THE UNIVERSITY OF
WARWICK

University of Warwick institutional repository: http://go.warwick.ac.uk/wrap

A Thesis Submitted for the Degree of PhD at the University of Warwick

http://go.warwick.ac.uk/wrap/3759

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.

THE UNIVERSITY OF
WARWICK

AUTHOR: **Tao Li** DEGREE: **Ph.D.**

TITLE: **Relational Clustering Models for Knowledge Discovery and Recommender Systems**

DATE OF DEPOSIT: .....

I agree that this thesis shall be available in accordance with the regulations governing the University of Warwick theses.

I agree that the summary of this thesis may be submitted for publication.

I **agree** that the thesis may be photocopied (single copies for study purposes only).

Theses with no restriction on photocopying will also be made available to the British Library for microfilming. The British Library may supply copies to individuals or libraries. subject to a statement from them that the copy is supplied for non-publishing purposes. All copies supplied by the British Library will carry the following statement:

> "Attention is drawn to the fact that the copyright of this thesis rests with its author. This copy of the thesis has been supplied on the condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's written consent."

AUTHOR'S SIGNATURE: .....

USER'S DECLARATION

1. I undertake not to quote or make use of any information from this thesis without making acknowledgement to the author.
2. I further undertake to allow no-one else to use this thesis while it is in my care.

DATE SIGNATURE ADDRESS

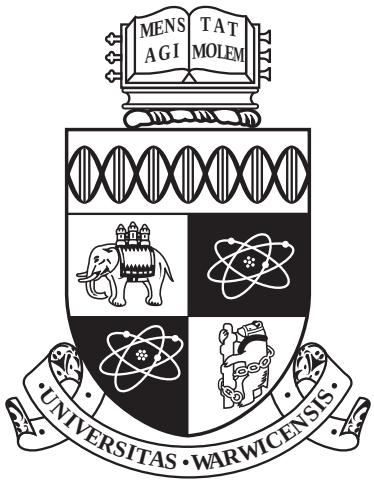
.....

.....

.....

.....

.....



Relational Clustering Models for Knowledge Discovery and Recommender Systems

by

Tao Li

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science, University of Warwick

April 2010

THE UNIVERSITY OF
WARWICK

Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgments

First of all, I would like to express my sincere gratitude to Dr. Sarabjot Singh Anand, my supervisor at the University of Warwick. He has supported me in all his endeavors and provided guidance throughout my PhD career. Besides, his insight and enthusiasm in scientific research encourage me all the time.

I would also thank my advisor, Dr. Nathan Griffiths, for his invaluable comments on my annual reports and PhD thesis. Thanks as well to Dr. Alípio M. Jorge and Dr. Sara Kalvala for their kind willingness to read and evaluate my dissertation.

As a member of the IAS group in the Department of Computer Science, I often received constructive suggestions and technical support from other departmental staff, in particular Dr. Alexandra Cristea, Dr. Mike Joy, Shanshan Yang, Jane Yau, Nick Landia, Henry Franks, Maurice Hendrix, Fawaz Ghali, Dr. Sarah Lim Choi Keung, Dr. Charles Care, Russell Boyatt, Dr. Zhan En (Eric) Chan, Shuangyan (Jenny) Liu, Dr. Christine Leigh, Dr. Roger Packwood, Mr. Paul Williamson, Mr. Richard Cunningham, Mr. Rod Moore, Ms. Catherine Pillet and Ms. Angie Cross. I would sincerely thank Prof. Stephen Jarvis, Dr. Meurig Beynon, Prof. Kershaw Baz and Dr. Cherie Wang for their supervision during my work as a teaching and research assistant.

My acknowledgement also goes to Prof. Hilary Marland, Ms. Ros Lucas, Dr. Joanne Allen, Dr. Joanne Anderson, Dr. Chris Boyce, Dr. Jing Kang, Dr. Lorraine Lim, Dr. Lydia Plath and Dr. Deborah Toner for the very pleasant atmosphere in the IAS semiars.

Last but by no means least, I would like to thank my best friends, Wee-Hoe Tan, Kai Zhang, Dr. Kai Yu, Dr. Tao Chen, Dr. Xiao-Ling Ou, Dr. Deng Cai, Dr. Jie

Liu, Dr. Ke Tang, Dr. Jie Bai, Dr. Zhi-Gang Zuo, Shuai Li, Wei-Dong Li, Jing-Long Liu, Bo Wang, Jing-Bo Sun, Yu-Jia Xiang and others for their affection, support and encouragement. Also I cannot forget my beloved parents for their unselfish love and devotion, helping me in any conceivable respect. Finally, I want to thank Xiao (Grace) Hu, for remind me that there are other important things in life besides machines and algorithms.

Financial support has been provided by the Graduate School, the Department of Computer Science and the Institute of Advanced Study in the University of Warwick, through the schemes of Warwick Postgraduate Research Fellowship, Departmental Scholarship and Early Career Fellowship.

Declarations

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. No part of the work contained in this thesis has been submitted for any degree or qualification at any other university.

Parts of this thesis were published previously in our papers: Chapter 2 extends a state-of-the-art review of the relational clustering that was published in [92]. The fundamental ideas of Chapters 3 and 4, including the recursive RO-selection algorithm as well as the DIVA clustering framework, were first proposed in [89]. Later, the incremental implementation of the DIVA framework was developed and reported in [88]. Chapters 6 and 7, which cover the techniques of automated taxonomy generation for relational data and the algorithm of labeling taxonomic nodes by utilizing Kullback-Liebler Divergence, were respectively published in [90] and [91]. Finally Chapter 9, i.e. the integration of automatically derived taxonomy into the recommender systems, was published in [93]. Two journal papers are currently under preparation, which include the detailed results of Part I and II respectively.

