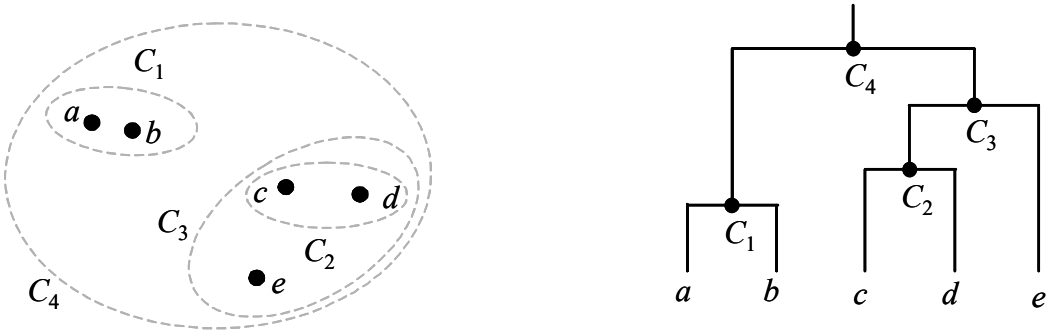
The EPaCH algorithm does not consider the underlying similarity measures of the input dendrograms. It works by considering only the taxonomic structure of the dendrograms. EPaCH was based on the assumption that the strength of association of two objects is dependent on the depths of the sub-clusters these two objects belong to. Although this notion can capture how closely two objects are related, the performance of EPaCH can be improved by utilizing the underlying similarity measures. This will result in more informative combined dendrogram. In this section, we present a modif cation of EPaCH where the similarity measures associated with each cluster are used to combine the dendrograms. We call this algorithm EPaCHW (EPaCH-weighted).

4.3.1 Potential Problem with EPaCH

Let us f rst try to explain a potential problem with EPaCH. When the strength of association is computed for a pair of objects, the only consideration is the depths of the sub-clusters containing these objects, or equivalently, the number of sub-clusters containing

these objects. This approach may result in two objects getting a lower value of strength of association than another pair having a lower value of actual similarity (based on the metric used to compute the original dendrogram).

Let us consider the five data objects in two dimensional space as shown in Figure 4.9(a) and the corresponding dendrogram computed by an average link agglomerative method as shown in Figure 4.9(b). Even though objects a and b have the maximum pairwise similarity and are combined into a sub-cluster in the very first iteration of the agglomerative algorithm, EPaCH will assign a strength of association of 1/2 for $a\ b$ and a strength of association of 3/2 for $c\ d$. If we can incorporate the similarity measure associated with each sub-cluster, this can be overcome and the contribution of each sub-cluster to the strength of association of each pair of objects can be more appropriately weighted.



(a) Nested Clustering (b) Dendrogram

Figure 4.9 Problem with EPaCH

4.3.2 Overview of EPaCHW

Each internal node of a dendrogram is associated with a similarity value that represents the intra cluster similarity for the corresponding cluster. Let $\ ^{i}_{j}$ be the intra cluster similarity for C^i_j, i.e., $\ ^{i}_{j}$ is the similarity between its two descendants. We argue that the strength of association of two objects should be correlated to the intra-cluster similarity of the clusters to which they belong, i.e., if a pair of objects belong to a cluster having a higher similarity, then their strength of association should be higher than the strength of association of two objects belonging to a cluster having a smaller similarity.

For EPaCHW, we will redef ne the strength of association of two object with respect to cluster C^i_j. Note that if average-link method is used to merge two sub-clusters, then the intra-cluster similarity of a sub-cluster gives an estimate of the similarity between two objects in the sub-cluster. If the sub-cluster consists of only two objects, then the intra-cluster similarity is essentially the actual similarity between them. As one travels upward from the lowest sub-clusters that two objects belong to, the estimate of the similarity between the two objects decrease monotonically. This is in accordance with our earlier assumption for EPaCH that sub-clusters higher up the dendrogram should contribute less to the strength of association of two objects.

To compute the strength of association of two objects with respect to a particular sub-cluster C^i_j, instead of just taking $d^i_j\ \ d^i_{max}$ as we did in EPaCH, we will multiply the intra-

cluster similarity $\frac{i}{j}$ with $d^i_j \ d^i_{max}$. That requires us to redef ne the strength of association as:

$$\frac{i}{ab} = \frac{d^i_j \quad \frac{i}{j}}{d^i_{max}}$$
$$a\ b \ C^i_j$$

Accordingly we will need to modify line 8 of EPaCH (Figure 4.6). Since the contribution of each sub-cluster is weighted by the intra-cluster similarity, we will normalize the combined strength of association using the average intra-cluster similarity.

Since a dendrogram already includes the intra-cluster similarity associated with each sub-cluster, no extra computation is needed to compute that. Therefore, the complexity of EPaCHW will be the same as EPaCH, i.e., $O(n^2)$ in best-case, $O(n^2 \lg n)$ in average-case and $O(n^3)$ in worst-case. We argue that EPaCHW computes the strength of association of two objects more appropriately compared to EPaCH. It is expected to capture the association of objects in the input dendrograms better and this will result in a more informative partitional clusters as output.

CHAPTER V

EXPERIMENTAL RESULTS

In Chapter IV, we presented the EPaCH family of algorithms that are used to generate a single set of partitional clusters from multiple heterogeneous datasets. Clustering of multiple heterogeneous datasets can take different forms depending upon the physical characteristics of data and the type of mapping between datasets. In this chapter, we present an evaluation of the effectiveness of the EPaCH algorithms in two situations. In the first situation, two datasets are clustered that represent the same set of objects. In the second situation, the datasets share some but not all objects - we call this situation dealing with overlapping datasets. In this chapter, we describe the datasets used for the evaluation, the experimental design, and the evaluation methods used to compare the effectiveness of algorithms. We present the experimental results and provide an analysis of the results. We also present a modification of one of the cluster evaluation measures, namely the Davies-Bouldin Index.

5.1 Datasets

A document clustering problem was selected as the application domain. We used a document collection compiled by Wang [60] consisting of ten thousand journal abstracts that belong to ten different subject areas. The abstracts were divided evenly into five non-

overlapping subsets of two thousand where each subset contained two hundred abstracts from each category. Basic natural language preprocessing steps were applied to all documents including sentence parsing, tokenization, morphological analysis and part-of-speech tagging using the software developed by Wang [60].

Feature vectors were generated for each abstract using four different preprocessing methods. One method extracts syntactic features and the other three methods extract semantic features. Features extracted using each method were used to build a dataset for each of the five document subsets resulting in a total of 20 data subsets. Each subset consists of 2000 documents. Different preprocessing procedures can capture different aspects of the documents and result in different feature spaces.

For syntactic preprocessing, Wang [60] used the Collins parser [10] to identify non-recursive noun phrases. These non-recursive noun phrases were treated as keywords and each abstract was treated as a "bag of keywords". The cosine coefficient method was used to calculate the similarity between each pair of feature vectors. For constructing semantic feature sets, Wang [60] used the WordNet semantic network[1]. The sense of each word was identified with a sense disambiguation method based on the semantic relatedness between senses. The relatedness of two senses in a semantic network can generally be computed using node-based or edge-based methods [29]. A node-based method computes the relatedness of two senses using the information content of each node in the semantic network and an edge-based method uses the path length between them.

[1] http://wordnet.princeton.edu

The WordNet::Similarity package[2] implements a number of different node-based and edge-based semantic relatedness measures. We used one node-based method (WordNet::Similarity::res), one edge-based method (WordNet::Similarity::lch), and one combined node and edge based method (WordNet::Similarity::jcn) to extract semantic relatedness. Feature vectors were then constructed as a "bag of senses" and the cosine coeff cient method was again used to calculate the similarity between two feature vectors.

5.2 Experimental Design

A similarity matrix was computed from each set of syntactic feature vectors and from each set of semantic feature vectors. Average-link hierarchical clustering was performed using each similarity matrix. As mentioned earlier in Chapter III, our approach is based on clustering the individual datasets and then combining the resulting clusterings.

To test the effectiveness of EPaCH and EPaCHW, we selected several baselines. One of the baselines was the result of hierarchical clustering based on individual feature sets. Another baseline was average-link agglomerative hierarchical clustering based on a concatenation of the feature vectors for the corresponding objects in each dataset. Ten clusters were generated during each clustering because the datasets were known to have ten categories. There are two ways to extract partitional clusters from a dendrogram, cutting the dendrogram at a given height or pruning the dendrograms by selecting clusters at different heights [57]. We used a recursive bottom-up approach to extract partitional clusters

[2] `http://search.cpan.org/dist/WordNet-Similarity`

from the baseline dendrograms. We started by merging the lowest subclusters and stopped when the number of clusters left in the dendrogram was equal to the expected number of partitional clusters.

We also compared the performance of our algorithms against two phylogenetic tree-combination methods, i.e., the consensus tree [7] and supertree [49] methods. For the consensus tree, we used PHYLIP[3] that provides an implementation of the greedy consensus tree algorithm. For the supertree, we used an implementation of the MINCUT algorithm[4]. Again, a bottom-up approach was used to extract partitional clusters from the combined dendrograms.

We also compared our algorithm against partitional clusters generated by the k-means algorithm and a graph-theoretic approach on concatenated feature vectors. For the graph-based clustering, the similarity matrix generated from a dataset was converted into an adjacency matrix for a graph before applying the graph-partitioning [62]. Each object was treated as a vertex and each similarity value between a pair of objects was treated as the weight of the edge between the two corresponding vertices. We used the `kmetis` module from the state-of-the-art METIS[5] software for the graph partitioning.

To test our research hypothesis and the effectiveness of our methods, we also wanted to observe how the methods perform with related datasets that share some but not all objects. This is a case of *bi-regular mapping* that we def ned in Chapter III. To generate

[3] `http://evolution.genetics.washington.edu/phylip.html`

[4] `http://darwin.zoology.gla.ac.uk/cgi-bin/supertree.pl`

[5] `http://www-users.cs.umn.edu/~karypis/metis/metis/index.html`

data for these experiments, the general approach was to select an exclusion percentage and then randomly select that many objects for exclusion from each of the twenty datasets with the constraint that each class remained equal sized. We randomly selected f fty percent of the objects from each dataset for exclusion. On average, this resulted in an overlap of approximately f fty percent between two datasets constructed from the same document subset using different feature sets, i.e., each pair of combined datasets had approximately f fteen hundred objects and each individual dataset had exactly one thousand objects.

When we were constructing a concatenated feature set using two overlapping object sets, we had to deal with missing values for objects that appear only in one of the datasets. There are two general methods for handling missing feature values in clustering: imputation and marginalization [23]. With imputation, missing values are replaced by created values. Some common approaches are to replace the missing values with zeros [1], replace the missing values with the observed mean for that feature [54], and to infer the missing values based on the objects observed features and its similarity to other objects [54]. With marginalization, missing values are ignored and new values are not created. One possible approach is to use only the features that have observed values for two objects when calculating their similarity (known as pairwise deletion) [63]. Another approach is to use the set of missing features as constraints to decide how strongly a pair of objects is related [58]. In our document clustering domain, the feature space is extremely sparse and therefore many of these methods will not be effective. We decided to employ marginalization with pairwise deletion while dealing with the concatenated feature sets.

5.3 Evaluation Methods

In general, a cluster is interesting if it is valid and potentially useful. It is very important to have an effective mechanism for evaluating the results of a clustering algorithm to validate that the clusters have relevance in the context of the domain. While human inspection may sound like the most intuitive evaluation method since it compares the clustering results with the user's intention in a natural way, it lacks scalability and is not always desirable and feasible in real-life applications. Therefore, quantitative assessment of clustering quality is of great importance for various clustering applications. Cluster evaluation, also referred to as cluster validation, is a non-trivial task and can be performed with two broad approaches: internal criteria and external criteria [28].

In an internal criteria based approach, the results of a clustering algorithm are evaluated in terms of quantities that involve the given data, e.g., the distance matrix. In an external criteria based approach, the results of a clustering algorithm are evaluated based on a pre-specif ed structure that ref ects our intuition about the clustering structure of the data set. For example, category labels can be attached to the clusters and then some method can be used to measure the discrepancy between the external categorization and the clustering. Cluster evaluation based on Davies-Bouldin Index and Dunn's Index are examples of external approaches and cluster evaluation based on entropy, purity, F-measure, etc., are examples of internal approaches. Regardless of the approach, the goal of any evaluation mechanism is to assign better scores to a scheme that achieves high intra-cluster similar-

ity low inter-intra-cluster similarity. We used a modif ed *Davies-Bouldin* index, *entropy*, *purity*, and *F-Measure* to compare our results.

5.3.1 Davies-Bouldin Index

The *Davies-Bouldin* index [13] is based on inter-cluster and intra-cluster distances. Intuitively, the Davies-Bouldin index is the average similarity between each cluster and its most similar one. Let $(C_i\ C_j)$ be the distance between two clusters C_i and C_j and (C_i) be the intra-cluster distance, i.e., the measure of dispersion of cluster C_i. Both $(C_i\ C_j)$ and (C_i) can be calculated based on the notion of shortest/longest/average distance between cluster members. The goal is to have a DB index as small as possible since the clusters should be internally compact and well separated from each other. The Davies-Bouldin index is def ned as [13]:

$$\frac{1}{p}\sum_{i=1}^{p}\max_{i=j}\frac{(C_i)+(C_j)}{(C_i\ C_j)}$$

5.3.2 Entropy

Entropy gives a measure of how the various categories of objects are distributed within each cluster. The goal is to minimize entropy and for a set of disjoint clusters, the entropy should be 0. If n_i^j is the number of objects of the jth category C_j that is assigned to C_i,

p is the number of clusters, and q is the number of actual categories[6], then the entropy is def ned as [66]:

$$Entropy(C) = \sum_{i=1}^{p} \frac{C_i}{D} E(C_i)$$

where

$$E(C_i) = \frac{1}{\log q} \sum_{j=1}^{q} \frac{n_i^j}{C_i} \log \frac{n_i^j}{C_i}$$

5.3.3 Purity

Purity gives a measure of how each cluster contains objects from primarily one category. The goal is to maximize purity and for a set of disjoint clusters, the purity should be

1. Using the notation for entropy, purity is def ned as [66]:

$$Purity(C) = \sum_{i=1}^{p} \frac{C_i}{D} P(C_i)$$

where

$$P(C_i) = \frac{1}{C_i} \max_j (n_i^j)$$

5.3.4 F-measure

The F-measure [35] gives a measure of how the clustering f ts the actual classif cation of data, i.e., how most elements in a cluster are from the same category and also how most

[6]For describing the validation indices, we use C_i to represent a computed cluster and C'_j to represent an actual category.

elements from a category are grouped into the same cluster. The goal is to maximize the F-measure and ideally it should be 1.

The F-measure of a cluster C_i with respect to an actual category C_j is calculated as the harmonic average of the *precision* and *recall* of C_i with respect to C_j. If pre_i^j is the precision i.e., the fraction of the objects of C_i that belongs to C_j and rec_i^j is the recall, i.e., the fraction of the objects of C_j that is assigned to C_i, then the F-measure of C_i with respect to C_j is defined as [35]:

$$F_i^j = \frac{2}{(1\ \ pre_i^j) + (1\ \ rec_i^j)} = \frac{2 \quad pre_i^j \quad rec_i^j}{pre_i^j + rec_i^j}$$

$$= \frac{2 \quad (n_i^j \quad C_i\ ) \quad (n_i^j \quad C_j\ )}{(n_i^j \quad C_i\ ) + (n_i^j \quad C_j\ )} = \frac{2 \quad n_i^j}{C_i \ + \ C_j}$$

The F-measure for C_j is taken to be the maximum over all clusters and the weighted sum of the individual F-measures for all the actual categories is calculated to obtain the overall F-measure:

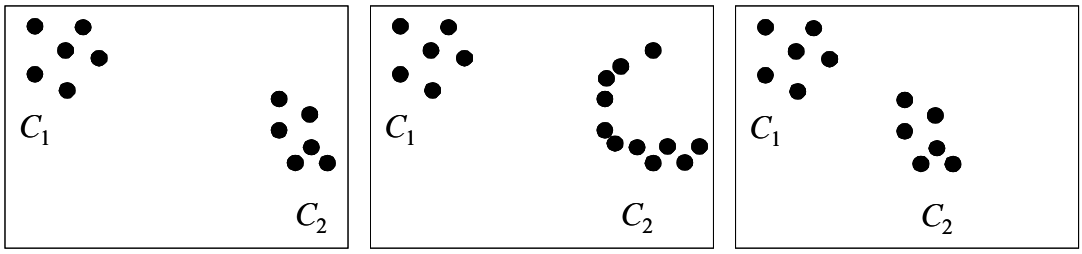
$$\sum_{j=1}^{q} \frac{C_j}{D} \max_{1 \ i \ p} \quad \frac{2 \quad n_i^j}{C_i \ + \ C_j}$$

5.4 A Modified DB-Index

The goal of the DB-index is to produce better (low) values for a clustering with high compactness among the objects of an individual cluster and high separation between clusters themselves. This index is based on a geometric view of the clustering and works well

when the objects are distributed with a uniform density around cluster centers, e.g., when clusters are of spherical shape. However, with real world data, this may not hold since there can be large variability in cluster shapes.

For example, let us consider Figure 5.1 that shows three different instances of clustering. Let us assume that all three clusterings are perfect with respect to the actual categorization of the objects but each is based on a different set of features. If we compare Figure 5.1(b) with Figure 5.1(a), we can see that the DB-index will be higher (worse) for the clustering in Figure 5.1(b) since the C_2 will have a higher intra-cluster distance compared to C_2 in Figure 5.1(a). Note that the DB-index will be the ratio of the sum of the intra-cluster distances of C_1 and C_2 and the inter-cluster distance between C_1 and C_2. Again, if we compare Figure 5.1(c) with Figure 5.1(a), we can see that the inter-cluster distance between C_1 and C_2 will be lower in Figure 5.1(c). Even though both clusterings in Figures 5.1(b) and 5.1(c) are assumed to be perfect, the DB-index values will be higher (worse) compared to the clustering in Figure 5.1(a).