Abstract

Cluster analysis is a fundamental research field in Knowledge Discovery and Data Mining (KDD). It aims at partitioning a given dataset into some homogeneous clusters so as to reflect the natural hidden data structure. Various heuristic or statistical approaches have been developed for analyzing propositional datasets. Nevertheless, in relational clustering the existence of multi-type relationships will greatly degrade the performance of traditional clustering algorithms. This issue motivates us to find more effective algorithms to conduct the cluster analysis upon relational datasets. In this thesis we comprehensively study the idea of Representative Objects for approximating data distribution and then design a multi-phase clustering framework for analyzing relational datasets with high effectiveness and efficiency.

The second task considered in this thesis is to provide some better data models for people as well as machines to browse and navigate a dataset. The hierarchical taxonomy is widely used for this purpose. Compared with manually created taxonomies, automatically derived ones are more appealing because of their low creation/maintenance cost and high scalability. Up to now, the taxonomy generation techniques are mainly used to organize document corpus. We investigate the possibility of utilizing them upon relational datasets and then propose some algorithmic improvements. Another non-trivial problem is how to assign suitable labels for the taxonomic nodes so as to credibly summarize the content of each node. Unfortunately, this field has not been investigated sufficiently to the best of our knowledge, and so we attempt to fill the gap by proposing some novel approaches.

The final goal of our cluster analysis and taxonomy generation techniques is to improve the scalability of recommender systems that are developed to tackle the problem of information overload. Recent research in recommender systems integrates the exploitation of domain knowledge to improve the recommendation quality, which however reduces the scalability of the whole system at the same time. We address this issue by applying the automatically derived taxonomy to preserve the pair-wise similarities between items, and then modeling the user visits by another hierarchical structure. Experimental results show that the computational complexity of the recommendation procedure can be greatly reduced and thus the system scalability be improved.

Chapter 1

Preface

Cluster analysis is an important research field in Knowledge Discovery and Data Mining (KDD). The aim is to discover the intrinsic structure of the underlying dataset, which is generally referred to as the *unsupervised learning* problem. This is different from the *supervised learning* tasks (such as classification or regression) in which the data model is first constructed from the training dataset and then to be applied upon the test dataset. Many clustering algorithms have been proposed since the 1960s with the utilization of various ideas in such fields as statistics, combinatorial mathematics, artificial intelligence, spectral theory etc., but most of them are only suitable for analyzing propositional datasets. In practice relational datasets that contain different data types and relationships between them are more common. The existence of these multi-type relationships greatly degrades the performance of the traditional clustering algorithms if they are applied to the relational datasets naively. This issue motivates us to find more effective algorithms to conduct cluster analysis upon the relational datasets.

After the cluster result has been obtained, a hierarchical taxonomy can be generated that provides a better mechanism for people to browse and navigate the dataset. Automatically derived taxonomies are more appealing than manually derived ones because of their low creation/maintenance cost and high scalability. Generally speaking, taxonomy generation techniques are mainly used to organize a document corpus. We

investigate the possibility of utilizing them upon relational datasets as well as propose some algorithmic improvements. Another non-trivial problem is how to assign suitable labels for the taxonomic nodes so as to help people more easily understand the content of each node. Unfortunately, this field has not been investigated sufficiently to the best of our knowledge, and so we attempt to fill the gap by proposing some novel approaches.

To validate their applicability in practice, we utilize our cluster analysis and taxonomy generation techniques within recommender systems. Recent research in this field focuses on the incorporation of domain knowledge: meaningful neighbouring users are identified based on the similarity of their visiting items, so the quality of user-based recommendations will hopefully be improved as more relational domain information are exploited during the item similarity computation. But as the tradeoff, the system scalability is often decreased because exploiting such information needs more computational effort than that of traditional recommender systems computing based on propositional data. In this thesis, we propose to use an automatically derived taxonomy to preserve the pair-wise similarities between items in an offline stage. These item similarities are then retrieved directly from the taxonomy instead of obtained from the real-time calculation. By this way, the online computational expense of identifying neighbouring users can be reduced and the system scalability be improved effectively. In addition, the user visits are to be grouped within another hierarchical structure for further improving the system scalability.

Figure 1.1 explains the layers of our research framework and the thesis is organized accordingly. We study relational clustering algorithms in Part I, which covers the following chapters:

- Chapter 2 provides a comprehensive review of relational clustering, including some fundamental definitions and a categorization of different clustering algorithms with a brief discussion of key algorithms;

- Chapter 3 introduces the concept of Representative Objects, which constitute the foundation of our research in this thesis. In addition, we provide several typical

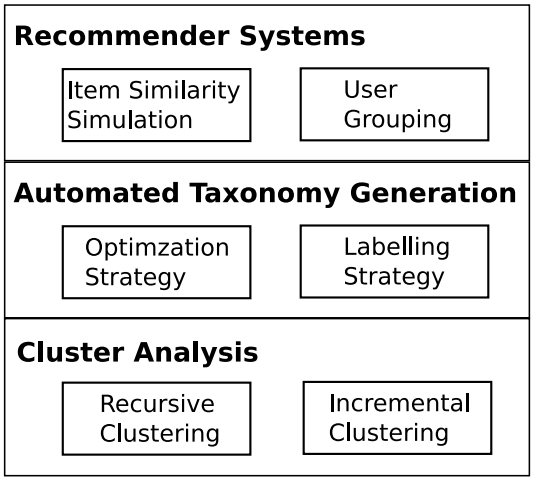


Figure 1.1: Our Research Framework

strategies of identifying Representative Objects in the dataset;

- Chapter 4 proposes a multi-phase clustering framework that is especially efficient in analyzing relational datasets. Two implementations of the framework are developed that are suitable for static and incremental learning tasks respectively.

Then in Part II we turn to the automated taxonomy generation and discuss the following topics:

- Chapter 5 reviews some important issues and related works in the field of automated taxonomy generation;
- Chapter 6 explains our taxonomy generation algorithm that is developed from the relational clustering framework;
- Chapter 7 discusses our approach of assigning labels for taxonomic nodes based on the idea of Kullback-Liebler Divergence.

After that, in Part III we will investigate the application of our data mining techniques in practice, i.e. to improve the scalability of recommender systems. After an overview of recommender systems in Chapter 8, Chapter 9 shows two possible applications of the

automatically derived taxonomy in the scenario of recommender systems: either preserve the pair-wise similarities between items or group the candidate users in order to accelerate the recommendation procedure. Finally, Chapter 10 summarizes our contributions and points out some future works.

Part I

Relational Clustering

Clustering is an important and active research field in Knowledge Discovery and Data Mining (KDD). By utilizing various heuristic or statistical approaches, the whole dataset is partitioned into a certain number of groups (clusters) so that data objects assigned into the same group share more common traits than those assigned into different groups [62]. In contrast to the supervised learning tasks such as classification, the categorical labels of the data are generally unknown beforehand in cluster analysis, so the target here is to discover the real or "natural" hidden structure within the data rather than providing an accurate prediction for the unobserved samples.

A large number of clustering algorithms have been proposed since the 1960s. They can be distinguished roughly as two categories: Partitional algorithms iteratively assign data objects into disjoint clusters and update the cluster features (e.g., means or k-prototypes) accordingly. On the contrary, hierarchical algorithms organize data using a hierarchical structure, in which upper-level clusters contain lower-level clusters. The hierarchy can be built either by treating each data object as separate clusters and then gradually merging them in a bottom-up fashion (known as Agglomerative Hierarchical Clustering), or by starting from a cluster containing all the data objects and recursively dividing the clusters in a top-down fashion (known as Divisive Hierarchical Clustering).

Traditional clustering algorithms are mainly used to analyze propositional dataset, in which data objects are of the same type and described using a fixed number of attributes. However, many datasets in practice contain different types of data objects and, more importantly, relationships between them. Naively applying propositional clustering algorithms cannot fully exploit information contained in these datasets or leads to inappropriate conclusions [64]. To address the challenge of analyzing heterogeneous datasets with interrelated data objects, some relational clustering algorithms have been developed in recent years, but their effectiveness and efficiency are not always satisfactory due to the impact of complexity relationship structure.

Our key contributions in the field of relational clustering are as follows:

1. We define the concept of Representative Objects and use them to effectively represent a cluster. Several typical strategies are developed to efficiently identify these representatives for the given clustering.

2. Using the representative objects as the cluster prototypes, we design a multi-phase clustering framework for analyzing relational datasets. Two implementations of the framework are also provided that are suitable for different learning tasks and both of them have linear complexity with respect to the data size.

3. We provided a state-of-the-art review in this field as the basis of our research, in which different relational clustering algorithms are systematically studied and compared with the propositional algorithms. We also evaluate our algorithms with a selection of these algorithms with respect to accuracy and efficiency.

This part of the thesis is organized as follows: Before presenting our research in relational clustering, we first provide the state-of-the-art review in Chapter 2. Then the idea of Representative Objects is introduced in Chapter 3 together with several strategies of identifying the representatives. Based on that, a novel relational clustering framework is explained in Chapter 4 with two implementations. Comprehensive experiments were conducted for evaluating the idea of Representative Objects as well as our clustering framework. The analysis of the experimental results are arranged in respective sections.

Chapter 2

Review of Relational Clustering

Clustering algorithms are generally developed based on the concept of proximity, i.e. similarity or distance. A set of clusters are constructed so that all data objects assigned into the same cluster are more similar to each other while data objects in different clusters are less similar. This criterion is also formulated as "maximizing intra-cluster similarity and inter-cluster distance", or "optimizing internal homogeneity and external separation" [61]. More formally, the clustering problem can be defined as follows [155]:

Definition 2.1. Given a dataset $\boldsymbol{D} = \{x_1, x_2, \ldots, x_N\}$, $x_i \in \boldsymbol{S}$ where $\boldsymbol{S}$ is the data space to be studied. A function $f(x_i, x_j)$ is defined in $\boldsymbol{S}$ to evaluate the proximity between x_i and x_j $(1 \leq i, j \leq N)$. The cluster analysis aims at finding the optimal partitioning $\boldsymbol{C} = \{C_k\}$ $(K < N$ and $1 \leq k \leq K)$ upon $\boldsymbol{D}$, given Functions $F_{intra}(C_k)$ and $F_{inter}(C_k, C_{k'})$ for evaluating the intra- and inter-cluster proximities respectively. The clusters $\{C_k\}$ are expected to satifsy the following properties :

1. $C_k \neq \emptyset$;
2. $\bigcup_{k=1}^{K} C_k = \boldsymbol{D}$;
3. There are two cases in flat partitional clustering: for hard partitioning, $C_k \bigcap C_{k'} = \emptyset$ when $k \neq k'$, while for soft partitioning, a data instance might belong to

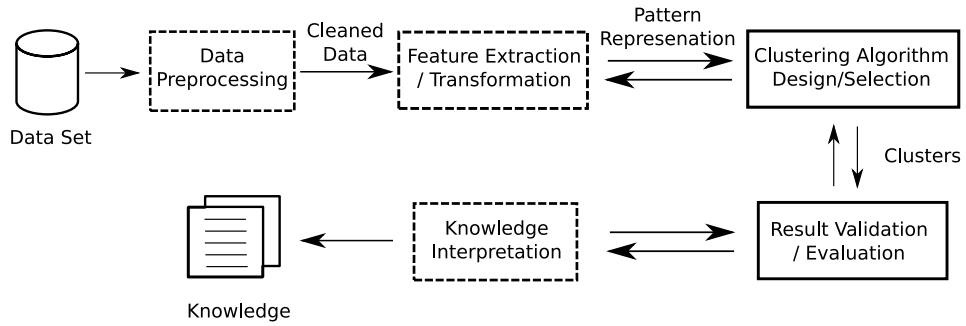


Figure 2.1: Five Steps of Cluster Analysis

more than one clusters. For hierarchical clustering we have either $C_k \subseteq C_{k'}$ or $C_k \bigcap C_{k'} = \emptyset$;

4. The sum of $F_{intra}(C_k)$ over k is maximized, while the sum of $F_{inter}(C_k, C_{k'})$ over k and k' is minimized.