(a) (b) (c)

Figure 5.1 Problems with DB-Index

In our initial experiments with document clustering datasets, the original DB-index did not appear to give informative results. One would expect low DB-index values for a perfect clustering of the data. However, when we evaluated the perfect partitional clusters for the syntactically preprocessed and semantically preprocessed (node-based) subsets with respect to the respective distance matrices, we obtained the DB-index values as shown in Table 5.1. Note that DB-index lies between 0 and 2, with 0 being the best case. So, even though we tested a perfect clustering, we obtained DB-index values close to the worst possible case.

Table 5.1 DB-Index values for perfect clustering

Feature sets	
Syntactic	Semantic
1.95711	1.96472
1.95801	1.96411
1.95468	1.96167
1.95801	1.96459
1.95685	1.96399

To achieve a more informative DB-index, we have developed a modified definition of the DB-index. As mentioned earlier, the calculation of DB-index is based on the intra-cluster and inter-cluster distances. These distances are calculated using a distance metric applied on the actual feature space. We have modified the DB-index to use "expected" intra-cluster and inter-cluster distances instead. This approach offsets the negative effect of variability in the geometric shapes of clusters.

Definition 4 Let C_i be a cluster, n_i be the number of objects in C_i, and $_e(C_i)$ be the *expected* intra-cluster distance of C_i. Then,

$$_e(C_i) = \frac{1}{(n_i)^2} \sum_{a \ C_i} \sum_{b \ C_i} {}_e(a \ b)$$

where

$$_e(a \ b) = \begin{cases} 1 & \text{if } a \text{ and } b \text{ belong to the same category} \\ 0 & \text{otherwise} \end{cases}$$

Definition 5 Let C_i and C_j be two clusters, n_i and n_j be the respective number of objects, and $_e(C_i \ C_j)$ be the *expected* inter-cluster distance between C_i and C_j.Then,

$$_e(C_i \ C_j) = \frac{1}{n_i \ n_j} \sum_{a \ C_i} \sum_{b \ C_j} {}_e(a \ b)$$

Definition 6 Given the definitions of $_e(C_i)$ and $_e(C_i \ C_j)$, we define a modified Davies-Bouldin index as:

$$MDB = \frac{1}{p} \sum_{i=1}^{p} \max_{i=j} \frac{{}_e(C_i) + {}_e(C_j)}{{}_e(C_i \ C_j)}$$

5.4.1 Simulation with MDB

In order to test the effectiveness of the modified DB-index, we generated simulated distance matrices representing progressively "better" clusters. Thirty synthetic matrices were generated where each matrix represented two thousand objects. We assumed that the objects are distributed among ten categories. We performed average-link agglomerative

hierarchical clustering on all these distance matrices and then extracted ten partitional clusters from each.

We computed the original DB-index, MDB, and F-measure for all the resulting clusters. Figure 5.2 shows how the original DB-index and the modif ed DB-index change with the F-measure. Note that the F-measure is known to yield appropriate values for different clusters, i.e., high values for the higher quality clusters and low values for the lower quality clusters. It can be seen that the MDB yields progressively lower (better) values with increasing F-measure values. For a perfect clustering, the F-measure yields a value of 1 and the MDB yields a value of 0. Moreover, the MDB changes almost linearly (with a negative slope) with F-measure. On the other hand, the original DB-index does not yield low values for high quality clusters, but tends to have values limited to a narrower range.

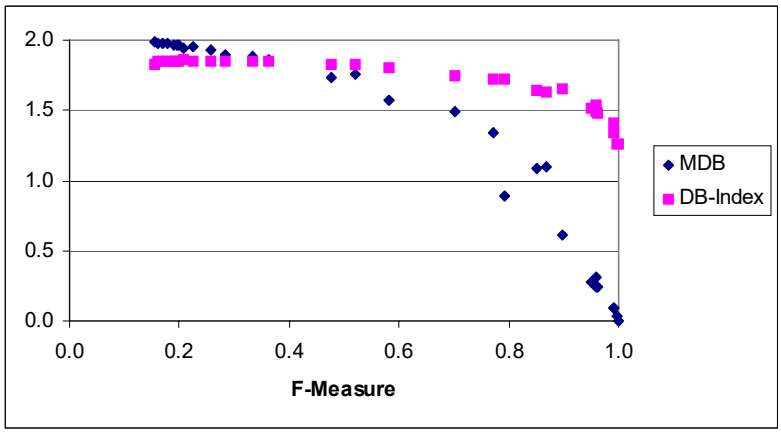


Figure 5.2 Changes in original DB-Index and modif ed DB-Index against changes in F-measure for synthetic distance matrices

5.5 Results of Algorithm Evaluation

5.5.1 Results with Datasets having Identical Sets of Objects

The MDB, *entropy*, *purity*, and *F-Measure* were used to evaluate the quality of different clustering algorithms. Ensemble clustering results using EPaCH and EPaCHW were compared to baselines described in section 5.2. Table 5.2 - Table 5.5 present the evaluation results for the four different validation indices. Note that the results are for ten partitional clusters generated using each clustering scheme. Each table presents the values for a particular validation index for different clustering schemes. Each subset of f ve rows represents a different combination of feature sets where each row represents the particular index for one of the f ve document subsets. The clustering schemes used in the experiment and the notation used in the tables to represent them are:

Hierarchical clustering based on a single feature set ($\mathcal{C}_1$ $\mathcal{C}_2$)

Hierarchical clustering with a concatenated feature set ($\mathcal{C}_{con}$)

Clusters resulting from the combination of individual hierarchical clusterings using the consensus tree method ($\mathcal{C}_{cns}$)

Clusters resulting from the combination of individual hierarchical clusterings using the supertree method ($\mathcal{C}_{sup}$)

k-means clustering with a concatenated feature set ($\mathcal{C}_{km}$)

Graph-based clustering with a concatenated feature set ($\mathcal{C}_{gr}$)

Clusters resulting from the combination of individual hierarchical clusterings using EPaCH ($\mathcal{C}_{ep}$)

Clusters resulting from the combination of individual hierarchical clusterings using EPaCHW ($\mathcal{C}_{epw}$)

Table 5.2 Comparison of modif ed DB-Index for EPaCH algorithms

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cns}$	$\mathcal{C}_{sup}$	$\mathcal{C}_{km}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
Syntactic and node-based semantic feature sets									
Dataset#1	1.27	1.64	1.46	1.61	1.46	0.75	0.70	0.83	0.79
Dataset#2	1.57	1.49	1.45	1.62	1.70	1.40	0.88	1.35	1.29
Dataset#3	1.60	1.26	1.32	1.67	1.28	1.27	0.64	0.79	0.69
Dataset#4	1.38	1.50	1.46	1.59	1.45	1.09	0.80	0.80	0.73
Dataset#5	1.25	1.58	1.34	1.78	1.62	0.93	0.78	1.23	0.76
Syntactic and edge-based semantic feature sets									
Dataset#1	1.27	1.42	1.40	1.47	1.27	1.09	0.78	0.95	0.88
Dataset#2	1.57	1.53	1.44	1.67	1.67	1.45	0.86	1.30	1.28
Dataset#3	1.60	1.38	1.56	1.36	1.44	1.37	0.61	0.94	0.79
Dataset#4	1.38	1.59	1.36	1.65	1.72	1.47	0.82	1.30	1.10
Dataset#5	1.25	1.56	1.28	1.36	1.42	1.14	0.78	1.04	0.99
Syntactic and node-and-edge-based semantic feature sets									
Dataset#1	1.27	1.32	1.47	1.45	1.53	1.49	0.77	0.70	0.72
Dataset#2	1.57	1.63	1.60	1.48	1.58	1.41	1.00	1.32	0.92
Dataset#3	1.60	1.45	1.38	1.47	1.49	1.43	0.60	0.97	0.80
Dataset#4	1.38	1.55	1.32	1.40	1.61	1.50	0.77	1.38	0.91
Dataset#5	1.25	1.50	1.33	1.29	1.51	1.21	0.62	0.85	0.80
Node-based and edge-based semantic feature sets									
Dataset#1	1.64	1.42	1.49	1.68	1.65	1.26	0.77	1.03	0.78
Dataset#2	1.49	1.53	1.33	1.49	1.67	1.38	0.92	0.98	0.84
Dataset#3	1.26	1.38	1.27	1.29	1.36	1.28	0.58	0.72	0.72
Dataset#4	1.50	1.59	1.50	1.30	1.64	1.44	0.71	1.32	0.81
Dataset#5	1.58	1.56	1.48	1.28	1.70	1.48	0.72	1.03	0.95
Node-based and node-and-edge-based semantic feature sets									
Dataset#1	1.64	1.32	1.47	1.72	1.36	1.31	0.71	0.76	0.77
Dataset#2	1.49	1.63	1.58	1.60	1.68	1.22	0.90	0.99	0.82
Dataset#3	1.26	1.45	1.24	1.32	1.60	1.12	0.55	1.24	0.73
Dataset#4	1.50	1.55	1.44	1.62	1.33	1.43	0.72	0.96	1.02
Dataset#5	1.58	1.50	1.37	1.57	1.58	1.14	1.36	0.77	0.80
Edge-based and node-and-edge-based semantic feature sets									
Dataset#1	1.42	1.32	1.40	1.48	1.71	1.24	0.91	0.89	0.81
Dataset#2	1.53	1.63	1.51	1.45	1.66	1.32	0.96	1.15	0.80
Dataset#3	1.38	1.45	1.57	1.52	1.32	1.31	0.53	1.38	0.80
Dataset#4	1.59	1.55	1.38	1.35	1.55	1.65	0.74	1.34	0.95
Dataset#5	1.56	1.50	1.55	1.48	1.51	1.39	0.76	0.86	0.76

Table 5.3 Comparison of entropy for EPaCH algorithms

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cns}$	$\mathcal{C}_{sup}$	$\mathcal{C}_{km}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
Syntactic and node-based semantic feature sets									
Dataset#1	0.41	0.54	0.36	0.50	0.54	0.19	0.23	0.25	0.21
Dataset#2	0.57	0.56	0.59	0.55	0.53	0.35	0.28	0.38	0.34
Dataset#3	0.57	0.36	0.37	0.46	0.43	0.33	0.22	0.24	0.22
Dataset#4	0.35	0.48	0.51	0.47	0.49	0.27	0.27	0.26	0.24
Dataset#5	0.39	0.42	0.41	0.55	0.43	0.24	0.24	0.32	0.24
Syntactic and edge-based semantic feature sets									
Dataset#1	0.41	0.39	0.41	0.45	0.54	0.27	0.25	0.25	0.23
Dataset#2	0.57	0.60	0.39	0.46	0.58	0.34	0.28	0.37	0.34
Dataset#3	0.57	0.38	0.39	0.40	0.52	0.33	0.20	0.26	0.23
Dataset#4	0.35	0.54	0.44	0.47	0.61	0.37	0.29	0.34	0.30
Dataset#5	0.39	0.55	0.37	0.39	0.42	0.29	0.24	0.32	0.30
Syntactic and node-and-edge-based semantic feature sets									
Dataset#1	0.41	0.47	0.41	0.34	0.47	0.40	0.25	0.21	0.20
Dataset#2	0.57	0.69	0.46	0.45	0.66	0.33	0.30	0.36	0.28
Dataset#3	0.57	0.36	0.44	0.39	0.43	0.37	0.21	0.28	0.23
Dataset#4	0.35	0.56	0.35	0.40	0.61	0.32	0.25	0.38	0.27
Dataset#5	0.39	0.42	0.39	0.39	0.46	0.28	0.21	0.28	0.25
Node-based and edge-based semantic feature sets									
Dataset#1	0.54	0.39	0.54	0.46	0.56	0.33	0.25	0.26	0.22
Dataset#2	0.56	0.60	0.44	0.46	0.55	0.33	0.29	0.30	0.26
Dataset#3	0.36	0.38	0.34	0.35	0.46	0.31	0.19	0.23	0.22
Dataset#4	0.48	0.54	0.54	0.39	0.53	0.40	0.25	0.33	0.25
Dataset#5	0.42	0.55	0.61	0.35	0.64	0.36	0.24	0.35	0.30
Node-based and node-and-edge-based semantic feature sets									
Dataset#1	0.54	0.47	0.48	0.49	0.49	0.28	0.24	0.22	0.21
Dataset#2	0.56	0.69	0.44	0.43	0.59	0.34	0.30	0.32	0.25
Dataset#3	0.36	0.36	0.36	0.33	0.47	0.24	0.21	0.29	0.22
Dataset#4	0.48	0.56	0.59	0.51	0.47	0.37	0.27	0.27	0.29
Dataset#5	0.42	0.42	0.50	0.38	0.60	0.27	0.34	0.24	0.25
Edge-based and node-and-edge-based semantic feature sets									
Dataset#1	0.39	0.47	0.51	0.42	0.58	0.30	0.29	0.25	0.20
Dataset#2	0.60	0.69	0.41	0.37	0.76	0.31	0.29	0.33	0.26
Dataset#3	0.38	0.36	0.43	0.48	0.48	0.29	0.20	0.37	0.25
Dataset#4	0.54	0.56	0.51	0.41	0.42	0.52	0.26	0.37	0.26
Dataset#5	0.55	0.42	0.40	0.36	0.36	0.30	0.24	0.27	0.25

Table 5.4 Comparison of purity for EPaCH algorithms

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cns}$	$\mathcal{C}_{sup}$	$\mathcal{C}_{km}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
Syntactic and node-based semantic feature sets									
Dataset#1	0.57	0.43	0.58	0.59	0.38	0.87	0.86	0.86	0.88
Dataset#2	0.45	0.43	0.39	0.50	0.46	0.72	0.84	0.70	0.75
Dataset#3	0.44	0.63	0.63	0.60	0.55	0.69	0.89	0.86	0.88
Dataset#4	0.64	0.51	0.43	0.60	0.53	0.80	0.84	0.84	0.86
Dataset#5	0.57	0.55	0.57	0.53	0.48	0.84	0.86	0.78	0.86
Syntactic and edge-based semantic feature sets									
Dataset#1	0.57	0.57	0.57	0.61	0.39	0.77	0.85	0.83	0.84
Dataset#2	0.45	0.41	0.61	0.63	0.45	0.66	0.84	0.73	0.76
Dataset#3	0.44	0.63	0.56	0.68	0.45	0.71	0.89	0.84	0.87
Dataset#4	0.64	0.40	0.49	0.61	0.30	0.64	0.84	0.75	0.81
Dataset#5	0.57	0.43	0.65	0.66	0.55	0.76	0.86	0.81	0.81
Syntactic and node-and-edge-based semantic feature sets									
Dataset#1	0.57	0.49	0.57	0.73	0.48	0.62	0.85	0.88	0.88
Dataset#2	0.45	0.32	0.52	0.59	0.37	0.72	0.81	0.75	0.84
Dataset#3	0.44	0.64	0.55	0.69	0.54	0.64	0.89	0.83	0.87
Dataset#4	0.64	0.40	0.63	0.70	0.29	0.73	0.85	0.72	0.84
Dataset#5	0.57	0.57	0.59	0.71	0.48	0.78	0.88	0.84	0.86
Node-based and edge-based semantic feature sets									
Dataset#1	0.43	0.57	0.39	0.56	0.39	0.70	0.85	0.81	0.87
Dataset#2	0.43	0.41	0.54	0.64	0.44	0.73	0.83	0.81	0.85
Dataset#3	0.63	0.63	0.64	0.72	0.49	0.70	0.90	0.87	0.88
Dataset#4	0.51	0.40	0.45	0.72	0.47	0.63	0.86	0.75	0.85
Dataset#5	0.55	0.43	0.38	0.74	0.34	0.68	0.86	0.79	0.82
Node-based and node-and-edge-based semantic feature sets									
Dataset#1	0.43	0.49	0.49	0.48	0.49	0.74	0.86	0.88	0.88
Dataset#2	0.43	0.32	0.52	0.64	0.37	0.73	0.82	0.80	0.86
Dataset#3	0.63	0.64	0.65	0.71	0.47	0.82	0.89	0.78	0.87
Dataset#4	0.51	0.40	0.38	0.53	0.54	0.67	0.85	0.83	0.81
Dataset#5	0.55	0.57	0.47	0.62	0.38	0.80	0.74	0.86	0.86
Edge-based and node-and-edge-based semantic feature sets									
Dataset#1	0.57	0.49	0.47	0.67	0.39	0.77	0.82	0.84	0.88
Dataset#2	0.41	0.32	0.54	0.67	0.26	0.73	0.81	0.79	0.86
Dataset#3	0.63	0.64	0.56	0.56	0.49	0.75	0.90	0.73	0.85
Dataset#4	0.40	0.40	0.50	0.69	0.57	0.53	0.85	0.72	0.84
Dataset#5	0.43	0.57	0.56	0.70	0.58	0.72	0.86	0.84	0.86

Table 5.5 Comparison of F-measure for EPaCH algorithms

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cns}$	$\mathcal{C}_{sup}$	$\mathcal{C}_{km}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
Syntactic and node-based semantic feature sets									
Dataset#1	0.65	0.51	0.68	0.54	0.51	0.86	0.86	0.86	0.88
Dataset#2	0.51	0.50	0.48	0.50	0.49	0.71	0.84	0.72	0.75
Dataset#3	0.51	0.71	0.69	0.56	0.64	0.72	0.89	0.86	0.88
Dataset#4	0.71	0.57	0.54	0.55	0.58	0.77	0.84	0.84	0.86
Dataset#5	0.66	0.63	0.64	0.50	0.60	0.84	0.86	0.78	0.86
Syntactic and edge-based semantic feature sets									
Dataset#1	0.65	0.66	0.65	0.58	0.52	0.77	0.85	0.83	0.84
Dataset#2	0.51	0.47	0.66	0.58	0.49	0.69	0.84	0.73	0.76
Dataset#3	0.51	0.69	0.67	0.64	0.55	0.73	0.89	0.84	0.87
Dataset#4	0.71	0.51	0.60	0.57	0.43	0.67	0.83	0.76	0.81
Dataset#5	0.66	0.54	0.71	0.67	0.65	0.75	0.86	0.81	0.81
Syntactic and node-and-edge-based semantic feature sets									
Dataset#1	0.65	0.57	0.65	0.69	0.58	0.63	0.85	0.88	0.88
Dataset#2	0.51	0.40	0.61	0.59	0.42	0.70	0.81	0.75	0.84
Dataset#3	0.51	0.70	0.64	0.67	0.64	0.65	0.89	0.83	0.87
Dataset#4	0.71	0.51	0.71	0.67	0.43	0.74	0.85	0.74	0.84
Dataset#5	0.66	0.64	0.65	0.71	0.59	0.79	0.88	0.84	0.86
Node-based and edge-based semantic feature sets									
Dataset#1	0.51	0.66	0.51	0.57	0.49	0.69	0.85	0.81	0.87
Dataset#2	0.50	0.47	0.60	0.62	0.49	0.72	0.82	0.82	0.85
Dataset#3	0.71	0.69	0.72	0.68	0.53	0.71	0.90	0.87	0.88
Dataset#4	0.57	0.51	0.52	0.70	0.52	0.60	0.86	0.76	0.85
Dataset#5	0.63	0.54	0.47	0.72	0.42	0.64	0.86	0.79	0.82
Node-based and node-and-edge-based semantic feature sets									
Dataset#1	0.51	0.57	0.57	0.50	0.56	0.74	0.86	0.88	0.88
Dataset#2	0.50	0.40	0.61	0.61	0.44	0.73	0.82	0.80	0.86
Dataset#3	0.71	0.70	0.71	0.71	0.58	0.83	0.89	0.79	0.87
Dataset#4	0.57	0.51	0.46	0.49	0.60	0.68	0.85	0.83	0.81
Dataset#5	0.63	0.64	0.56	0.62	0.46	0.77	0.75	0.86	0.86
Edge-based and node-and-edge-based semantic feature sets									
Dataset#1	0.66	0.57	0.56	0.62	0.48	0.79	0.82	0.84	0.88
Dataset#2	0.47	0.40	0.64	0.67	0.35	0.74	0.81	0.79	0.86
Dataset#3	0.69	0.70	0.61	0.54	0.58	0.77	0.90	0.75	0.85
Dataset#4	0.51	0.51	0.55	0.67	0.64	0.54	0.85	0.73	0.84
Dataset#5	0.54	0.64	0.65	0.67	0.65	0.76	0.86	0.84	0.86

The information presented in Table 5.2 - Table 5.5 is summarized graphically in Figure 5.3 and results of paired T-tests for selected hypotheses are given in Table 5.6. If we compare the performance (over all indices) of the clustering performed on the concatenated feature sets with clustering based on individual feature sets, we can see that the concatenated feature set gives clusters that are typically intermediate in quality when compared to the results based on individual feature sets. Only in a few instances does the clustering on concatenated feature set outperform both clusterings based on individual feature sets. This demonstrates that feature set concatenation does not necessarily take advantage of the mutual information in the individual feature sets to give improved clustering. On the other hand, if we compare EPaCH with both the clustering based on individual feature sets and clustering based on concatenated feature sets, EPaCH consistently yields higher quality clusters for all validation indices for all datasets. Also note that EPaCHW outperforms EPaCH for all indices. This indicates that taking the intra-cluster similarity into consideration provides critical information for improving the quality of the clusters.

EPaCH and EPaCHW also outperform the clustering based on phylogenetic tree combination methods (consensus tree and supertree). The phylogenetic methods do not even consistently outperform clustering based on individual features sets or the clustering based on concatenated feature sets. In Chapter IV, we explained the drawbacks of the consensus tree and supertree methods when the goal is to extract partitional clusters. We forced ten clusters from each output dendrogram generated by the consensus tree and the supertree method by putting all the small clusters into one big one. Also, the supertree algorithm is

not memory efficient and becomes very time consuming when dealing with large datasets. It recursively splits a graph (that contains all the objects) several times and keeps all the sub-graphs in memory.

Finally, the quality of clusters generated by EPaCH and EPaCHW were compared with the clusters generated by two partitional algorithms (k-means and graph-based) on the concatenated feature sets. Since the performance of k-means depends on the random selection of the initial cluster centers, we ran k-means five times and the results presented are averages over five runs. Out of all thirty instances, only in four cases, EPaCH performs worse than k-means for all validation indices. There are also six more instances when k-means yields better entropy and two more instances when k-means yields better purity. But, in general, EPaCH and EPaCHW yield higher quality clusters compared to k-means with the concatenated feature sets.

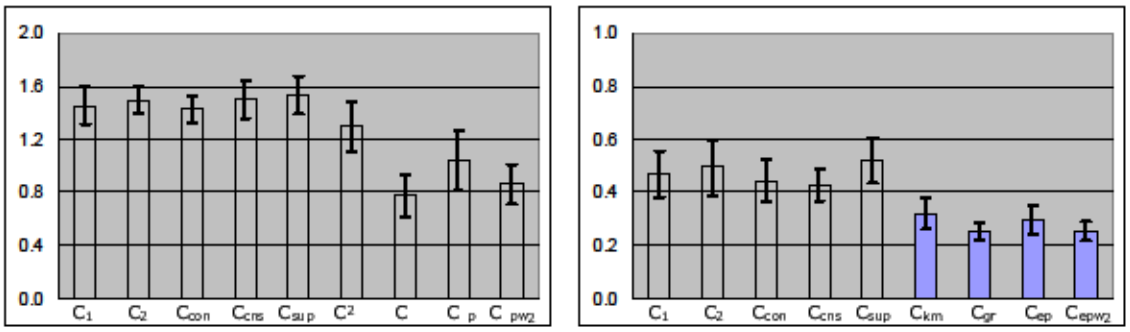
On the flip side, EPaCH does not compare well with the graph-based algorithm. Only in four cases does EPaCH performs better than the graph-based clustering performed on concatenated feature sets for all validation indices. EPaCHW is more competitive with the graph-based clustering on concatenated feature sets although the latter still seems better than EPaCHW. This is discussed further in terms of statistical tests of significance below.

Figure 5.3 shows a graphical representation of the comparison of different validation indices for the different clustering schemes. The results shown are averages over all observations (thirty subsets) along with the standard deviations. For the individual clusterings, $\mathcal{C}_1$ represents the better of the two clusterings and $\mathcal{C}_2$ represents the worse of the

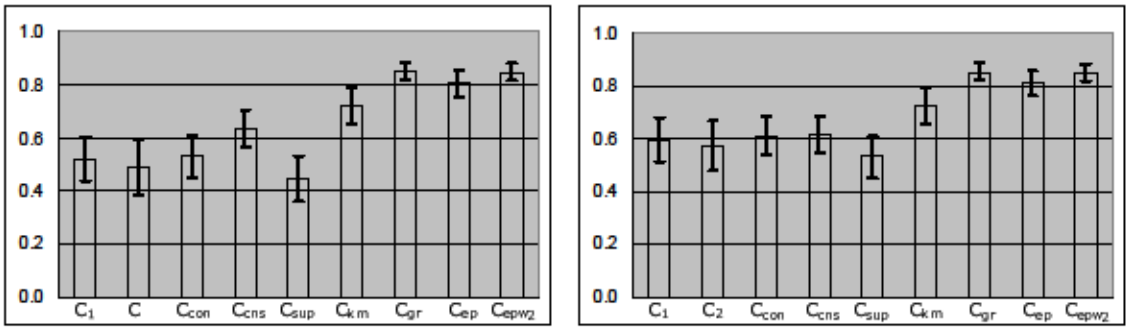
two clusterings. The graphs clearly indicate that EPaCH and EPaCHW yield higher quality clusters than the other approaches with the exception of the graph-based clustering on concatenated feature sets. In addition to the average values, it can be seen that the standard deviations are lower for EPaCH and EPaCHW than all the baselines except the graph-based clustering on concatenated feature sets.

Table 5.6 show the results of paired *T*-tests for selected hypotheses. In the notation used, $\mathcal{C}_x \quad \mathcal{C}_y$ means the index values for $\mathcal{C}_x$ are signif cantly better than those for $\mathcal{C}_y$. The clustering subscripts have been previously introduced. $\mathcal{C}_b$ represents the better of the two individual clusterings and $\mathcal{C}_w$ represents the worse of the two individual clusterings. Each row shows the T-test results for a particular validation index. The tests were performed with $= 05$ and the corresponding $T_{critical} = 1\ 699$ for one-tail tests. For each hypothesis, there are two columns. The f rst column shows the value of the *T*-statistic and the second column shows the *p*-values, i.e., the probability of error involved in accepting the hypothesis. A *T*-statistic of 1.699 or higher and a *p*-value of 0.05 or lower provides evidence that the hypothesis is true.

Note that these results again show that hierarchical clustering with the concatenated feature set is signif cantly more effective than the worst individual clustering and signif cantly less effective than the best individual clustering. The *T*-test results verify our claim that both EPaCH and EPaCHW signif cantly outperform clustering based on the individual feature sets. EPaCH and EPaCHW are consistently signif cantly better than the best clustering based on individual feature sets and is also signif cantly better than clustering



(a) MDB (b) Entropy



(c) Purity (d) F-Measure

Figure 5.3 Comparison of averages of different validation indices for EPaCH algorithms and baseline clustering schemes

Table 5.6 The one-tailed T-test results for comparing EPaCH algorithms with other clustering schemes where $= 05$ and $T_{critical} = 1\ 699$

	T_{stat}	$P_{T\ t}$	T_{stat}	$P_{T\ t}$	T_{stat}	$P_{T\ t}$	T_{stat}	$P_{T\ t}$
	$\mathcal{C}_{con}$	$\mathcal{C}_b$	$\mathcal{C}_{con}$	$\mathcal{C}_w$	$\mathcal{C}_{ep}$	$\mathcal{C}_b$	$\mathcal{C}_{ep}$	$\mathcal{C}_{con}$
MDB	-1.32	0.097	6.38	2.85e-07	9.42	1.26e-10	8.87	4.67e-10
Entropy	-0.55	0.294	4.22	0.0001	9.97	3.51e-11	9.56	9.18e-11
Purity	-1.23	0.114	3.92	0.0003	17.56	2.71e-17	16.34	1.81e-16
F-Meas	-0.87	0.196	3.98	0.0002	13.59	2.09e-14	13.45	2.68e-14
	$\mathcal{C}_{cns}$	$\mathcal{C}_{con}$	$\mathcal{C}_{sup}$	$\mathcal{C}_{con}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{cns}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{sup}$
MDB	-2.64	0.007	-3.69	0.0005	9.37	1.42e-10	12.48	1.72e-13
Entropy	1.14	0.132	-4.08	0.0002	9.81	5.07e-11	14.70	2.85e-15
Purity	6.65	1.35e-07	-4.18	0.0001	10.77	5.93e-12	22.21	4.71e-20
F-Meas	0.21	0.418	-4.43	6.17e-05	12.75	1.02e-13	18.67	5.23e-18
	$\mathcal{C}_{ep}$	$\mathcal{C}_{km}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{epw}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$	$\mathcal{C}_{gr}$
MDB	5.95	9.16e-07	-5.21	7.07e-06	5.21	7.01e-06	-2.40	0.011
Entropy	2.07	0.024	-4.34	7.92e-05	6.99	5.37e-08	-0.06	0.476
Purity	5.98	8.41e-07	-4.33	8.19e-05	6.09	6.23e-07	-0.48	0.317
F-Meas	6.42	2.53e-07	-4.34	7.87e-05	6.30	3.52e-07	-0.47	0.322

based on the concatenated feature sets using hierarchical clustering or *k*-means clustering. EPaCHW is always signif cantly better than EPaCH for all indices. It should also be noted that the statistical tests produce very low p-values when EPaCH and EPaCHW are compared against other baselines except the graph based clustering on concatenated feature sets. For our document datasets, EPaCH does not perform as well as the graph based clustering on concatenated feature sets. EPaCHW performs slightly worse than the graph based clustering on concatenated feature sets as indicated by the negative T-statistic, but the differences are not signif cantly different.

5.5.2 Results with Datasets having Overlapping Sets of Objects

As described in Section 5.2, we generated overlapping datasets to evaluate the performance of the EPaCH algorithms when the two datasets do not represent exactly the same data objects but share some objects. These datasets were constructed by randomly removing 50% of the objects from each dataset. We have excluded the consensus tree, supertree, and k-means from these experiments. The consensus tree method cannot be applied to two dendrograms consisting of overlapping objects sets. The supertree method performed worse than most other methods even in the case of identical objects sets and so was not considered further. The performance of EPaCH and EPaCHW were compared with that of the graph-based clustering for the concatenated feature sets since that in general performs better than k-means.

Table 5.7 - Table 5.10 present the results for the different validation indices for 30 datasets. Each table presents a particular index for different clustering schemes. Each subset of five rows represents a different combination of feature sets and each row represents the particular index for one of the five document subsets. The clustering schemes are: hierarchical clustering based on individual feature sets ($\mathcal{C}_1$ $\mathcal{C}_2$), hierarchical clustering with concatenated feature sets ($\mathcal{C}_{con}$), graph-based clustering with concatenated feature sets ($\mathcal{C}_{gr}$), combination of individual hierarchical clusterings using EPaCH ($\mathcal{C}_{ep}$), and combination of individual hierarchical clusterings using EPaCHW ($\mathcal{C}_{epw}$). Figure 5.4 gives a graphical summary of these results and Table 5.11 shows the results of paired T-tests done on selected hypotheses with overlapping object sets.

As observed with the clustering done on identical object sets, in most instances, hierarchical clustering with the concatenated feature sets performs worse than one of the two individual clusterings. EPaCH yields better values for all of the validation indices compared to the better of clusterings based on individual feature sets with the exception of entropy. It is known that the entropy measure yields lower (better) values for a clustering where each cluster has fewer objects with respect to a set of reference clusters. Note that the reference clusters for the evaluations described here consist of objects appearing in both the datasets under consideration. When these are used to measure the entropy, the individual clustering, which has fewer objects than a clustering based on the combined dataset, will result in lower (better) entropy values.

Table 5.7 Comparison of modified DB-index for EPaCH algorithms - results are for datasets with 50% of the objects randomly removed

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
	Syntactic and node-based semantic feature sets						Node-based and edge-based semantic feature sets					
#1	1.49	1.37	1.62	1.67	1.22	1.11	1.37	1.44	1.47	1.79	1.11	1.05
#2	1.56	1.64	1.79	1.70	1.30	1.23	1.64	1.64	1.62	1.70	1.48	1.16
#3	1.51	1.45	1.49	1.76	0.87	0.83	1.45	1.29	1.29	1.71	1.47	1.21
#4	1.31	1.61	1.57	1.72	1.28	1.13	1.61	1.59	1.72	1.76	1.50	1.29
#5	1.39	1.55	1.59	1.85	1.45	1.24	1.55	1.55	1.34	1.50	1.43	1.32
	Syntactic and edge-based semantic feature sets						Node-based and node-and-edge-based semantic feature sets					
#1	1.49	1.44	1.65	1.72	1.45	0.74	1.37	1.39	1.47	1.77	1.34	0.95
#2	1.56	1.64	1.57	1.70	1.47	1.35	1.64	1.44	1.71	1.61	1.54	1.10
#3	1.51	1.29	1.62	1.71	1.48	1.08	1.45	1.55	1.69	1.66	1.23	1.03
#4	1.31	1.59	1.60	1.62	1.39	1.31	1.61	1.52	1.73	1.63	1.41	1.15
#5	1.39	1.55	1.52	1.67	1.26	1.18	1.55	1.46	1.49	1.60	1.45	1.31
	Syntactic and node-and-edge-based semantic feature sets						Edge-based and node-and-edge-based semantic feature sets					
#1	1.49	1.39	1.56	1.71	1.31	1.01	1.44	1.39	1.51	1.66	1.33	0.82
#2	1.56	1.44	1.72	1.55	1.47	1.27	1.64	1.44	1.62	1.76	1.65	1.36
#3	1.51	1.55	1.48	1.58	1.35	1.03	1.29	1.55	1.45	1.66	1.30	1.32
#4	1.31	1.52	1.70	1.73	1.34	1.12	1.59	1.52	1.51	1.72	1.43	1.30
#5	1.39	1.46	1.74	1.60	1.32	1.10	1.55	1.46	1.67	1.67	1.45	1.38

Each row represents the modified DB-index for one of the five document subsets where randomly selected 50% objects were excluded from each subset.

Table 5.8 Comparison of entropy for EPaCH algorithms - results are for datasets with 50% of the objects randomly removed

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
	Syntactic and node-based semantic feature sets						Node-based and edge-based semantic feature sets					
#1	0.23	0.28	0.49	0.48	0.33	0.29	0.28	0.26	0.59	0.54	0.29	0.27
#2	0.29	0.36	0.66	0.53	0.41	0.36	0.36	0.36	0.63	0.50	0.45	0.35
#3	0.22	0.27	0.48	0.48	0.28	0.28	0.27	0.37	0.52	0.51	0.46	0.32
#4	0.30	0.37	0.52	0.51	0.41	0.34	0.37	0.42	0.56	0.52	0.45	0.37
#5	0.23	0.42	0.58	0.50	0.43	0.36	0.42	0.35	0.63	0.45	0.39	0.39
	Syntactic and edge-based semantic feature sets						Node-based and node-and-edge-based semantic feature sets					
#1	0.23	0.26	0.59	0.48	0.36	0.24	0.28	0.27	0.68	0.55	0.36	0.32
#2	0.30	0.36	0.61	0.53	0.45	0.40	0.36	0.45	0.63	0.49	0.49	0.32
#3	0.23	0.37	0.61	0.52	0.41	0.31	0.27	0.35	0.57	0.52	0.35	0.33
#4	0.30	0.42	0.55	0.46	0.44	0.40	0.37	0.27	0.65	0.52	0.46	0.31
#5	0.23	0.36	0.49	0.53	0.36	0.33	0.42	0.31	0.55	0.47	0.44	0.38
	Syntactic and node-and-edge-based semantic feature sets						Edge-based and node-and-edge-based semantic feature sets					
#1	0.23	0.28	0.52	0.50	0.36	0.28	0.26	0.27	0.63	0.53	0.35	0.26
#2	0.29	0.45	0.61	0.47	0.45	0.37	0.36	0.46	0.60	0.55	0.48	0.39
#3	0.23	0.35	0.49	0.46	0.38	0.27	0.38	0.35	0.44	0.48	0.36	0.33
#4	0.30	0.27	0.73	0.52	0.41	0.35	0.43	0.28	0.73	0.54	0.45	0.37
#5	0.23	0.31	0.49	0.50	0.37	0.34	0.35	0.31	0.70	0.53	0.46	0.41

Each row represents the entropy values for one of the f ve document subsets where randomly selected 50% objects were excluded from each subset.