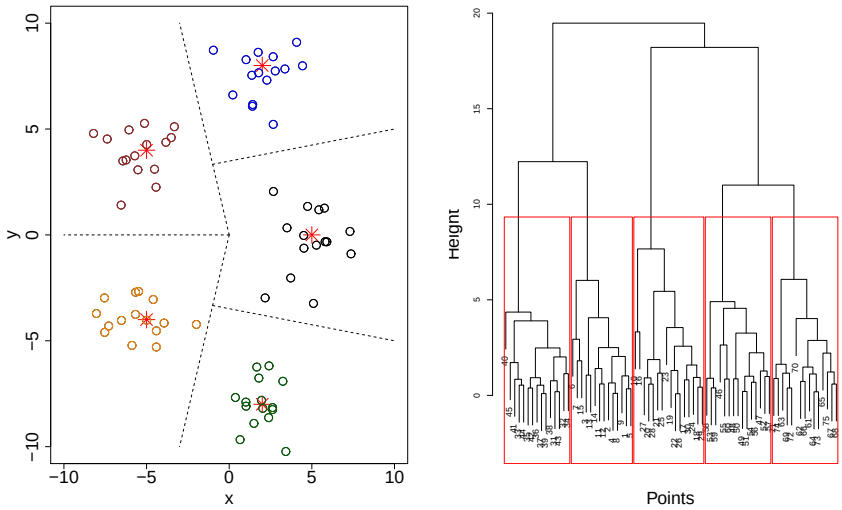
Figure 2.1 shows the different phases in the procedure of cluster analysis that are discussed in [62][155][55], including:

1. *Data Pre-processing* covers the operations of removing inconsistent data and noise data, integrating data from multiple sources, etc.;

2. *Feature Extraction and Transformation* means to extract existing features or construct new features that are most relevant to the analysis task;

3. *Algorithm Selection or Design* is the key step where intelligent methods are applied to discover patterns from the dataset. Since no clustering algorithms are universally suitable, to solve different problems in specific fields it is necessary to determine whether to adopt an existing algorithms or develop a new one;

4. *Result Validation and Evaluation* is to show the credibility of the derived cluster result. Also some criteria are utilized here to identify the truly useful patterns among the result;



(a) k-means (b) HAC

Figure 2.2: Comparison of Flat and Hierarchical Clustering

5. *Knowledge Interpretation* presents the learned useful patterns, i.e. the derived knowledge, in a user-friendly way.

The above five phases are often launched in a cyclic way. In many circumstances, a series of trials and repetitions are necessary to improve the final result. Additionally the phases (1), (2) and (5) are heavily dependent on the background knowledge. Usually the domain experts can easily point out which features are most relevant to the learning task and how to translate machine-understandable patterns into user-understandable ones. Hence in this part we mainly focus on the clustering algorithm itself, i.e. the phases (3) and (4) which are surrounded by the solid line in Figure 2.1.

A large number of clustering algorithms have been proposed since the 1960s. Comprehensive surveys in this field are provided in [62][14][155]. Roughly speaking, propositional algorithms can be categorized as *flat clustering* and *hierarchical clustering*:

Flat clustering algorithms divide the dataset into a number of disjoint clusters, assigning each data object to its nearest cluster and updating the cluster's geomet-

rical features (e.g. centroid or medoid[1] iteratively. Different algorithms differ in the similarity/distance functions, the strategies of grouping or splitting clusters and their choice of prototypes (such as mean, medoid, core/border points, etc). In this category the most well-known and widely applied algorithm is k-means, owing to its advantages of easy implementation and good scalability. Figure 2.2a shows the cluster result obtained by utilizing k-means upon a 2D dataset[2] with the Euclidean distance function (points labelled as $*$ in the figure are the centroids of the clusters). Other flat clustering algorithms are based on the assumption that all the data are generated by some mixture of underlying probability models (e.g., the Gaussian Mixture Model is commonly used), so the clustering problem is transformed into a parameter estimation problem where the parameters are determined by the underlying probability models. Then the standard Expectation-Maximization (EM) approach can be applied: in the E-step the cluster membership of all the data are identified and in the M-step the parameters of the probabilistic models are optimized. Generally model-based clustering algorithms are more mathematically understandable and reasonable than the proximity-based ones, but care must be taken when choosing the functional form of the underlying probability distributions.

Hierarchical clustering algorithms in contrast use a hierarchical structure to organize data, in which upper-level clusters contain the lower-level clusters. To generate the hierarchy, we can either treat each data object as a separate cluster and then gradually merge them in a bottom-up fashion, or start from a cluster containing all the data objects and recursively divide the clusters in a top-down fashion. In addition to calculating the similarities between data objects, these algorithms require the definition of similarity measure between clusters. The single-linkage and complete-linkage methods are often used for hierarchy generation. Figure 2.2b shows the hierarchical structure built by utilizing the hierarchical agglomerative clustering (HAC) algorithm upon the

[1] *Medoid* is the data object within a dataset of which the average distance to all the other objects in the dataset is minimal.

[2] This dataset contains five groups of normally distributed data points in a 2D space. The flat and hierarchical cluster results are generated by the functions *kmeans* and *hclust* in the GNU software R [40].

same 2D dataset as in Figure 2.2a. Then an appropriate level in the hierarchy may be selected as the cutting point to obtain a partitioning of the data (as the subtrees surrounded by the red boxes in the Figure 2.2b).

Besides the above categorization, the clustering algorithms may also be distinguished as incremental or non-incremental, depending on whether or not they are able to incrementally improve the cluster models when new data become available. Additionally, modern clustering algorithms can be categorized according to the techniques they adopt, such as graph theory, spectral theory etc.