Table 5.9 Comparison of purity for EPaCH algorithms - results are for datasets with 50% of the objects randomly removed

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
	Syntactic and node-based semantic feature sets						Node-based and edge-based semantic feature sets					
#1	0.45	0.38	0.50	0.55	0.75	0.80	0.38	0.38	0.42	0.50	0.79	0.81
#2	0.34	0.29	0.39	0.53	0.70	0.74	0.29	0.29	0.38	0.56	0.65	0.75
#3	0.42	0.41	0.54	0.54	0.84	0.85	0.41	0.28	0.48	0.50	0.65	0.77
#4	0.38	0.33	0.42	0.54	0.74	0.78	0.32	0.24	0.45	0.50	0.66	0.74
#5	0.44	0.25	0.45	0.49	0.65	0.75	0.25	0.31	0.34	0.62	0.67	0.72
	Syntactic and edge-based semantic feature sets						Node-based and node-and-edge-based semantic feature sets					
#1	0.44	0.38	0.47	0.56	0.69	0.86	0.38	0.37	0.35	0.51	0.70	0.80
#2	0.34	0.29	0.42	0.50	0.64	0.71	0.29	0.20	0.38	0.56	0.61	0.78
#3	0.42	0.29	0.44	0.49	0.69	0.81	0.41	0.29	0.45	0.53	0.75	0.80
#4	0.38	0.25	0.46	0.58	0.68	0.71	0.32	0.38	0.39	0.51	0.63	0.78
#5	0.44	0.31	0.51	0.54	0.73	0.78	0.25	0.36	0.52	0.59	0.67	0.72
	Syntactic and node-and-edge-based semantic feature sets						Edge-based and node-and-edge-based semantic feature sets					
#1	0.45	0.38	0.50	0.53	0.72	0.81	0.38	0.37	0.41	0.51	0.72	0.85
#2	0.34	0.20	0.42	0.55	0.66	0.74	0.29	0.20	0.41	0.50	0.56	0.71
#3	0.42	0.28	0.50	0.56	0.69	0.82	0.29	0.29	0.57	0.56	0.72	0.72
#4	0.37	0.38	0.30	0.49	0.71	0.78	0.25	0.39	0.33	0.48	0.66	0.73
#5	0.43	0.36	0.51	0.57	0.75	0.79	0.31	0.36	0.34	0.52	0.65	0.72

Each row represents the purity values for one of the f ve document subsets where randomly selected 50% objects were excluded from each subset.

Table 5.10 Comparison of F-measure for EPaCH algorithms - results are for datasets with 50% of the objects randomly removed

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{epw}$
	Syntactic and node-based semantic feature sets						Node-based and edge-based semantic feature sets					
#1	0.60	0.53	0.52	0.55	0.76	0.80	0.53	0.55	0.47	0.48	0.79	0.81
#2	0.50	0.44	0.42	0.53	0.70	0.74	0.44	0.44	0.44	0.56	0.65	0.75
#3	0.59	0.54	0.56	0.53	0.84	0.85	0.54	0.44	0.53	0.51	0.63	0.77
#4	0.52	0.45	0.52	0.54	0.74	0.78	0.44	0.38	0.47	0.50	0.66	0.74
#5	0.58	0.38	0.47	0.49	0.65	0.75	0.38	0.47	0.43	0.63	0.67	0.71
	Syntactic and edge-based semantic feature sets						Node-based and node-and-edge-based semantic feature sets					
#1	0.60	0.55	0.48	0.55	0.69	0.86	0.53	0.53	0.41	0.49	0.70	0.80
#2	0.50	0.45	0.48	0.49	0.65	0.71	0.44	0.31	0.41	0.56	0.61	0.78
#3	0.59	0.44	0.46	0.49	0.68	0.81	0.55	0.44	0.48	0.52	0.75	0.80
#4	0.52	0.39	0.49	0.57	0.68	0.72	0.44	0.53	0.39	0.51	0.63	0.78
#5	0.58	0.47	0.51	0.54	0.73	0.78	0.38	0.50	0.54	0.59	0.66	0.72
	Syntactic and node-and-edge-based semantic feature sets						Edge-based and node-and-edge-based semantic feature sets					
#1	0.60	0.54	0.54	0.51	0.73	0.82	0.55	0.53	0.45	0.51	0.72	0.85
#2	0.50	0.31	0.47	0.55	0.66	0.75	0.45	0.32	0.44	0.49	0.56	0.71
#3	0.59	0.44	0.54	0.55	0.67	0.82	0.44	0.44	0.59	0.57	0.73	0.72
#4	0.51	0.53	0.37	0.48	0.70	0.78	0.38	0.54	0.40	0.47	0.66	0.73
#5	0.58	0.50	0.52	0.58	0.75	0.79	0.47	0.49	0.37	0.51	0.64	0.71

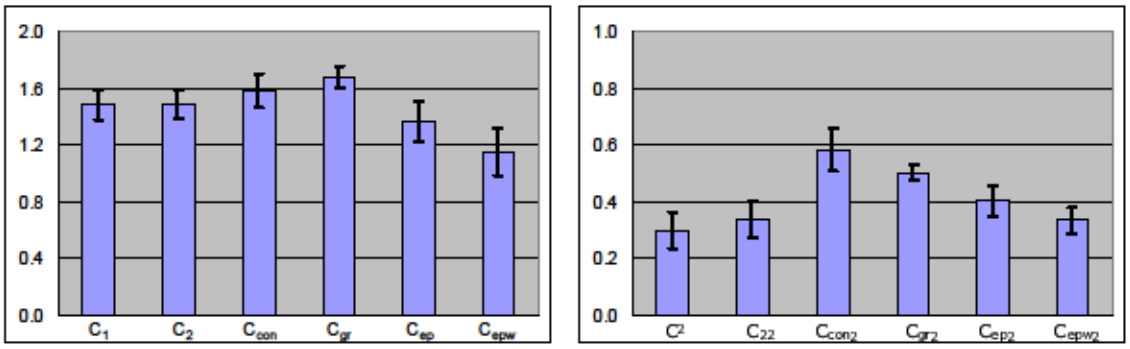
Each row represents the F-measure values for one of the five document subsets where randomly selected 50% objects were excluded from each subset.

EPaCH also outperforms clustering done on concatenated feature sets using the hierarchical algorithm. Again, EPaCHW produces better quality clusters compared to EPaCH. Both EPaCH and EPaCHW achieve signif cantly better performance than the graph based clustering on concatenated feature sets for all the validation indices with all datasets.

The results in Table 5.11 showing paired T-tests done on selected hypotheses for overlapping object sets conf rm our conclusions. It should be noted that the T-statistics are higher compared to the T-statistics that we obtained when comparing algorithms with identical object sets for (1) EPaCH and clustering on concatenated feature sets, (2) EPaCH and graph-based clustering, and (3) EPaCHW and EPaCH. This implies the performance improvement in more prominent when the algorithms are used with datasets with overlapping objects. These results indicate that the EPaCH family of algorithms can effectively handle related heterogenous datasets in which some of the objects represented are common, but some are not.

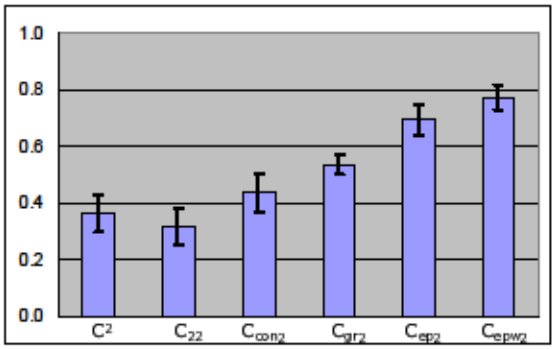
5.5.3 Effect of Decreasing Overlap between Datasets

Finally, we conducted an experiment to observe the effect of decreasing the amount of overlap between datasets. Previously, the results we presented were based on datasets having perfect mapping and bi-regular mapping with 49.7% overlap. We achieved that by randomly removing 50% of the objects from each dataset. We also conducted experiments by randomly removing 25% and 75% of the objects from each dataset. In the f rst case,

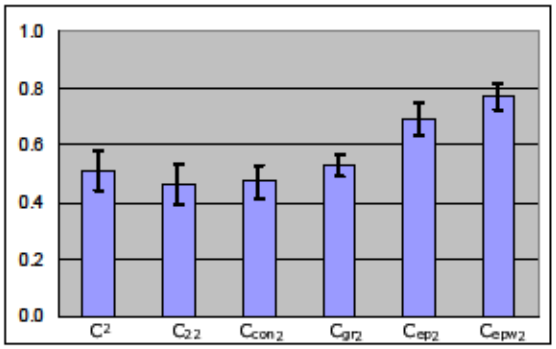


(a) MDB

(b) Entropy



(c) Purity



(d) F-Measure

Figure 5.4 Comparison of averages of different validation indices for EPaCH algorithms and baseline clustering schemes - results are for datasets with 50% of the objects randomly removed

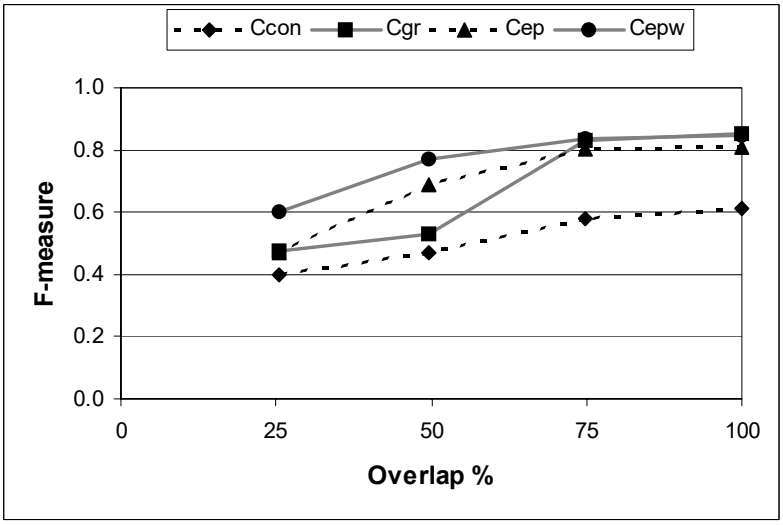
Table 5.11 The one-tailed T-test results for comparing EPaCH algorithms with other clustering schemes where $= 05$ and $T_{critical} = 1\,699$ - results are for datasets with 50% of the objects randomly removed

	T_{stat}	$P_{T \ t}$	T_{stat}	$P_{T \ t}$	T_{stat}	$P_{T \ t}$
	$\mathcal{C}_{con}$	$\mathcal{C}_b$	$\mathcal{C}_{con}$	$\mathcal{C}_w$	$\mathcal{C}_{ep}$	$\mathcal{C}_b$
MDB	-6.32	3.34e-07	-1.91	0.0334	2.19	0.018
Entropy	-20.76	2.98e-19	-13.37	3.14e-14	-14.01	9.61e-15
Purity	4.69	2.95e-05	8.82	5.22e-10	32.44	1.21e-24
F-Measure	-4.72	2.75e-05	1.78	0.0425	16.47	1.48e-16
	$\mathcal{C}_{ep}$	$\mathcal{C}_{con}$	$\mathcal{C}_{ep}$	$\mathcal{C}_{gr}$	$\mathcal{C}_{epw}$	$\mathcal{C}_{ep}$
MDB	6.94	6.33e-08	9.91	4.00e-11	7.52	1.37e-08
Entropy	13.70	1.71e-14	9.21	2.08e-10	8.88	4.51e-10
Purity	20.67	3.36e-19	13.83	1.34e-14	9.34	1.51e-10
F-Measure	20.01	8.14e-19	13.47	2.63e-14	8.89	4.47e-10

there was 74.8% overlap and in the second case, there was 25.6% overlap in the resulting datasets.

Figure 5.5 shows the change in the average F-measure for different clustering schemes with decreasing overlap between datasets. We chose F-measure because of its robustness. We tested EPaCH and EPaCHW against hierarchical clustering on concatenated feature sets and graph based clustering on concatenated feature sets. We selected the graph-based clustering algorithm since it demonstrated the best performance among the baseline clustering schemes.

It can be seen from 5.5 that although the graph based clustering on concatenated feature sets outperforms EPaCH and has performance similar to EPaCHW for identical object sets, both EPaCH and EPaCHW start outperforming graph based clustering on concate-



$\mathcal{C}_{con}$: Hierarchical clustering based on concatenated feature sets, $\mathcal{C}_{gr}$: Graph-based clustering with concatenated feature sets, $\mathcal{C}_{ep}$: Combination of individual hierarchical clusterings using EPaCH, $\mathcal{C}_{epw}$: Combination of individual hierarchical clusterings using EPaCHW

Figure 5.5 Changes in F-measure averages with decreasingly overlapped datasets

nated feature sets as the amount of overlap decreases. Only when the overlap among two datasets reaches approximately 25%, the graph based clustering on concatenated feature sets performs similar to EPaCH. But EPaCHW still outperforms the graph based clustering on concatenated feature sets at that point.

5.6 Summary

In the document clustering domain, EPaCH and EPaCHW were seen to yield higher quality clusters with heterogeneous datasets than hierarchical clustering based on individual datasets and hierarchical clustering based on concatenated feature sets. The algorithms also outperform the supertree and consensus tree methods and *k*-means applied to concate-

nated feature sets. EPaCHW consistently outperforms EPaCH. Graph-based partitional clustering with concatenated feature sets outperforms EPaCH when the datasets represent the same set of objects. EPaCHW appears to be slightly worse than graph-based clustering in this case, but the differences are not statistically signif cant. However, in cases where the two heterogeneous datasets have only a subset of objects in common, both EPaCH and EPaCHW signif cantly outperform all other methods including graph-based partitioning for all indices.

Therefore, EPaCH and EPaCHW offer a new approach for making use of the complementary knowledge contained in two related heterogeneous datasets. The two algorithms are particularly useful in cases where some objects are represented in both datasets, but some are different.

CHAPTER VI

PARTITIONAL CLUSTERING OF HETEROGENEOUS DATASETS USING MUTUAL ENTROPY

In chapter IV, we presented algorithms that can combine individual cluster hierarchies built from related heterogenous datasets. In this chapter, we present a class of algorithms called CEMENT (Cluster Ensemble using Mutual ENTropy) to address the problem of clustering two related datasets where the datasets represent the same or overlapping sets of objects but use different feature sets. These algorithms take the partitional clusters generated from two datasets as input and use a constraint-based approach to generate a single set of clusters. Our method uses an EM (expectation maximization) approach where the objective function is the mutual entropy between the two sets of clusters. We also present experimental results using the datasets described in chapter V and demonstrate the effectiveness of the CEMENT algorithms.

6.1 Motivation

Even though clustering is traditionally perceived to be an unsupervised process, in semi-supervised clustering [3, 6, 33, 50, 59, 64] framework, the performance of an unsupervised clustering algorithm can be improved with some supervision in the form of some

labeled data or constraints. In constraint-based semi-supervised clustering approaches, user-provided labels or some form of constraints are introduced using prior knowledge. These labels or constraints are then used to guide the clustering process. In the context of clustering heterogeneous datasets, once individual datasets are clustered, each clustering can be used to mutually inform the clustering of the other to provide a combined clustering of the two datasets.

The diff culty with many popular unsupervised clustering methods is that there is no clear notion of what a cluster is. A model based method like Expectation Maximization [47] treats the clustering problem as f nding a subpopulation with a certain distribution. When dealing with two individual clusterings and performing further cluster assignments based on constraints provided by mutual information, an EM approach can be used to maximize an appropriate objective function.

In the context of clustering heterogeneous datasets, the purpose of the objective function is to allow each individual clustering to inform the cluster membership of objects in the other clustering so that at the end of each iteration, the algorithm improves the overall cluster quality. Mutual entropy [45, 34, 52, 36, 67] can be used to represent the information contained in one random variable that can describe another random variable. A high mutual entropy represents a high similarity between the two variables. Given two individual clusterings, combining them to generate a single set of clusters can be viewed as maximizing the mutual entropy between the individual clusterings.

6.2 An EM Algorithm for Clustering Heterogenous Datasets

6.2.1 Model-based Clustering

Expectation Maximization (EM) is a probabilistic approach used for f nding the maximum likelihood estimates of hidden or hypothetical variables of a model. In general, EM algorithms use an iterative two-step process that converges after a number of *E* and *M* steps. The *E*-step computes an expectation of the likelihood of the hidden variables and the *M*-step maximizes the expected likelihood found in the *E*-step.

EM has been used in the context of model-based data clustering. Model based clustering is based on the framework of density estimation and views clustering as identifying the dense regions of the dataset. In model-based clustering, data are assumed to be generated by a mixture of underlying probability distributions in which each model represents a different group or cluster. Let $o_1\ o_2 \quad o_n$ be the set of n data points and $C_1\ C_2 \quad C_k$ be the set of k underlying groups. Let $f_j(o_i \quad _j)$ be the density of a data point o_i from the jth group C_j, where $\ _j$ is the set of corresponding parameters, and $\ _j$ is the probability that an observation belongs to the jth group ($_j \quad 0$ and $\sum_{j=1}^{k} \ _k = 1$). Let $= \ _1 \ \ _1 \quad \ _k$ and $= \ _1 \ \ _1 \quad \ _k$. By assuming that each data point is contained in one of the groups, the goal of EM clustering is to maximize the likelihood function [20]:

$$l(\quad) = \prod_{i=1}^{n} \sum_{j=1}^{k} \ _k f_j(o_i \quad _j)$$

The general approach is to initialize each model and then iteratively adjust the model to f t known data. In the E-step, the probability $f_j(o_i \quad _j)$ with which the object o_i would belong to class C_j is computed. In the M-step, the parameters of each class are adjusted so that it increases the likelihood of all the objects belonging to some cluster. EM clustering algorithms differ in the choice of the density function $f_j(o_i \quad _j)$. The choice of objective function is dependent on the form of the underlying probability distribution.