Propositional clustering algorithms are mainly proposed for analyzing datasets in the multi-dimensional vector space, i.e. we have all $x_i \in \mathbb{R}^n$ in Definition 2.1. Based on this assumption, many algebraic or geometric approaches can be utilized to facilitate the calculation of the clustering procedure[3]. For example, BIRCH [162] adopted the "Clustering Feature" (CF) to summarize the properties of a cluster, which is essentially the linear sum and the squared sum of data vectors in the cluster. Other algebraic and geometric features of the cluster such as the centroid, radius/diameter, the intra-/inter-cluster distances can be computed from the CF vector. Moreover, the CF vector is easy to be updated when new data are absorbed into the cluster.

However, many datasets in practice cannot be represented as multi-dimensional numeric or nominal vectors. They are composed of heterogeneous data types and inter-relationships. We refer to them as the *relational datasets* to emphasize their substantial differences from the previous *propositional datasets*. Obviously, the algebraic and geometric theorems in the vector space $\mathbb{R}^n$ are not necessarily correct in the relational data space, for example the relational data instances are not additive or divisible as numeric vectors. Naively applying propositional clustering algorithms cannot fully exploit information contained in the dataset or leads to inappropriate conclusions about the data [64], so analyzing relational datasets has become a critical challenge and an active research direction in recent years. In the rest of this chapter, some fundamental concepts

[3]When processing a categorical dataset, we can use a different boolean dimension to represent each possible nominal value and thus translate categorical data into numerical data.

Table 2.1: Propositional Iris Dataset

ID (#)	Sepal Length	Sepal Width	Petal Length	Petal Width	Class
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
...					
51	7.0	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
...					
101	6.3	3.3	6.0	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3.0	5.9	2.1	Iris-virginica
...					

of relational clustering are introduced in Section 2.1 and then a state-of-the-art review of relational clustering algorithms is provided in Section 2.2.

2.1 Fundamentals of Relational Clustering

As follows, Section 2.1.1 explains the properties of relational data and the methodology of constructing relational data objects. Section 2.1.2 introduces a recursive similarity measure to compare relational data objects. Section 2.1.3 provides some criteria to evaluate the cluster quality.

2.1.1 Relational Data

Traditional data mining algorithms assume all data are represented as attribute-value pairs. With such assumption, each data instance corresponds to a row (or tuple) in a table and each attribute to a column [121]. In this paper, datasets that can be mapped into a single table are called *propositional*. Table 2.1 is an example of propositional dataset in the UCI Machine Learning Repository [7]. By analyzing these tuples, we can generate some propositional patterns, e.g. $X_{iris-setosa} = \{x|PetalWidth(x) \leq 0.6\}$

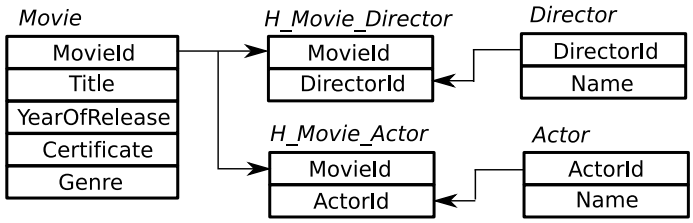


Figure 2.3: Schema of a movie dataset

or $X_{iris-virginica} = \{x|0.6 < PetalWidth(x) \leq 1.5 \text{ and } PetalLength(x) > 4.9\} \cup \{x|PetalWidth(x) > 1.7\}$.

In contrast, relational datasets pertain to domains with different data types and sets of relationships between them [36]. These data types together with the relationships constitute a far more sophisticated feature space than those in propositional datasets. According to the theory of relational database [28], a table stores a set of tuples with the same attributes and a link between tables means that the referencing tuple has the referenced tuple as part of its attributes, so we have:

Definition 2.2. A *relational dataset* contains a set of tables $\boldsymbol{D} = \{X_1, X_2, \ldots, X_M\}$ and a set of relations (links, associations) between pairs of tables. All the tuples in the table X_i are of the same type and tuples in different tables X_i and X_j are semantically associated by the relation(s) between X_i and X_j. To simplify our discussion, we use $X_i.c$ to denote the concept c of table X_i, where c might be an attribute in X_i or the name of another table that is associated to X_i.

Consider an example of a relational movie dataset, of which the schema is shown in Figure 2.3. Each table contains a set of attributes to define an entity type (e.g. *Movie*, *Actor* or *Director*) or a many-to-many relationship (e.g. director-*Direct*-movie or actor-*Act*-movie). Links between tables indicate a one-to-many relationship (along the direction of arrows). Based on this database schema, the original data instances are decomposed before being stored in order to minimize the data redundancy. In the retrieval procedure all the related tuples in different tables are extracted and combined to recover the original instance. For example, with the aid of relational algebra operations

such as *join*, we can easily retrieve all the information about an actor: his name and all the movies he acted in as well as all the directors he worked with.

Because tuples in different tables of a relational dataset are semantically associated by the relationship(s) between these tables, *Relational Patterns* extends the Propositional Patterns in the scenario of relational learning [76][121]:

Definition 2.3. *Relational learning* is "the study of machine learning and data mining within expressive knowledge representation formalisms encompassing relational or first-order logic. It specifically targets learning problems involving multiple entities and relationships amongst them."

From the above definition, we can see that attribute values of tuples as well as their pairwise linkage information are both important for relational learning. When searching for patterns in a relational dataset, not only the tuples in the target table but also those in the associated tables should be considered. Therefore, in the movie dataset (Figure 2.3) the analysis of any actor should be based on the movies he has acted in as well as the directors or other actors he cooperated with, so information stored in the associated tables *Movie* and *Director* needs to be exploited. For example, a relational pattern to strictly identify action movie stars can be described as $\tilde{X}_{action} = \{x \mid x \in X_{Actor}, (x.\text{movie}).\text{genre} = Action\}$.

Sometimes propositional clustering algorithms are still applicable for processing relational datasets by means of merging multiple tables into one (this operation is called "propositionalization" [78]) , but it is not a good choice because [47][100]:

- Transforming relational linkage information as additional attributes in the target table often leads to a very high dimensional feature space. The derived tuples are very sparse and thus inevitably degrades the performance of clustering algorithms.