6.2.2 Problem Def nition

We will use some of the notation developed in Chapter III. Let us assume that we are dealing with two datasets $D_x = \ o_1^x \ o_2^x \qquad o_{n_x}^x$ and $D_y = \ o_1^y \ o_2^y \qquad o_{n_y}^y$ that consist of feature sets F_x and F_y. Here we will assume that $F_x = F_y$ and $D_x \quad D_y$, i.e., the two datasets may consist of the same objects or they may share some common objects.

Let us assume that each object is represented by a unique index $1 \ n$, where n is the total number of objects from both datasets, i.e., $n = \ D_x \ D_y$. Using this indexing, we can write, $D_x \quad D_y = \ o_1 \ o_2 \qquad o_n$. Let $V_x = \ v_1^x \ v_2^x \qquad v_n^x$ be an object representation vector, where v_i^x is one if o_i is contained in D_x and v_i^x is zero if o_i is not contained in D_x. Similarly, we def ne another object representation vector V_y corresponding to D_y.

Let $\mathcal{C}_x = \ C_1^x \ C_2^x \qquad C_k^x$ and $\mathcal{C}_y = \ C_1^y \ C_2^y \qquad C_k^y$ be the sets of clusters computed from the individual datasets D_x and D_y respectively. Let $\mathcal{A}_x = \ A_1^x \ A_2^x \qquad A_k^x$ be the set of cluster assignment vectors for $\mathcal{C}_x$, where $A_i^x = \ a_{i1}^x \ a_{i2}^x \qquad a_{in}^x$; a_{ij}^x is one if the object o_j is in cluster C_i^x and zero if object o_j is not in C_i^x. Similarly, we def ne

$\mathcal{A}_y = \begin{matrix} A_1^y & A_2^y & \cdots & A_k^y \end{matrix}$. Let $\begin{matrix} \overset{x}{1} & \overset{x}{2} & \cdots & \overset{x}{k} \end{matrix}$ and $\begin{matrix} \overset{y}{1} & \overset{y}{2} & \cdots & \overset{y}{k} \end{matrix}$ be the sets of mean vectors for $\mathcal{C}_x$ and $\mathcal{C}_y$ respectively. Our goal is to compute a single set of k partitional clusters $\mathcal{C} = \begin{matrix} C_1 & C_2 & \cdots & C_k \end{matrix}$.

6.2.3 The Probability Function

For our approach, we will def ne the probability density function $f_j(o_i \quad \overset{}{}_j)$ of the EM algorithm as a discrete probability function $p_j(o_i \quad \overset{u}{j})$. For an object o_i and class C_j^u, the probability $p_j(o_i \quad \overset{u}{j})$ is def ned as one if $\overset{u}{j}$ is the closest mean to o_i and zero otherwise. The parameters $\overset{u}{j}$ consist of the corresponding cluster assignment vector A_j^u and the mean vector $\overset{u}{j}$. Mathematically,

$$p_j(o_i \quad \overset{u}{j}) = \begin{matrix} 1 & \text{if } \operatorname{argmax}_k \left[s(o_i \quad \overset{u}{k}) \right] = j \\ 0 & \text{if } \operatorname{argmax}_k \left[s(o_i \quad \overset{u}{k}) \right] = j \end{matrix}$$

where $s(o_i \quad \overset{u}{k})$ is a similarity function between the object i and the mean vector for cluster C_k^u. The following constraints are also imposed:

$$\begin{matrix} n & k \\ & \\ i=1 & j=1 \end{matrix} p_j(o_i \quad \overset{x}{j}) = n_x$$

$$\begin{matrix} n & k \\ & \\ i=1 & j=1 \end{matrix} p_j(o_i \quad \overset{y}{j}) = n_y$$

6.2.4 The Likelihood Function

The likelihood function we use is the *mutual entropy* between the two partitions. The notion of mutual entropy is based on Shannon's information theory and quantifies the amount of information two random variables share with each other. If two random variables are independent, their mutual entropy is zero, i.e., none contains any information about the other. If they are identical, then all the information conveyed by one is shared by the other. The mutual entropy of two random variables X and Y is defined using the individual entropies and the joint entropy and can be written as [67]:

$$\sum_{i=1}^{n_x}\sum_{j=1}^{n_y} P(X = x_i\ Y = y_j) \log \frac{P(X = x_i\ Y = y_j)}{P(X = x_i)P(Y = y_j)}$$

In our case, each random variable is a given clustering and each possible value of a random variable is a cluster assignment vector. $P(X = x_j)$ is estimated as the fraction of the objects that are in a cluster in the first clustering and $P(X = x_i\ Y = y_j)$ as the fraction of the objects that are common in two clusters from the two clusterings. Let n_i^x be the number of objects in cluster i in $\mathcal{C}_x$, i.e., $n_i^x = C_i^x$ and n_i^y be the number of objects in C_i^y. Let n_{ij} be the number of objects that are common in cluster i of $\mathcal{C}_x$ and cluster j of $\mathcal{C}_y$, i.e., $n_{ij} = C_i^x \quad C_j^y$.

Then the mutual entropy between the two clustering $\mathcal{C}_x$ and $\mathcal{C}_y$ can be written as:

$$\sum_{i=1}^{k}\sum_{j=1}^{k} \frac{n_{ij}}{n} \log \frac{n_{ij}\ n}{(n_i^x\ n)\ n_j^y\ n} = \frac{1}{n} \sum_{i=1}^{k}\sum_{j=1}^{k} n_{ij} \log \frac{n_{ij} n}{n_i^x n_j^y}$$

For brevity, we will use $_{xy}(t)$ to represent the mutual entropy between the two clustering $\mathcal{C}_x$ and $\mathcal{C}_y$ in iteration t. The motivation for considering the mutual entropy as the likelihood function stems from its ability to measure a general dependence among random variables. In our case, when the mutual entropy between the two partitions is large, it means they are more similar. The purpose of our algorithm will be to increase the value of this mutual entropy through convergence so that we obtain more meaningful partitions in the context of the two heterogeneous datasets.

6.2.5 Algorithm Overview

In this section, we present an overview of the f rst of our CEMENT algorithms. We call this CEMENT$_1$. Our algorithm is based on computing a set of seed clusters and then recomputing each clustering around the seed clusters iteratively so that the mutual entropy between the two clusterings converges. At the end, a f nal adjustment step is carried out to generate a single set of clusters.

Def nition 7 Let D_x and D_x be two datasets and $D_x \quad D_y = \ o_1 \ o_2 \quad o_n \ $. We def ne a set of seed clusters $\mathcal{S} = \ S_1 \ S_2 \quad S_k \ $ where each seed cluster S_i consists of objects that are in both D_x and D_y, i.e., $S_i \quad D_x \quad D_y$. Let $V_s = \ v_{s1} \ v_{s2} \quad v_{sn} \ $ be an object representation vector corresponding to $\mathcal{S}$, where v_{si} is one if o_i is contained in any of the seed clusters and v_{si} is zero if o_i is not. Let $\mathcal{A}_s = \ A_1^s \ A_2^s \quad A_k^s \ $ be the set of cluster assignment vectors for $\mathcal{S}$, where $A_i^s = \ a_{i1}^s \ a_{i2}^s \quad a_{in}^s \ $; a_{ij}^s is one if the object o_j is in seed cluster S_i and zero if object o_j is not in S_i.

The algorithm starts by computing two individual partitional clusterings using the two different datasets. Each clustering is assumed to generate k clusters. We can use any suitable partitional clustering algorithm for this purpose. Then we will generate k seed clusters and further cluster ref nement will be performed using these seed clusters. An important feature of an EM algorithm is that it does not put equal importance on all the data points when computing a model. Similarly, in our method, we will only consider the pairs of clusters from the two clusterings having a "good" match to compute the seed clusters. We use the seed clusters as constraints to guide subsequent ref nement of each clustering. In a particular iteration, we will not reassign any of the seed objects (objects belonging to seed clusters). Rather, the non-seed objects will be reassigned to appropriate clusters. After generating the seed clusters, the mean vectors are computed for both datasets based on the seed objects.

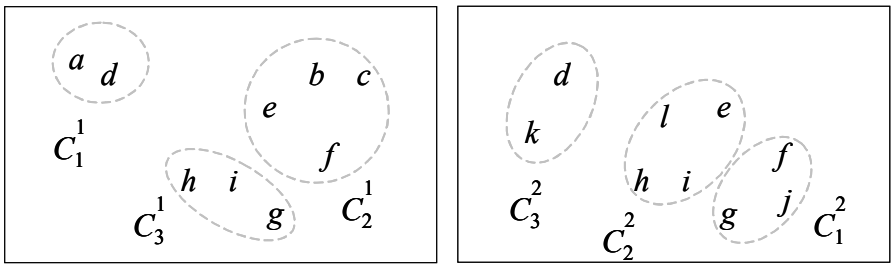
To generate seed clusters, we identify k pairs of similar clusters from the two clusterings. One straightforward approach for computing the similarity between two clusters is to count the number of common objects shared between the clusters. However, this will often create a tie between two pairs of clusters and may select large diverse clusters over smaller purer clusters. Instead, we will use the following information theoretic measure to compute the similarity between C_i^x and C_j^y:

$$\frac{2n_{ij}}{n_i^x + n_j^y}$$

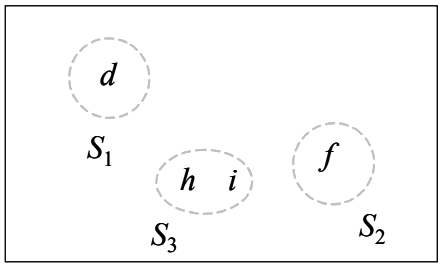
This measure will give us a continuous value between 0 and 1. If there is no common object between C_i^x and C_j^y, it will evaluate to 0, and if the clusters are exactly the same, it will evaluate to 1. We need to compute the similarity between all pairs of clusters.

We considered two methods for pairing similar clusters from the two datasets. In one approach, the seed clusters are based on the "best" matching pairs of clusters. Unfortunately, this is a combinatorial optimization problem and is therefore computationally intensive if there are many clusters. Instead we have used a greedy approach for selecting cluster pairs where, for each of the k clusters from one clustering, we select the best matching cluster from the other clustering that has not yet been paired with a cluster from the first set of clusters. Once the k pairs of similar clusters are identified, the seed clusters are computed from the intersection of each pair of clusters.

Let us explain the seed cluster computation with an example as shown in Figure 6.1. There are two datasets, $D_1 = \{a, b, c, d, e, f, g, h, i\}$ and $D_2 = \{d, e, f, g, h, i, j, k, l\}$. The first set of clusters $\mathcal{C}_1$ consists of three clusters $\{a, d\}$, $\{b, c, e, f\}$, and $\{g, h, i\}$. The second set of clusters $\mathcal{C}_2$ consists of three clusters $\{f, g, j\}$, $\{e, h, i, l\}$, and $\{d, k\}$. For the first cluster on the first clustering, C_1^1, the best match is the third cluster on the second clustering, C_3^2. These two clusters will be used to generate the first seed cluster $S_1 = C_1^1 \cap C_3^2 = \{d\}$. Then, for the second cluster on the first clustering, C_2^1, the best match is the first cluster on the second clustering, C_1^2. These two clusters will be used to generate the second seed cluster $S_2 = C_2^1 \cap C_1^2 = \{f\}$. Finally, the only match for C_3^1 is C_2^2 and $S_3 = C_3^1 \cap C_2^2 = \{h, i\}$.



(a) The clustering $\mathcal{C}_1$ (b) The clustering $\mathcal{C}_2$



(c) The seed clusters $\mathcal{S} = \ S_1 \ S_2 \ S_3$

Figure 6.1 Computing seed clusters from two partitional clusterings

Once seed clusters have been computed, the algorithm progresses iteratively through an E-step and an M-step. In the E-step, for each dataset, the discrete probability function is computed for each non-seed object belonging to the dataset. In the M-step, for each dataset, the cluster assignment vector is recomputed for each non-seed object belonging to the dataset. Thus, in the maximization step, essentially each cluster mean makes a shift towards the mean of the cluster expected to contain data elements in the context of both datasets. Before the start of next E-step, seed clusters and the two mean vectors are recomputed. These EM steps are repeated as long as the mutual entropy between the two separate clusterings increases. We def ne a stopping criteria threshold , where is the number of previous iterations for which $_{xy}(t)$ will be averaged to check for convergence. These iterative EM steps will allow the individual clustering to be ref ned using the information from each other.

Let us explain one EM iteration with the example shown in Figure 6.1. As explained before, the initial seed clusters will be d , f , and $h\ i$. These will replace the existing clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$ and two sets of mean vectors will be computed. Then, each non-seed object from D_1, i.e., a, b, c, e, and g, will be assigned to one of the three initial clusters in $\mathcal{C}_1$. Similarly, each non-seed object from D_2, i.e., e, g, j, k, and l, will be assigned to one of the three initial clusters in $\mathcal{C}_2$.

After the EM steps converge, the seed clusters will be computed one more time. There will still be some objects in both the datasets that will not be in any of the seed clusters. So, we will perform a f nal adjustment step that will assign each of these objects to one of

the seed clusters. In this step, there can be two situations. First, the non-seed object can be in both the datasets. In this case, the object will be reassigned to the seed cluster that has the closest mean based on both the datasets. Second, the non-seed object can be in one set but not in the other. In this case, the object will be reassigned to the closest seed cluster based on the dataset it comes from.

Even though the algorithm presented here is a specialized version of EM and deals with hard clustering, it can be generalized to address probabilistic clustering where data points can belong to multiple clusters with different probabilities. Figure 6.2 describes the algorithm. The input to the algorithm is two sets of k partitional clusters computed using two different datasets, $\mathcal{C}_x = \begin{array}{cccc} C_1^x & C_2^x & & C_k^x \end{array}$ and $\mathcal{C}_y = \begin{array}{cccc} C_1^y & C_2^y & & C_k^y \end{array}$. The output of the algorithm is a single set of k partitional clusters, $\mathcal{C} = \begin{array}{cccc} C_1 & C_2 & & C_k \end{array}$. Initially the value of the number of iterations t is set to zero.

6.2.6 Complexity

In the initialization step of the algorithm, the computationally expensive operation is the computation of seed clusters. There are k^2 pairs of clusters where the k is the number of clusters and the computation of the similarity between each pair of clusters requires $O((n\ \ k)^2)$ operations. Hence, the computation of seed clusters requires $O(n^2)$ operations. In the EM step, the algorithm needs to perform linear scans of all the objects on a per-cluster basis. If n is the number of total objects from both the datasets and k is the number of clusters, then the complexity of the EM step is $O(2nk)$. If t is the

1.Initialization:
  (a) Compute $\mathcal{S} = S_1\ S_2 \quad S_k$
  (b) **for** $i = 1$ **to** n **do**
        for $j = 1$ **to** k **do**
          $a^x_{ji} \quad a^y_{ji} \quad a^s_{ji}$
  (c) Compute $\begin{smallmatrix} x & x & & x \\ 1 & 2 & & k \end{smallmatrix}$ and $\begin{smallmatrix} y & y & & y \\ 1 & 2 & & k \end{smallmatrix}$
2.Expectation/Maximization:
  (a) **for** $i \quad 1$ **to** n **do**
        if $v_{si} = 0$ **then**
          for $j \quad 1$ **to** k **do**
            $a^x_{ji} \quad v^x_i \quad p_j(o_i \quad \begin{smallmatrix} x \\ j \end{smallmatrix})$
            $a^y_{ji} \quad v^y_i \quad p_j(o_i \quad \begin{smallmatrix} y \\ j \end{smallmatrix})$
  (b) $t \quad t+1$
3.Repeat 1-2 while $_{xy}(t) > \underline{1} \quad \begin{smallmatrix} t \\ i=t \end{smallmatrix} \quad _{xy}(i)$
4.Final Adjustment:
  (a) Perform Step 1
  (b) **for** $j = 1$ **to** k **do**
        $C_j \quad S_j$
  (c) **for** $i = 1$ **to** n **do**
        if $v_{si} = 0$ **then**
          $p \quad \text{argmax}_k \ \max(v^x_i s(o_i \quad \begin{smallmatrix} x \\ k \end{smallmatrix}) \ v^y_i s(o_i \quad \begin{smallmatrix} y \\ k \end{smallmatrix}))$
          $C_p \quad C_p \quad o_i$

Figure 6.2 The CEMENT$_1$ algorithm

number of iterations required for convergence, then the initialization and EM steps will be performed t times, resulting in an overall complexity of $O(t(2nk + n^2))$ for these two steps. In the final adjustment step, the seed clusters will be computed once ($O(n^2)$) and each non-seed object will be assigned to a cluster ($O(n)$). Thus the overall complexity is $O(t(2nk + n^2) + n^2 + n)$. In general, $n \quad k$ and $n \quad t$, and the complexity can be formalized as $O(n^2)$.

6.3 CEMENT$_2$ - A Modification of CEMENT$_1$

Since the CEMENT$_1$ algorithm works around a set of seed clusters, it does not take full advantage of the distribution of the other dataset. We argue that a mutual cluster refinement, where one clustering can be refined using the distribution of objects within the other cluster, can yield better quality clusters. Also, the computation of the seed clusters at the beginning of each iteration in CEMENT$_1$ requires $O(n^2)$ operations. In this section, we present a variation of the CEMENT$_1$ algorithm that will not require the computation of seed clusters at each iteration. Rather the seed clusters need to be computed only once during the final adjustment step. We call this algorithm CEMENT$_2$.

6.3.1 Algorithm Overview

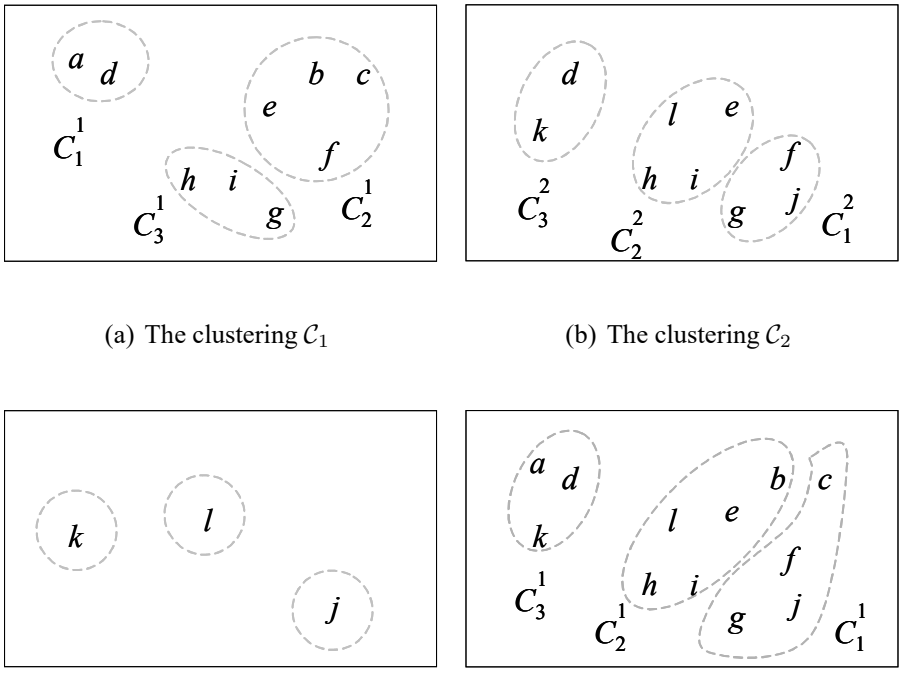
As with CEMENT$_1$, this algorithm will start by collecting two sets of clusters generated from two datasets. In the EM step, it will make adjustments to the first set of clusters using the second set of clusters. In order to do this, before the start of each EM-step, it will create a temporary set of clusters. This temporary set of clusters will be initially empty

and will correspond to the clusters from the second set of clusters. A mean vector will be computed for each of these initial clusters using only the objects that are in both datasets. The mean computation will be based on the feature vectors of the first dataset. After the mean vectors have been computed, the objects that are exclusively in the second dataset will be added to the respective clusters. Then, each object that appears in the first dataset will be added to the most similar temporary cluster. At the end of this, each temporary cluster will consist of objects from both datasets and the set of temporary clusters will replace the existing clusters in the first set of clusters. The same process will be repeated for the second set of clusters, i.e., the clusters in the second set of clusters will be recomputed using the first set of clusters. This process is similar to a semi-supervised approach in the sense that one set of clusters guides the computation of the other set of clusters.

At the end of each iteration, the mutual entropy between the two sets of clusters will be computed. The EM iterations will continue as long as there is a positive change in the mutual entropy. This will be performed the same way as CEMENT$_1$. When the EM iterations converge, the final adjustment step will be performed. The final adjustment step will be similar to CEMENT$_1$.

Let us explain the algorithm with the example shown in Figures 6.3(a) and 6.3(b). The first set of clusters $\mathcal{C}_1$ consists of three clusters $a\ d$, $b\ c\ e\ f$, and $g\ h\ i$. The second set of clusters $\mathcal{C}_2$ consists of three clusters $f\ g\ j$, $e\ h\ i\ l$, and $d\ k$. The process will start by creating a set of three temporary clusters. The mean vectors will be computed for these three clusters using $f\ g$, $e\ h\ i$, and d respectively. Once the

mean vectors are computed, as shown in Figure 6.3(c), the temporary clusters will consist of j , l , and k since j, l, and k are the objects exclusively appearing in D_2. Then, each object of D_1 will be assigned to one of the temporary clusters. Let us assume that this reassignment results in the clustering shown if Figure 6.3(d). This is essentially the readjustment of $\mathcal{C}_1$ using $\mathcal{C}_2$. The same process will be repeated to perform a readjustment of $\mathcal{C}_2$ using the original $\mathcal{C}_1$. This process will be repeated until it converges and then the final adjustment will be carried out as explained for CEMENT$_1$.



(a) The clustering $\mathcal{C}_1$ (b) The clustering $\mathcal{C}_2$

(c) Creating temporary clusters from $\mathcal{C}_2$ (d) Adjusting $\mathcal{C}_1$ using temporary clusters

Figure 6.3 Adjustment of one clustering using another in CEMENT$_2$

CEMENT$_2$ can easily be used as an ensemble algorithm to deal with the same object sets. In that case, the cluster assignment of one clustering will be used to compute a set of mean vectors and the cluster assignment of the other clustering will be readjusted using these mean vectors. Figure 6.4 describes the CEMENT$_2$ algorithm. The input to the algorithm is two sets of k partitional clusters computed using two different datasets, $\mathcal{C}_x = \begin{array}{cccc} C_1^x & C_2^x & & C_k^x \end{array}$ and $\mathcal{C}_y = \begin{array}{cccc} C_1^y & C_2^y & & C_k^y \end{array}$. The output of the algorithm is a single set of k partitional clusters, $\mathcal{C} = \begin{array}{cccc} C_1 & C_2 & & C_k \end{array}$. Initially the value of the number of iterations t is set to zero. The algorithm uses two sets of temporary clusters $\mathcal{C}_x$ and $\mathcal{C}_y$[1].

6.3.2 Complexity

In the initialization and EM step of the algorithm, the algorithm needs to perform linear scans of all the objects on a per-cluster basis. If n is the number of total objects from both the datasets, k is the number of clusters, and t is the number of iterations required for convergence, then the overall complexity of these steps is $O(2nkt)$. As in CEMENT$_1$, the final adjustment step will require $O(n^2)$ operations. Hence, the overall complexity is $O(n^2)$. Even though CEMENT$_2$ has the same complexity as CEMENT$_1$ and does not require the tn^2 operations for the seed cluster generation in the convergence loop, it will be a little bit slower. It will require a few more iterations for convergence since the first few iterations may change the distribution of objects within each set of clusters significantly.

[1]For describing the CEMENT$_2$ algorithm, we use x and y to represent the parameters corresponding to the respective temporary clusterings.

1. Initialization:
   (a) **for** $i = 1$ **to** n **do**
       for $j = 1$ **to** k **do**
           a^x_{ji} v^x_i v^y_i a^y_{ji}
           a^y_{ji} v^x_i v^y_i a^x_{ji}
   (b) Compute $\begin{smallmatrix} x & x & & x \\ 1 & 2 & & k \end{smallmatrix}$ and $\begin{smallmatrix} y & y & & y \\ 1 & 2 & & k \end{smallmatrix}$
   (c) **for** $i = 1$ **to** n **do**
       for $j = 1$ **to** k **do**
           a^x_{ji} $(1$ $v^x_i)$ a^y_{ji}
           a^y_{ji} $(1$ $v^y_i)$ a^x_{ji}
2. Expectation/Maximization:
   (a) **for** i 1 **to** n **do**
       for j 1 **to** k **do**
           a^x_{ji} v^x_i $p_j(o_i$ $\begin{smallmatrix} x \\ j \end{smallmatrix})$
           a^y_{ji} v^y_i $p_j(o_i$ $\begin{smallmatrix} y \\ j \end{smallmatrix})$
   (b) **for** $i = 1$ **to** n **do**
       for $j = 1$ **to** k **do**
           a^x_{ji} a^x_{ji}
           a^y_{ji} a^y_{ji}
   (c) t $t + 1$
3. Repeat 1-2 while $_{xy}(t) > \frac{1}{\begin{smallmatrix} t \\ i=t \end{smallmatrix}}$ $_{xy}(i)$
4. Final Adjustment:
   (a) Compute $\mathcal{S} =$ S_1 S_2 S_k
   (b) **for** $i = 1$ **to** n **do**
       for $j = 1$ **to** k **do**
           a^x_{ji} a^y_{ji} a^s_{ji}
   (c) Compute $\begin{smallmatrix} x & x & & x \\ 1 & 2 & & k \end{smallmatrix}$ and $\begin{smallmatrix} y & y & & y \\ 1 & 2 & & k \end{smallmatrix}$
   (d) **for** $j = 1$ **to** k **do**
       C_j S_j
   (e) **for** $i = 1$ **to** n **do**
       if $v_{si} = 0$ **then**
           p argmax_k $\max(v^x_i s(o_i$ $\begin{smallmatrix} x \\ k \end{smallmatrix})$ $v^y_i s(o_i$ $\begin{smallmatrix} y \\ k \end{smallmatrix}))$
           C_p C_p o_i

Figure 6.4 The CEMENT$_2$ algorithm

Also, after the first EM iteration, both sets of clusters will consist of the combined set of objects. This will require CEMENT$_2$ to reassign more objects to different clusters.

6.4 Results of Algorithm Evaluation

6.4.1 Results with Datasets having Identical Sets of Objects

Our experiments with the CEMENT algorithms were performed using the datasets described in section 5.1. To test the effectiveness of these two algorithms, we used several baselines. The first set of baselines was the partitional clusterings based on individual feature sets. For these baselines, we performed graph-based partitional clustering using the METIS[2] software. Another baseline was partitional clustering based on a concatenated feature set obtained from the two respective feature sets and we used the same graph-based approach for the clustering. We also compared the performance of the CEMENT algorithms against the ensemble algorithm called Cluster-based Similarity Partitioning Algorithm (CSPA) proposed by Strehl and Ghosh [53]. As done in the experiments presented in Chapter V, ten clusters were generated during each clustering.

Table 6.1 - Table 6.4 present the evaluation results for the four different validation indices. Each table presents a particular index for individual clustering ($\mathcal{C}_1$ $\mathcal{C}_2$), clustering with concatenated feature sets ($\mathcal{C}_{con}$), combining individual clusterings using the CSPA algorithm ($\mathcal{C}_{cspa}$), CEMENT$_1$ ($\mathcal{C}_{cem1}$) and CEMENT$_2$ ($\mathcal{C}_{cem2}$). For each feature set or a combination of feature sets, five subsets were used. Each row represents the particular

[2] `http://www-users.cs.umn.edu/~karypis/metis/metis/index.html`

index for one of the five document subsets. Note that smaller values are better for DB-index and entropy and larger values are better for purity and F-measure.

It can be seen from Table 6.1 - Table 6.4 that CEMENT$_1$ and CEMENT$_2$ outperform the other baseline schemes for all four validation indices in all instances. CEMENT$_2$ is seen to perform slightly better than CEMENT$_1$. It is also evident from the values of the validation measures that the clusters produced by CEMENT$_1$ and CEMENT$_2$ are high-quality clusters. If we consider the F-measure in Table 6.4, it can be seen that those values for CEMENT$_1$ and CEMENT$_2$ are around 0.9. Note that the range of possible F-measure is 0-1 with 1 being the best. If we consider the entropy measure in Table 6.2, it can be seen that those are around 0.15 where the range is 0-1 with 0 being the best.

Figure 6.5 shows a graphical representation of the comparison of different validation index for CEMENT$_1$ and CEMENT$_2$ with the baseline clustering schemes. The results shown are averages over all observations (thirty subsets) along with the standard deviation. Each subfigure shows the results for one of the four validation indices used in the experiment. It can be seen that the CEMENT algorithms yield higher quality clusters. Also, the standard deviations are lowest for CEMENT$_1$ and CEMENT$_2$ and this implies the stability of their performance. Even though CEMENT$_2$ demonstrates slightly better performance compared to CEMENT$_1$, it has a slightly higher standard deviation compared to CEMENT$_1$. Note that, as in previous experiments, clustering with concatenated feature sets does not always produce higher quality clusters compared to clustering based on individual data sets.

Table 6.1 Comparison of modif ed DB-index for CEMENT algorithms

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cspa}$	$\mathcal{C}_{cem1}$	$\mathcal{C}_{cem2}$
Syntactic and node-based semantic feature sets						
Dataset#1	0.711	0.952	0.697	0.844	0.498	0.457
Dataset#2	0.888	0.872	0.877	0.906	0.649	0.635
Dataset#3	0.686	0.604	0.636	0.694	0.497	0.471
Dataset#4	0.708	0.791	0.802	0.777	0.551	0.540
Dataset#5	0.745	0.739	0.776	0.771	0.551	0.550
Syntactic and edge-based semantic feature sets						
Dataset#1	0.711	0.861	0.776	0.887	0.533	0.484
Dataset#2	0.888	0.928	0.860	0.929	0.658	0.632
Dataset#3	0.686	0.632	0.612	0.641	0.430	0.431
Dataset#4	0.708	0.708	0.822	0.755	0.538	0.486
Dataset#5	0.745	0.707	0.782	0.750	0.535	0.546
Syntactic and node-and-edge-based semantic feature sets						
Dataset#1	0.711	0.719	0.772	0.780	0.499	0.480
Dataset#2	0.888	0.908	0.998	0.908	0.617	0.641
Dataset#3	0.686	0.720	0.605	0.738	0.476	0.461
Dataset#4	0.708	0.808	0.772	0.796	0.561	0.541
Dataset#5	0.745	0.759	0.621	0.835	0.571	0.536
Node-based and edge-based semantic feature sets						
Dataset#1	0.952	0.861	0.773	0.934	0.596	0.547
Dataset#2	0.872	0.928	0.916	0.921	0.654	0.620
Dataset#3	0.604	0.632	0.575	0.702	0.433	0.379
Dataset#4	0.791	0.708	0.711	0.759	0.528	0.490
Dataset#5	0.739	0.707	0.723	0.743	0.543	0.522
Node-based and node-and-edge-based semantic feature sets						
Dataset#1	0.952	0.719	0.714	0.837	0.555	0.538
Dataset#2	0.872	0.908	0.901	0.901	0.651	0.610
Dataset#3	0.604	0.720	0.550	0.715	0.464	0.415
Dataset#4	0.791	0.808	0.719	0.824	0.589	0.575
Dataset#5	0.739	0.759	1.360	0.752	0.515	0.504
Edge-based and node-and-edge-based semantic feature sets						
Dataset#1	0.861	0.719	0.905	0.846	0.564	0.540
Dataset#2	0.928	0.908	0.960	0.930	0.621	0.621
Dataset#3	0.632	0.720	0.527	0.666	0.423	0.394
Dataset#4	0.708	0.808	0.740	0.804	0.588	0.543
Dataset#5	0.707	0.759	0.759	0.738	0.510	0.487

Each row represents the modif ed DB-index for one of the f ve document subsets.

Table 6.2 Comparison of entropy for CEMENT algorithms

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cspa}$	$\mathcal{C}_{cem1}$	$\mathcal{C}_{cem2}$
Syntactic and node-based semantic feature sets						
Dataset#1	0.229	0.286	0.235	0.274	0.165	0.143
Dataset#2	0.271	0.278	0.28	0.281	0.171	0.162
Dataset#3	0.211	0.205	0.216	0.217	0.141	0.126
Dataset#4	0.257	0.276	0.27	0.276	0.166	0.150
Dataset#5	0.233	0.239	0.236	0.244	0.157	0.150
Syntactic and edge-based semantic feature sets						
Dataset#1	0.229	0.284	0.252	0.290	0.164	0.146
Dataset#2	0.271	0.301	0.279	0.309	0.177	0.159
Dataset#3	0.211	0.221	0.204	0.222	0.133	0.127
Dataset#4	0.257	0.245	0.286	0.265	0.157	0.138
Dataset#5	0.233	0.238	0.239	0.232	0.151	0.144
Syntactic and node-and-edge-based semantic feature sets						
Dataset#1	0.229	0.239	0.245	0.252	0.160	0.150
Dataset#2	0.271	0.281	0.305	0.283	0.168	0.160
Dataset#3	0.211	0.233	0.206	0.235	0.144	0.135
Dataset#4	0.257	0.267	0.253	0.271	0.176	0.157
Dataset#5	0.233	0.255	0.213	0.251	0.163	0.145
Node-based and edge-based semantic feature sets						
Dataset#1	0.286	0.284	0.246	0.300	0.183	0.169
Dataset#2	0.278	0.301	0.294	0.297	0.185	0.160
Dataset#3	0.205	0.221	0.194	0.233	0.136	0.114
Dataset#4	0.276	0.245	0.251	0.270	0.161	0.140
Dataset#5	0.239	0.238	0.237	0.244	0.175	0.160
Node-based and node-and-edge-based semantic feature sets						
Dataset#1	0.286	0.239	0.236	0.273	0.177	0.167
Dataset#2	0.278	0.281	0.297	0.294	0.172	0.158
Dataset#3	0.205	0.233	0.209	0.237	0.144	0.123
Dataset#4	0.276	0.267	0.267	0.281	0.176	0.158
Dataset#5	0.239	0.255	0.344	0.253	0.161	0.153
Edge-based and node-and-edge-based semantic feature sets						
Dataset#1	0.284	0.239	0.289	0.279	0.178	0.166
Dataset#2	0.301	0.281	0.286	0.303	0.176	0.161
Dataset#3	0.221	0.233	0.205	0.230	0.141	0.116
Dataset#4	0.245	0.267	0.259	0.268	0.170	0.149
Dataset#5	0.238	0.255	0.235	0.247	0.166	0.149

Each row represents the entropy for one of the f ve document subsets.