- The linkage structure itself has structural properties (e.g. degree or connectivity) that can provide important information for the cluster analysis, but such features might be lost during the procedure of transformation.

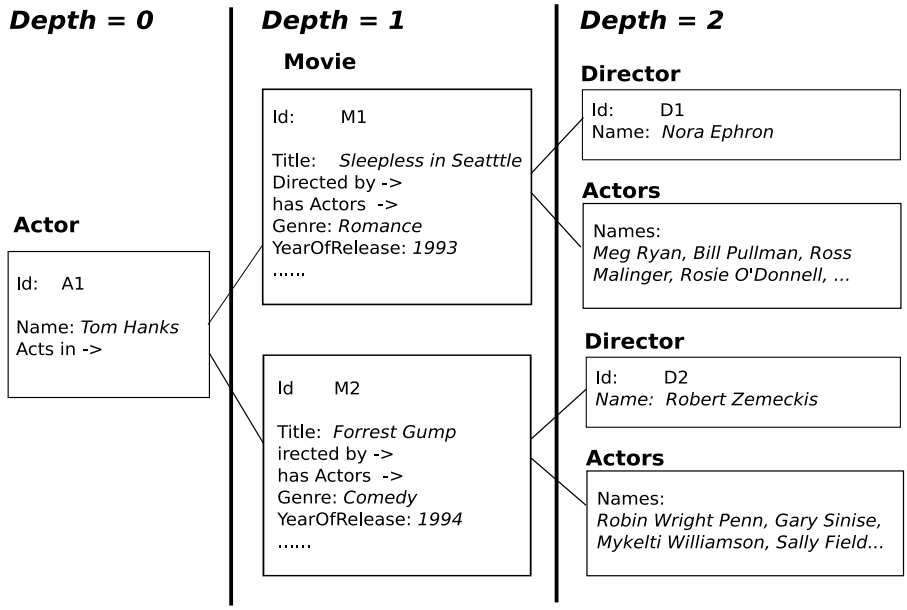


Figure 2.4: Example relational object for Tom Hanks

- The propositional clustering algorithms are originally designed for propositional datasets, and so they cannot capture the features that are propagated along the paths of relationships between multiple data types.

The above limitations of applying propositional clustering algorithms to relational datasets motivate the development of numerous relational clustering algorithms. As the basis of further discussion, here we will first introduce an approach of constructing relational data objects to systematically exploit the associated information for a target tuple.

Given a dataset schema represented as a directed graph $G = (V, E)$, in which vertices $V = \{c_i\}$ stand for the tables (concepts) in the schema and edges $E = \{\vec{e}_{st} | \text{ edge } \vec{e}_{st} : c_s \rightarrow c_t;\ c_s, c_t \in V\}$ for the relationships between concepts. An edge $\vec{e}_{st} : c_s \rightarrow c_t$ means that the source concept c_s references the target concept c_t as its associated feature. When constructing an object x of concept c_s, we find out all the associated features $\mathcal{F}(c_s)$ and link any object y of type c_t that satisfies the relation between c_s and c_t as an associated attribute value of $x.c_t$. Then for each $y \in x.c_t$, the above procedure is iteratively performed until $\mathcal{F}(c_t) = \emptyset$ or a depth bound is reached.

Table 2.2: Properties of distance functions

	Distance Function
Positivity	$fd(x_i, x_j) \geq 0$
Reflexivity	$fd(x_i, x_j) = 0$, iff $x_i = x_j$
Symmetry	$fd(x_i, x_j) = fd(x_j, x_i)$
Triangle Inequality*	$fd(x_i, x_k) \leq fd(x_i, x_j) + fd(x_j, x_k)$

In the example of the data schema shown in Figure 2.3, according to the relationship structure, we have $\mathcal{F}(\textit{Movie}) = \{\textit{Title}, \textit{Genre}, \textit{YearOfRelease}, \textit{Duration}, \textit{Actor}, \textit{Director}\}$ and $\mathcal{F}(\textit{Actor}) = \mathcal{F}(\textit{Director}) = \{\textit{Name}, \textit{Movie}\}$. Then all the associated data objects are recursively linked to the current object. Figure 2.4 shows parts of the relational data object constructed for Tom Hanks.

2.1.2 Proximity Measures

Many clustering algorithms as well as evaluation criteria are designed based on the proximity (i.e. similarity or distance) values between pairs of data objects. Before discussing the proximity measures for relational datasets, we first go through some classic measures that are widely used in propositional datasets.

For any propositional dataset $\boldsymbol{D} = \{x_1, x_2, \ldots, x_N\}$, we can either define a similarity measure $fs(\cdot, \cdot)$ or a distance measure $fd(\cdot, \cdot)$. These measures have some properties listed in Table 2.2 [35]. The property of triangle inequality is not necessary for a proximity measure. When it is satisfied, the measure is also called a *metric*. It is also worth noting that the distance measure $fd(\cdot, \cdot)$ is generally unbound. In most cases we can normalize the distance measure to limit its values within the interval $[0, 1]$. Then the normalized distance measure can be converted into a similarity measure or vice verse by utilizing the property $fs(x_i, x_j) + fd(x_i, x_j) = 1$.

The attributes of propositional data might be of types *nominal*, *numeric*, *string-based*, *vector-based*, etc. A variety of proximity measures are applicable for different data types to compute similarity or distance between data:

Table 2.3: Similarity Measure for Categorical dataset

α	β	Name of Similarity Measure
0	1	Jaccard Coefficient
	2	Sokal and Sneath Measure
	$1/2$	Gower and Legendre Measure
1	1	Simple Matching Coefficient
	2	Rogers and Tanimoto Measure
	$1/2$	Gower and Legendre Measure

- For nominal datasets, when x_i and x_j denote single nominal values, a straightforward similarity function can be defined as:

$$fs_{\text{nominal}}(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

  In some cases, we expect to know the similarity between two sets of nominal values $\hat{x}_i$ and $\hat{x}_j$. If we use n_{11} to represent the number of nominal values appearing in both $\hat{x}_i$ and $\hat{x}_j$, n_{00} for nominal values appearing in neither $\hat{x}_i$ or $\hat{x}_j$, n_{10} (resp. n_{01}) for nominal values that appear in $\hat{x}_i$ (resp. $\hat{x}_j$) only, a family of proximity measures can be defined by [35]:

$$fs'_{\text{nominal}}(\hat{x}_i, \hat{x}_j) = \frac{n_{11} + \alpha \cdot n_{00}}{n_{11} + \alpha \cdot n_{00} + \beta \cdot (n_{01} + n_{10})} \tag{2.2}$$

  Table 2.3 gives some possible combinations for parameters α and β, as well as their corresponding measures.