Table 6.3 Comparison of purity for CEMENT algorithms

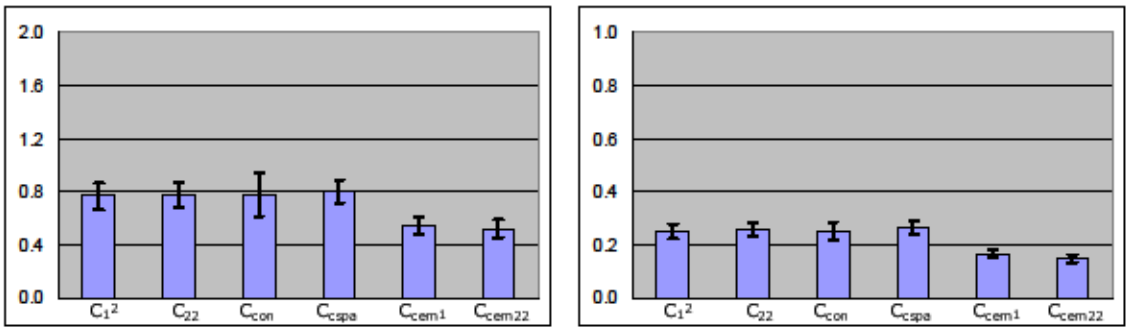
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cspa}$	$\mathcal{C}_{cem1}$	$\mathcal{C}_{cem2}$
Syntactic and node-based semantic feature sets						
Dataset#1	0.862	0.819	0.864	0.835	0.905	0.919
Dataset#2	0.839	0.835	0.836	0.835	0.897	0.903
Dataset#3	0.881	0.891	0.887	0.880	0.928	0.936
Dataset#4	0.853	0.841	0.844	0.842	0.909	0.918
Dataset#5	0.864	0.860	0.861	0.861	0.912	0.915
Syntactic and edge-based semantic feature sets						
Dataset#1	0.862	0.828	0.848	0.823	0.904	0.918
Dataset#2	0.839	0.823	0.835	0.820	0.895	0.904
Dataset#3	0.881	0.883	0.890	0.882	0.934	0.937
Dataset#4	0.853	0.859	0.835	0.845	0.914	0.926
Dataset#5	0.864	0.864	0.858	0.865	0.917	0.920
Syntactic and node-and-edge-based semantic feature sets						
Dataset#1	0.862	0.857	0.850	0.844	0.909	0.917
Dataset#2	0.839	0.835	0.812	0.835	0.902	0.903
Dataset#3	0.881	0.868	0.891	0.866	0.925	0.932
Dataset#4	0.853	0.843	0.852	0.842	0.902	0.914
Dataset#5	0.864	0.853	0.883	0.851	0.909	0.920
Node-based and edge-based semantic feature sets						
Dataset#1	0.819	0.828	0.851	0.818	0.890	0.900
Dataset#2	0.835	0.823	0.825	0.827	0.893	0.906
Dataset#3	0.891	0.883	0.896	0.873	0.933	0.946
Dataset#4	0.841	0.859	0.857	0.844	0.914	0.925
Dataset#5	0.860	0.864	0.863	0.857	0.904	0.912
Node-based and node-and-edge-based semantic feature sets						
Dataset#1	0.819	0.857	0.862	0.834	0.896	0.904
Dataset#2	0.835	0.835	0.819	0.826	0.899	0.907
Dataset#3	0.891	0.868	0.892	0.870	0.928	0.940
Dataset#4	0.841	0.843	0.851	0.837	0.904	0.914
Dataset#5	0.860	0.853	0.743	0.855	0.911	0.915
Edge-based and node-and-edge-based semantic feature sets						
Dataset#1	0.828	0.857	0.820	0.831	0.897	0.905
Dataset#2	0.823	0.835	0.812	0.821	0.899	0.905
Dataset#3	0.883	0.868	0.896	0.877	0.930	0.944
Dataset#4	0.859	0.843	0.853	0.843	0.907	0.920
Dataset#5	0.864	0.853	0.859	0.860	0.912	0.921

Each row represents the purity for one of the f ve document subsets.

Table 6.4 Comparison of F-measure for CEMENT algorithms

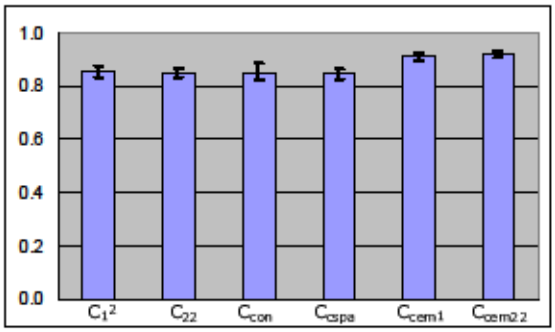
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_{con}$	$\mathcal{C}_{cspa}$	$\mathcal{C}_{cem1}$	$\mathcal{C}_{cem2}$
Syntactic and node-based semantic feature sets						
Dataset#1	0.862	0.819	0.863	0.834	0.904	0.919
Dataset#2	0.839	0.835	0.836	0.834	0.896	0.902
Dataset#3	0.880	0.890	0.886	0.879	0.928	0.936
Dataset#4	0.853	0.840	0.843	0.841	0.909	0.917
Dataset#5	0.864	0.860	0.860	0.860	0.912	0.915
Syntactic and edge-based semantic feature sets						
Dataset#1	0.862	0.828	0.848	0.822	0.903	0.918
Dataset#2	0.839	0.822	0.835	0.819	0.894	0.903
Dataset#3	0.880	0.882	0.890	0.881	0.934	0.937
Dataset#4	0.853	0.858	0.834	0.845	0.914	0.926
Dataset#5	0.864	0.863	0.857	0.864	0.917	0.919
Syntactic and node-and-edge-based semantic feature sets						
Dataset#1	0.862	0.856	0.850	0.843	0.908	0.916
Dataset#2	0.839	0.835	0.812	0.834	0.902	0.902
Dataset#3	0.880	0.867	0.890	0.865	0.925	0.932
Dataset#4	0.853	0.843	0.851	0.841	0.901	0.913
Dataset#5	0.864	0.852	0.882	0.850	0.909	0.920
Node-based and edge-based semantic feature sets						
Dataset#1	0.819	0.828	0.851	0.817	0.889	0.899
Dataset#2	0.835	0.822	0.825	0.826	0.892	0.906
Dataset#3	0.890	0.882	0.896	0.873	0.933	0.946
Dataset#4	0.840	0.858	0.857	0.843	0.914	0.925
Dataset#5	0.860	0.863	0.863	0.856	0.903	0.911
Node-based and node-and-edge-based semantic feature sets						
Dataset#1	0.819	0.856	0.861	0.832	0.896	0.903
Dataset#2	0.835	0.835	0.818	0.825	0.899	0.907
Dataset#3	0.890	0.867	0.891	0.870	0.928	0.940
Dataset#4	0.840	0.843	0.851	0.836	0.903	0.914
Dataset#5	0.860	0.852	0.745	0.854	0.911	0.915
Edge-based and node-and-edge-based semantic feature sets						
Dataset#1	0.828	0.856	0.819	0.831	0.896	0.904
Dataset#2	0.822	0.835	0.812	0.820	0.899	0.905
Dataset#3	0.882	0.867	0.896	0.876	0.930	0.944
Dataset#4	0.858	0.843	0.852	0.843	0.907	0.920
Dataset#5	0.863	0.852	0.859	0.859	0.911	0.921

Each row represents the purity for one of the f ve document subsets.

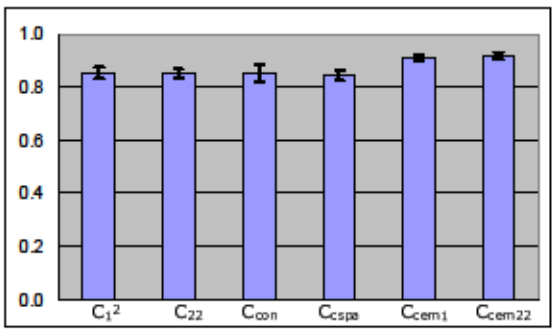


(a) MDB

(b) Entropy



(c) Purity



(d) F-Measure

Figure 6.5 Comparison of averages of different validation indices for CEMENT algorithms and baseline clustering schemes

To test the statistical signif cance of our results, we performed one-tailed paired T-tests. Table 6.5 shows the corresponding results. We used an value of .05 where $T_{critical} = 1\ 699$ for one-tail tests. The hypotheses tested are $\mathcal{C}_{cem1}$ $\mathcal{C}_b$ meaning that clusters generated by CEMENT$_1$ are of signif cantly higher quality than clusters obtained by clustering individual feature sets, $\mathcal{C}_{cem1}$ $\mathcal{C}_{con}$ meaning that clusters generated by CEMENT$_1$ are of signif cantly higher quality than clusters obtained using concatenated feature sets, $\mathcal{C}_{cem1}$ $\mathcal{C}_{cspa}$ meaning clusters generated by CEMENT$_1$ are of higher quality than clusters produced by combining the individual clusterings using the similarity partitioning ensemble algorithm, and $\mathcal{C}_{cem2}$ $\mathcal{C}_{cem1}$ meaning that clusters produced by CEMENT$_2$ are of higher quality than clusters produced by CEMENT$_1$.

Table 6.5 The one-tailed T-test results for comparing CEMENT algorithms with other clustering schemes where $= 05$ and $T_{critical} = 1\ 699$

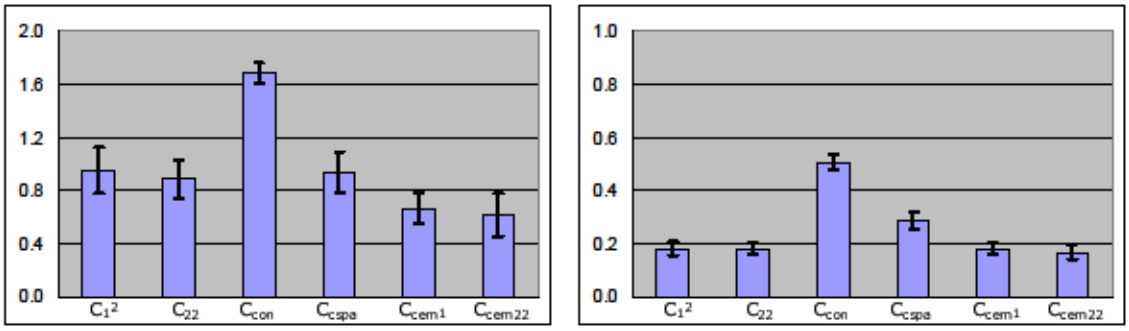
	T_{stat}	$P_{T\ t}$	T_{stat}	$P_{T\ t}$	T_{stat}	$P_{T\ t}$	T_{stat}	$P_{T\ t}$
	$\mathcal{C}_{cem1}$	$\mathcal{C}_b$	$\mathcal{C}_{cem1}$	$\mathcal{C}_{con}$	$\mathcal{C}_{cem1}$	$\mathcal{C}_{cspa}$	$\mathcal{C}_{cem2}$	$\mathcal{C}_{cem1}$
DB-Index	25.24	1.37e-21	19.56	1.51e-18	33.93	3.39e-25	6.77	9.96e-08
Entropy	30.23	8.86e-24	16.82	8.40e-17	35.11	1.30e-25	14.91	1.97e-15
Purity	32.92	7.99e-25	11.85	6.15e-13	38.55	9.06e-27	13.38	3.06e-14
F-Measure	34.00	3.21e-25	12.00	4.54e-13	39.35	5.06e-27	12.37	2.17e-13

Each row shows the T-test results for a particular validation index. For each hypothesis, the f rst column shows the T-value and the second column shows the p-values. A T-value of 1.699 or higher and a p-value of 0.05 or lower indicate evidence that the hy-

pothesis is true. The T-test results demonstrate that CEMENT$_1$ and CEMENT$_2$ are signif - cantly better than the baseline clustering schemes and this holds true for all four validation indices used in our experiment. Also, clustering based on a concatenated feature set does not always give signif cantly better performance compared to the individual clustering. Clustering based on a concatenated feature set outperforms the individual clusterings only for the F-measure index.

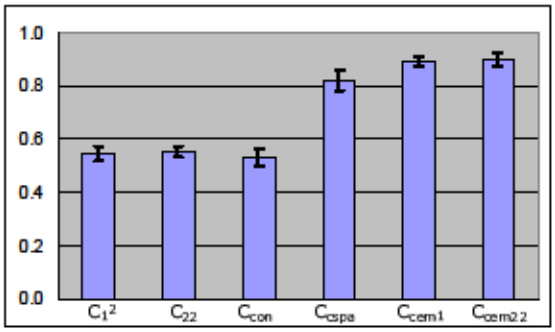
6.4.2 Results with Datasets having Overlapping Sets of Objects

Our experiments presented in this dissertation were performed using (1) two datasets having the same set of objects and (2) two datasets having some common objects. As done in Chapter V, we also used overlapping datasets by randomly removing 50% of the objects from each dataset. Figure 6.6 shows a graphical representation of the comparison of different validation index for CEMENT$_1$ and CEMENT$_2$ with clustering based on individual and concatenated feature sets. As before, the results shown are averages over all observations (thirty subsets) along with the standard deviation. When compared to clustering based on concatenated feature sets and the clustering generated by combining the individual clusterings using the similarity partitioning ensemble algorithm, we obtained higher quality clusters with both CEMENT$_1$ and CEMENT$_2$. This is true for all four validation indices. Also, CEMENT$_1$ and CEMENT$_2$ seem to have relatively low standard deviations for all four validation indices.

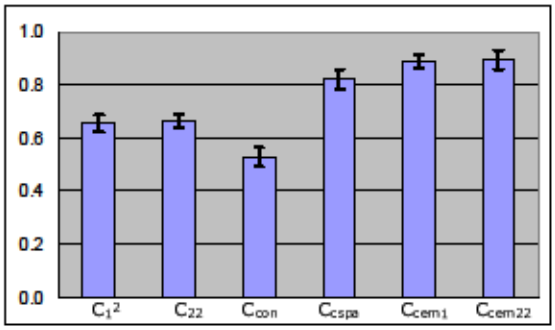


(a) MDB

(b) Entropy



(c) Purity



(d) F-Measure

Figure 6.6 Comparison of averages of different validation indices for CEMENT algorithms and baseline clustering schemes - results are for datasets with 50% of the objects randomly removed

When compared to the clustering based on the individual feature sets, the entropy measure gives better values for the individual clustering. Note that when clustering with the individual feature sets, we have 1000 objects in each datasets. On the other hand, when we are dealing with all the objects in both the datasets, we have approximately 50% more objects. This results in an increase of approximately 50% in the average cluster size. As explained in Chapter V, the entropy measure favors smaller clusters. Also, the individual clusterings are not actually compatible for performance comparison here. This is because the clusterings based on the individual feature sets do not represent the wider group of objects that our algorithm does.

If we compare Figure 6.5 and Figure 6.6, it is evident that CEMENT$_1$ and CEMENT$_2$ perform relatively better compared to the baseline clustering schemes when the two datasets are not equal but contain some common objects. Also, CEMENT$_2$ demonstrate slightly better performance compared to CEMENT$_1$. Even though CEMENT$_1$ runs a little faster than CEMENT$_2$, both exhibit fast convergence. In our experiments with the original datasets, for 30 different runs, CEMENT$_1$ needed approximately 5 iterations on average for convergence. On the other hand, CEMENT$_2$ needed approximately 7 iterations on average for convergence.

6.5 Summary

We presented a family of algorithms called CEMENT for combining partitional clusterings generated from heterogeneous datasets. These algorithms use mutual entropy to

converge towards having a maximal similarity between the two clusterings. We presented experimental results that show the effectiveness of the algorithms. The results demonstrated that for heterogeneous feature sets extracted from document collection, these algorithms yield high quality clusters. CEMENT algorithms can quickly converge to a final solution. The statistical tests also support our main research hypothesis that cluster combination produces better quality clusters compared to individual clustering and clustering based on concatenated feature sets. Even though the CEMENT algorithms were presented for discrete cluster membership, they can easily be generalized to a probabilistic form using soft cluster membership. This approach is expected to generate interesting clustering in some applications where the probability of membership in a cluster is meaningful.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

The traditional notion of clustering is based on a single dataset. In this dissertation, we addressed the problem of clustering multiple related heterogeneous datasets. This problem can be tackled in two ways. One approach is to integrate the individual feature sets into a unif ed feature space and then perform clustering on the unif ed feature space. The second approach is to cluster the individual datasets using the respective feature sets and then combine the resulting clusterings. The contribution of this dissertation was based on the second approach. In some domains, several feature sets may be available to represent the same objects, but it may not be easy to compute an integrated feature set. When the individual feature sets are complementary, we demonstrated that the second approach yields better quality clustering.

7.1 Contributions

This dissertation makes several contributions to the f eld of data mining. In particular, we developed general approaches and specif c algorithms to address the relatively new problem of clustering heterogeneous datasets. Not only are these approaches and algorithms expected to enrich the f eld of data mining, they can potentially have a signif -

cant impact in several application areas such as information retrieval, bioinformatics, and computer security.

We developed a notational framework for the problem of clustering multiple related heterogeneous datasets. To tackle this problem, we found it very important to introduce the notion of a "mapping" between the objects of two heterogeneous datasets. We used a bipartite graph to represent this mapping at a level of abstraction that describes different ways heterogeneous objects can be related. We used the notion of "mapping" to describe a general approach for combining heterogeneous cluster hierarchies that is based on the mutual ref nement of cluster hierarchies.

Ensemble approaches described in existing literature primarily deal with partitional clusterings. However, hierarchical clustering algorithms are very popular and the need to combine information from two hierarchical clusterings motivated the development of our methods for combining cluster hierarchies generated from heterogeneous datasets. We applied phylogenetic tree combination methods for combining hierarchical clusterings and observed mixed results. Even though these algorithms can combine multiple cluster hierarchies into a single dendrogram, they can not be effectively used to generate partitional clusters from multiple cluster hierarchies.

We developed a class of algorithms called EPaCH for generating a single set of partitional clusters from multiple hierarchical clusterings that can be used with heterogeneous datasets. These algorithms use a graph theoretic approach for combining the multiple hierarchies. They start by generating a weighted graph from the hierarchies based on the

association strengths of objects in the hierarchies. In the original version, called simply EPaCH, association strengths are calculated based on co-occurrence of objects in sub-clusters of the hierarchies. A graph partitioning algorithm is then applied to generate partitional clusters. In the second version, called EPaCHW, in addition to the co-occurrence of objects, the intra-cluster similarity of each sub-cluster is also used to compute the association strength. We showed how EPaCHW can utilize the intra-cluster similarity values to more effectively compute the association strength of objects and generate a more informative dendrogram compared to EPaCH. We have shown that the average-case complexity of the EPaCH algorithms is $O(n^2 \lg n)$ which is no worse than the complexity of the average-link agglomerative clustering algorithm.