- For numeric variables, x_i and x_j represent single values in the range (a, b) where $b > a$. Then the distance can be calculated by:

$$fd_{\text{numeric}}(x_i, x_j) = \frac{x_i - x_j}{b - a} \tag{2.3}$$

- For two strings, *Levenshtein Distance* [87] counts the minimum number of oper-

ations needed to transform one string into the other. Transformation operators include insertion, deletion, or substitution of a single character. Then the normalized distance function can be defined by:

$$fd_{\text{string}}(x_i, x_j) = \frac{levenshtein\_distance(x_i, x_j)}{string\_length(x_i) + string\_length(x_j)} \tag{2.4}$$

- For two points in the n-dimension Euclidean space, *Minkowski metric* is widely used:

$$fd_{\text{point}}(x_i, x_j) = \left(\sum_{p=1}^{m} | x_{ip} - x_{jp} |^{q} \right)^{1/q}, \quad q \geq 1 \tag{2.5}$$

  Setting $q = 1$ gives the *Manhattan metric* while $q = 2$ is the familiar *Euclidean metric*. When $q \to \infty$, the metric converges to $\max_p |x_{ip} - x_{jp}|$.

- When x_i and x_j are numeric vectors, the cosine value of the angle between them can be used as the similarity measure:

$$fs_{\text{vector}}(\vec{x}_i, \vec{x}_j) = \frac{\sum_p x_{ip} \cdot x_{jp}}{\sqrt{\sum_p x_{ip}^2} \cdot \sqrt{\sum_p x_{jp}^2}} \tag{2.6}$$

  Such a model has been applied successfully in the fields of Information Retrieval(IR) and Collaborative Filtering(CF). Each index term of the document or each user of the object is taken to be a orthogonal dimension in the vector space, and so documents or objects are represented as weight vectors. In the IR domain these weights can be obtained by TF-IDF ranking strategy [9], and in the CF domain they are the ratings provided by users.

- In statistics, *Correlation Coefficient* refers to the departure of two random variables x and y from independence: $\rho = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$. If $\rho = \pm 1$, it means x and y are maximally positively or maximally negatively correlated. If $\rho = 0$, then x and y are completely uncorrelated. Therefore, the absolute value of the correlation between two variables can be adopted to measure their similarity. Many correlation mea-

surements have been developed, among which *Pearson's Correlation Coefficient* is the best known:

$$fs_{\text{stat}}(x_i, x_j) = \frac{\sum_p (x_{ip} - \overline{x}_i)(x_{jp} - \overline{x}_j)}{\sqrt{\sum_p (x_{ip} - \overline{x}_i)^2} \cdot \sqrt{\sum_p (x_{jp} - \overline{x}_j)^2}} \tag{2.7}$$

where $\overline{x}_i = \frac{1}{m}\sum_{p=1}^{m} x_{ip}$ and $\overline{x}_j = \frac{1}{m}\sum_{p=1}^{m} x_{jp}$.

The propositional proximity measures introduced above are the foundation for constructing the relational measures. Given two relational objects associated with multiple tables of different data types, we can first utilize some appropriate propositional measures to compare their components and then synthesize the results as the total proximity value for the objects. Horváth et al. proposed the RIBL2 measure in [60] : Given two relational data objects x_i and x_j of concept c_s constructed as in Section 2.1.1, their relational similarity is computed as:

$$fs_{\text{object}}(x_i, x_j) = \sum_{c_t \in \mathcal{F}(c_s)} w_{st} \cdot fs_{\text{set}}(x_i.c_t, x_j.c_t) \tag{2.8}$$

where weight w_{st} $(w_{st} \leq 1 \text{ and } \sum_t w_{st} = 1)$ represent the importance of member concept c_t when describing the concept c_s. In Equation 2.8, $fs_{set}(x_i.c_t, x_j.c_t)$ is defined as:

$$fs'_{\text{set}}(x_i.c_t, x_j.c_t) = \begin{cases} \frac{1}{|x_i.c_t|} \sum_{y_l \in x_j.c_t} \max_{y_k \in x_i.c_t} fs_{\text{object}}(y_k, y_l), & \text{if } |x_i.c_t| \geq |x_j.c_t| > 0. \\ \frac{1}{|x_j.c_t|} \sum_{y_k \in x_i.c_t} \max_{y_l \in x_j.c_t} fs_{\text{object}}(y_k, y_l), & \text{if } |x_j.c_t| \geq |x_i.c_t| > 0. \\ 0, & \text{if } |x_i.c_t| = 0 \text{ or } |x_j.c_t| = 0. \end{cases} \tag{2.9}$$

If $\mathcal{F}(c_j) \neq \emptyset$, the value of $fs(y_k, y_l)$ in Equation 2.9 is recursively calculated by Equation 2.8. This measure explores the linkage structure of the relational objects in a recursive fashion. The procedure continues until $\mathcal{F}(c_j) = \emptyset$ or the depth bound is