We tested the EPaCH algorithms empirically with a collection of documents from ten different subject categories. Both syntactic and semantic feature sets were extracted and the resulting datasets were clustered individually using average-link agglomerative hierarchical clustering. EPaCH and EPaCHW were then used to generate a single set of partitional clusters from the dendrograms. Our experiments were performed taking two datasets at a time where each dataset was constructed from the same subset of documents using a different feature set. We considered two experimental settings. In one setting, each of the two datasets consisted of the same objects. In another setting, we randomly selected 50% objects from each dataset and excluded those objects.

In the document clustering domain, EPaCH algorithms were shown to yield higher quality clusters than clustering based on a single feature set, hierarchical clustering based

on concatenated feature sets, and phylogeny-based ensemble methods. A graph-based partitional algorithm using a concatenated feature set outperformed EPaCH when both datasets represented the same set of objects. EPaCHW was marginally inferior to graph-based partitional clustering on concatenated feature sets in the same situation. But, when the two object sets were not the same but shared some common objects, EPaCH and EPaCHW both signif cantly outperformed the graph-based clustering on concatenated feature sets. We also showed that as the overlap between two datasets decreases, EPaCH and EPaCHW increasingly outperform the other clustering schemes.

We also developed a class of algorithms called CEMENT for combining partitional clusterings generated from heterogeneous datasets. These algorithms are essentially expectation maximization methods and are based on maximizing an objective function. We used the mutual entropy between two clusterings as the objective function. In the original version, called CEMENT$_1$, a set of seed clusters are generated from the two clusterings and then subsequent cluster assignment is performed around the seed clusters. In the second version, called CEMENT$_2$, each clustering is used to ref ne the other clustering and this mutual reinforcement process continues until convergence. We showed that the complexity of these algorithms is $O(n^2)$. Empirical results showed that the CEMENT algorithms produce better quality clusters compared to individual clusterings, clustering based on the concatenated feature sets, and the well-studied CSPA (cluster-based similarity partitioning) ensemble algorithm.

We used several cluster validation indices to evaluate the cluster qualities. We showed that one of these, the DB-index, has some limitations and can not be used effectively to measure cluster quality when clusters are of non-spherical shape. We developed a modif ed version of the DB-index that is based on the "expected" inter-cluster and intra-cluster distances. Results of a simulation study suggested that the modif ed version gives more accurate quality measures compared to the original one with arbitrary shaped clusters.

7.2 Future Work

The paradigm of clustering heterogeneous datasets is relatively new. There are many additional issues that are worth investigating. We plan to extend this research along several directions.

We applied our algorithms in the document clustering domain. We want to test our algorithms using several datasets from the machine learning repository at UCI[1]. Some of these datasets consists of multiple feature types. We want to split a feature set into two and then construct two datasets based on the partial feature sets. These datasets can then be used to evaluate the effectiveness of our methods.

We want to extend our work to the biological domain where multiple heterogeneous datasets are all tightly bound and interconnected. With the availability of diverse biological datasets through public databases, it has become essential to be able to perform an integrated analysis of the data. The isolated clustering of the biological datasets tend to be less

[1] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

effective because of the presence of noise in the individual datasets. A unif ed clustering of these datasets can compensate for the noise and has the potential to reveal completely new biological information. This is expected to have far reaching consequences in discovering biological pathways.

We presented two classes of novel algorithms in the dissertation. Even though our experiments with the EPaCH algorithms were based on combining two datasets, EPaCH and EPaCHW can handle multiple (more than two) datasets. We want to carry out experiments to test the effectiveness of EPaCH and EPaCHW with more than two datasets. On the other hand, the CEMENT algorithms have been presented in the context of "two" related heterogeneous datasets. We want to generalize these so that they can be used with more than two datasets.

The CEMENT algorithms have been developed to generate hard clustering by using a discrete probability function for each object inside the EM loop. We want to modify the probability function so that it becomes a continuous probability function and utilize this to generate soft clustering where one object may belong to more than one clusters with certain probability. This is expected to be a more natural way of dealing with heterogeneous datasets.

The notion of mapping between two heterogeneous datasets was represented by a bipartite graph. We want to extend this to more than two datasets by considering a multi-layer abstraction, where each layer will represent one dataset. For obvious reasons, we plan to use an n-partite graph for this purpose.

The CEMENT algorithms are based on the convergence of the mutual entropy. The fact that these algorithms converge was based on an intuitive perspective and this was supported by empirical evidence. Nevertheless, we plan to develop a mathematical proof of convergence for the CEMENT algorithms.

We want to apply our methods in the computer security domain. Different intrusion detection system (IDS) senors are used to analyze audit log that contains normal and abnormal activities over a network. The output of two such sensors represent two different interpretations of the same set of events. We plan to apply different sensors on the publicly available DARPA IDS evaluation data[2] and then apply our methods on the resulting datasets.

[2] http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html

REFERENCES

[1] A. A. Alizadeh, M. Eisen, R. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. T. X. Yu, J. Powell, L. Yang, G. Marti, T. Moore, J. Hudson, L. Lu, D. Lewis, R. Tibshirani, G. Sherlock, W. Chan, T. Greiner, D. Weisenburger, J. Armitage, R. Warnke, , R. Levy, W. Wilson, M. Grever, J. Byrd, D. Botstein, and P. B. L. Staudt, "Distinct Types of Diffuse Large B-cell Lymphoma Identif ed by Gene Expression Prof ling," *Nature*, vol. 403, no. (6769, 2000, p. 503511.

[2] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel Circuit Partitioning," *Proceedings of 34th ACM/IEEE Conference on Design Automation, Anaheim, CA, June 9-13, 1997*, pp. 530–533.

[3] S. Basu, A. Banerjee, and R. J. Mooney, "Semi-supervised Clustering by Seeding," *Proceedings of 19th International Conference on Machine Learning (ICML-2002), Sydney, Australia, July 8-12, 2002*, pp. 19–26.

[4] S. Basu, M. Bilenko, and R. J. Mooney, "A Probabilistic Framework for Semi-Supervised Clustering," *Proceedings of 10th International Conference on Knowledge Discovery and Data Mining (KDD-2004), Seattle, WA, August 22-25, 2004*, pp. 59–68.

[5] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering Gene Expression Patterns," *Journal of Computational Biology*, vol. 6, no. 3/4, 1999, pp. 281–297.

[6] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating Constraints and Metric Learning in Semi-Supervised Clustering," *Proceedings of 21st International Conference on Machine learning (ICML-2004), Banff, Alberta, July 4-8, 2004*, pp. 81–88.

[7] D. Bryant, "A Classif cation of Consensus Methods for Phylogenetics," *Bioconsensus*, M. Janowitz, F. Lapointe, F. McMorris, B. Mirkin, and F. Roberts, eds., vol. 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, AMS Press, 2002, pp. 163–184.

[8] B. S. Caffo, W. Jank, and G. L. Jones, "Ascent-Based Monte Carlo EM," *Journal of the Royal Statistical Society, Series B*, vol. 67, no. 2, 2005, pp. 235–252.

[9] G. Celeux and G. Govaert, "A Classif cation EM Algorithm for Clustering and Two Stochastic Versions," *Computational Statistics & Data Analysis*, vol. 14, 1992, p. 315332.

[10] M. Collins, *Head-Driven Statistical Models for Natural Language Parsing*, doctoral dissertation, University of Pennsylvania, 1999.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd edition, McGraw-Hill, Cambridge, MA, 2002.

[12] I. Dagan, Z. Marx, and E. Shamir, "Cross-dataset Clustering: Revealing Corresponding Themes Across Multiple Corpora," *Proceedings of 6th Conference on Natural Language Learning, Taipei, Taiwan, August 31 - September 1, 2002*, pp. 15–21.

[13] D. Davis and D. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, 1979, pp. 224–227.

[14] A. Dayanik and C. Nevill-Manning, "Clustering in Relational Biological Data," http://www.cs.umd.edu/projects/srl2004/Papers/dayanik.pdf.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 34, 1977, pp. 1–38.

[16] I. S. Dhillon, "Co-clustering Documents and Words using Bipartite Spectral Graph Partitioning," *Proceedings of 7th International Conference on Knowledge Discovery and Data Mining (KDD-2001), San Francisco, CA, August 26-29, 2001*, pp. 269–274.

[17] M. Dunham, *Data Mining: Introductory and Advanced Topics*, Prentice Hall, Upper Saddle River, NJ, 2003.

[18] M. Eisen, P. Spellman, P. Brown, and D. Botstein, "Cluster Analysis and Display of Genome-wide Expression Patterns," *Proceedings of National Academy of Sciences USA*, vol. 95, no. 25, 1998, pp. 14863–14868.

[19] X. Z. Fern and C. E. Brodley, "Solving Cluster Ensemble Problems by Bipartite Graph Partitioning," *Proceedings of 21st International Conference on Machine learning (ICML-2004), Banff, Alberta, July 4-8, 2004*, p. 36.

[20] C. Fraley and A. E. Raftery, "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis," *The Computer Journal*, vol. 41, no. 8, 1998, pp. 578–588.

[21] A. Fred and A. Jain, "Data Clustering Using Evidence Accumulation," *Proceedings of 16 th International Conference on Pattern Recognition (ICPR'02), Quebec City, August 11-15*, 2002, pp. 276–280.

[22] A. D. Gordon, "Consensus Supertrees: The Synthesis of Rooted Trees Containing Overlapping Sets of Labeled Leaves," *Journal of Classif cation*, vol. 3, 1986, pp. 335–348.

[23] P. D. Green, J. Barker, M. P. Cooke, and L. Josifovski, "Handling Missing and Unreliable Information in Speech Recognition," *Proceedings of Eighth International Workshop on Artif cial Intelligence and Statistics, Key West, FLJanuary 4-7, 2001*.

[24] D. Gusf eld, "Eff cient Algorithms for Inferring Evolutionary Trees," *Networks*, vol. 21, 1991, pp. 19–28.

[25] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Franscisco, CA, 2001.

[26] E. Hartuv and R. Shamir, "A Clustering Algorithm Based on Graph Connectivity," *Information Processing Letters*, vol. 76, 2000, pp. 175–181.

[27] M. Hossain, S. Bridges, Y. Wang, and J. Hodges, "Combining Document Clusters Generated from Syntactic and Semantic Feature Sets using Tree Combination Methods," *Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources, Houston, TX, November 27, 2005*, pp. 16–24.

[28] X. Hu and I. Yoo, "Cluster Ensemble and its Applications in Gene Expression Analysis," *Proceedings of 2nd Asia-Pacif c Bioinformatics Conference (APBC 2004), Dunedin, New Zealand, January 18-22, 2004*, pp. 297–302.

[29] J. Jiang and D. Conrath, "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy," *Proceedings of International Conference on Research in Computational Linguistics, Taiwan, 1997*.

[30] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *Proceedings of International Conference on Parallel Processing, 1995*, pp. 113–122.

[31] G. Karypis and V. Kumar, "Multilevel k-way Hypergraph Partitioning," *Proceedings of 36th ACM/IEEE Conference on Design Automation, New Orleans, LA, June 21-25, 1999*, pp. 343–348.

[32] G. Karypis and V. Kumar, "Multilevel k-way Partitioning Scheme for Irregular Graphs," *Journal of Parallel Distributed Computing*, vol. 48, no. 1, 1998, pp. 96–129.

[33] D. Klein, S. D. Kamvar, and C. D. Manning, "From Instance-Level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering,"

Proceedings of 19th International Conference on Machine Learning (ICML-2002), Sydney, Australia, July 8-12, 2002, pp. 307–314.

[34] A. Kraskov, H. Stögbauer, R. G. Andrzejak, and P. Grassberger, "Hierarchical Clustering using Mutual Information," *Europhysics Letters*, vol. 70, no. 2, 2005, pp. 278–284.

[35] B. Larsen and C. Aone, "Fast and Effective Text Mining Using Linear-Time Document Clustering," *Proceedings of 5th International Conference on Knowledge Discovery and Data Mining (KDD99), San Diego, CA, August 15-18, 1999*, pp. 16–22.

[36] X. Liu, A. Krishnan, and A. Mondry, "An Entropy-Based Gene Selection Method for Cancer Classif cation using Microarray Data," *BMC Bioinformatics*, vol. 6, no. 76, 2005.

[37] J. B. MacQueen, "Some Methods for Classif cation and Analysis of Multivariate Observations," *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, June 21-July 18, 1965*, vol. 1, p. 281297.

[38] T. Margush and F. R. McMorris, "Consensus n-Trees," *Bulletin of Mathematical Biology*, vol. 43, no. 2, 1981, pp. 239–244.

[39] Z. Marx, I. Dagan, J. M. Buhmann, and E. Shamir, "Coupled Clustering: A Method for Detecting Structural Correspondence," *Journal of Machine Learning Research*, vol. 3, 2002, pp. 747–780.

[40] S. McClean, B. Scotney, and S. Robinson, "Conceptual Clustering of Heterogeneous Gene Expression Sequences," *Artif cial Intelligence Review*, vol. 20, no. 1-2, 2003, pp. 53–73.

[41] S. I. McClean, B. W. Scotney, and F. Palmer, "Conceptual Clustering of Heterogeneous Sequences via Schema Mapping," *Proceedings of 13th International Symposium on Foundations of Intelligent Systems, Lyon, France, June 27-29, 2002*, pp. 85–93.

[42] C. Ordonez, E. Omiecinski, and N. Ezquerra, "A Fast Algorithm to Cluster High Dimensional Basket Data," *Proceedings of IEEE International Conference on Data Mining, San Jose, CA, November 29 - December 2, 2001*, pp. 633–636.

[43] P. Pavlidis, J. Weston, J. Cai, and W. Grundy, "Gene Functional Classif cation from Heterogeneous Data," *Proceedings of 5th International Conference on Computational Biology, Montreal, Canada, April 22-25, 2001*, pp. 249–255.

[44] M. A. Ragan, "Phylogenetic Inference Based on Matrix Representation of Trees," *Molecular Phylogenetics and Evolution*, vol. 1, 1992, pp. 53–58.

[45] K. Robinson, D. Turgut, and M. Chatterjee, "Entropy based Clustering in Mobile Ad Hoc Networks," *Proceedings of 2006 IEEE International Conference on Networking, Sensing and Control (ICNSC), Ft. Lauderdale, FL, April 23-25, 2006*, pp. 1–5.

[46] R. Roiger and M. Geatz, *Data Mining: A Tutorial-Based Primer*, Addison Wesley, 2003.

[47] S. J. Russell and P. Norvig, *Artif cial Intelligence: A Modern Approach*, 2nd edition, Prentice Hall, Upper Saddle River, NJ, 2003.

[48] M. J. Sanderson, A. Purvis, and C. Henze, "Phylogenetic Supertrees: Assembling the Trees of Life," *Trends in Ecology and Evolution*, vol. 13, 1998, p. 105109.

[49] C. Semple and M. Steel, "A Supertree Method for Rooted Trees," *Discrete Applied Mathematics*, vol. 105, 2000, pp. 147–158.

[50] A. G. Skarmeta, A. Bensaid, and N. Tazi, "Data Mining for Text Categorization with Semi-Supervised Agglomerative Hierarchical Clustering," *International Journal of Intelligent Systems*, vol. 15, no. 7, 2000, pp. 633–646.

[51] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," *KDD-2000 Workshop on Text Mining, Boston, MA, August 20, 2000*.

[52] R. Steuer, J. Kurths, C. Daub, J. Weise, and J. Selbig, "The Mutual Information: Detecting and Evaluating Dependencies between Variables," *Bioinformatics*, vol. 18, 2002, pp. S231–S240.

[53] A. Strehl and J. Ghosh, "Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions," *Journal of Machine Learning Research*, vol. 3, 2002, pp. 583–617.

[54] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing Value Estimation Methods for DNA Microarrays," *Bioinformatics*, vol. 17, no. 6, 2001, p. 520525.

[55] M. Trutsch, T. D. Dinkova, and R. E. Rhoads, "Application of Machine Learning and Visualization of Heterogeneous Datasets to Uncover Relationships between Translation and Developmental Stage Expression of C. Elegans mRNAs," *Physioogical Genomics*, vol. 21, 2005, pp. 264–273.

[56] J. Verbeek, J. Nunnink, and N. Vlassis, "Accelerated EM-based Clustering of Large Datasets," *Data Mining and Knowledge Discovery*, vol. 13, no. 6, 2006, pp. 1–21.

[57] J. Vesanto and E. Alhoniemi, "Clustering of the Self-Organizing Map," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, 2000, pp. 586–600.

[58] K. Wagstaff, "Clustering with Missing Values: No Imputation Required," *Proceedings of the Meeting of the International Federation of Classif cation Societies, Chicago, IL, July 15-18, 2004*, pp. 649–658.

[59] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrdl, "Constrained K-means Clustering with Background Knowledge," *Proceedings of 18th International Conference on Machine Learning (ICML-2001), Williamstown, MA, June 28 - July 1, 2001*, pp. 577–584.

[60] Y. Wang, *Incorporating Semantic and Syntactic Information into Document Representation for Document Clustering*, doctoral dissertation, Department of Computer Science and Engineering, Mississippi State University, 2005.

[61] D. R. Wilson and T. R. Martinez, "Improved Heterogeneous Distance Functions," *Journal of Artif cial Intelligence Research*, vol. 6, no. 1, 1997, pp. 1–34.

[62] Z. Wu and R. Leahy, "An Optimal Graph Theoretic Approach to Data Clustering: Theory and its Application to Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, 1993, pp. 1101–1113.

[63] X. Xia and Z. Xie, "AMADA: Analysis of Microarray Data," *Bioinformatics*, vol. 17, no. 6, 2001, p. 569570.

[64] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance Metric Learning, with Application to Clustering with Side-Information," *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, eds., MIT Press, Cambridge, MA, 2003.

[65] Y. Xu, V. Olman, and D. Xu, "Minimum Spanning Trees for Gene Expression Data Clustering," *Genome Informatics*, vol. 12, 2001, pp. 24–33.

[66] Y. Zhao and G. Karypis, "Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering," *Machine Learning*, vol. 55, 2004, pp. 311–331.

[67] X. Zhou, X. Wang, E. R. Dougherty, D. Russ, and E. Suh, "Gene Clustering Based on Clusterwide Mutual Information," *Journal of Computational Biology*, vol. 11, no. 1, 2004, pp. 147–161.