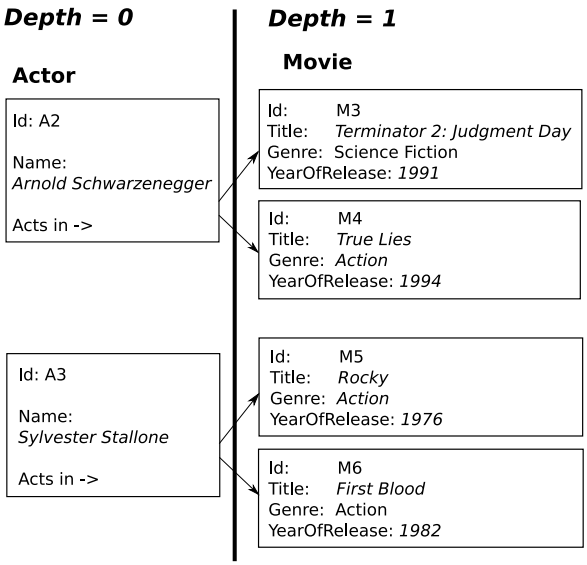


Figure 2.5: Two Example Movie Stars

reached, where the propositional similarity metrics can be applied. Generally, the whole linkage structure is not necessary to be exploited because the weights w_{st} between the referencing concept c_s and the referenced concept c_t make the impact of components in low levels of linkage structure decreases exponentially.

Here is an example to demonstrate the above recursive similarity measure: in the movie dataset we have $\mathcal{F}(\textit{Movie}) = \{\textit{Title}, \textit{Genre}, \textit{YearOfRelease}, \textit{Actor}, \textit{Director}\}$ and $\mathcal{F}(\textit{Actor}) = \{\textit{Name}, \textit{Movie}\}$. Figure 2.5 shows the relational objects for action movie stars "Arnold Schwarzenegger" and "Sylvester Stallone". We compare them with the actor "Tom Hanks" represented in Figure 2.4. In the scenario of propositional clustering, actors are compared based on their names since no relational data are considered there. If we regard the name of actors as a set of enumerated values and then utilize the true-or-false function, the similarity between every pair of actors will always be zero because they have different names. An alternative way is to use the Levenshtein distance to compare the actors, then we have $fs(o_{A2}, o_{A3}) = fs_{\text{string}}(\text{"Arnold Schwarzenegger"}, \text{"Sylvester Stallone"}) = 0.095$ and $fs(o_{A1}, o_{A3}) = fs_{\text{string}}(\text{"Tom Hanks"}, \text{"Sylvester Stallone"}) = 0.111$, which is still unreasonable.

Table 2.4: Linkage Matrix between Actors and Movies

Name	M1	M2	M3	M4	M5	M6
Tom Hanks ($A1$)	1	1	0	0	0	0
Arnold Schwarzenegger ($A2$)	0	0	1	1	0	0
Sylvester Stallone ($A3$)	0	0	0	0	1	1

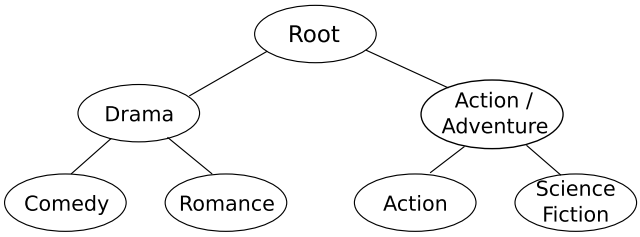


Figure 2.6: Genre Taxonomy

If we transform the relational information into a high dimensional vector space, e.g. constructing a binary vector to represent the movies that an actor has acted in, the pairwise similarity between the actors can be calculated using the cosine value of their movie vectors or simply using the equivalent Jaccard Coefficient. As shown in Table 2.4, such a transformation will produce sparse data and hence bias the cluster analysis. Another issue is that the semantic information about movie genres shown in Figure 2.6 will be lost when calculating pairwise similarity between actors.

To illustrate the RIBL2 measure, we first calculate the similarity value between movies "Terminator 2: Judgment Day" and "Rocky" (to simplify our calculation, the weights w_{st} for different associated concepts are set equally):

$$
\begin{aligned}
& fs_{\text{object}}(o_{M3}, o_{M5}) \\
&= \frac{1}{3}\Big[fs_{\text{string}}(o_{M3\cdot Title}, o_{M5\cdot Title}) + fs_{\text{taxonomy}}(o_{M3\cdot Genre}, o_{M5\cdot Genre}) \\
&\quad + fs_{\text{numeric}}(o_{M3\cdot Year}, o_{M5\cdot Year})\Big] \\
&= \frac{1}{3}\left[0.077 + 0.5 + \left(1 - \frac{|1991 - 1976|}{year_{max} - year_{min}}\right)\right] \\
&= 0.359
\end{aligned}
$$

Similarly we have:

$$
\begin{aligned}
fs_{\text{object}}(o_{M4}, o_{M5}) &= \frac{1}{3}\left[0 + 1 + \left(1 - \frac{|1994 - 1976|}{30}\right)\right] = 0.467 \\
fs_{\text{object}}(o_{M3}, o_{M6}) &= \frac{1}{3}\left[0.154 + 0.5 + \left(1 - \frac{|1991 - 1982|}{30}\right)\right] = 0.451 \\
fs_{\text{object}}(o_{M4}, o_{M6}) &= \frac{1}{3}\left[0.182 + 1 + \left(1 - \frac{|1994 - 1982|}{30}\right)\right] = 0.594
\end{aligned}
$$

Based on the above similarity values between movies, we can calculate the similarities between actors as follows:

$$
\begin{aligned}
& fs_{\text{object}}(o_{A2}, o_{A3}) \\
&= \frac{1}{2}\Big[fs_{\text{string}}(o_{A2 \cdot Name}, o_{A3 \cdot Name}) + fs_{\text{set}}(o_{A2 \cdot Movie}, o_{A3 \cdot Movie})\Big] \\
&= \frac{1}{2}\left[0.095 + \frac{1}{2}\sum_{y_l \in \{\text{M5, M6}\}} \max_{y_k \in \{\text{M3, M4}\}} fs_{\text{object}}(y_k, y_l)\right] \\
&= \frac{1}{2}\left[0.095 + \frac{1}{2}(0.467 + 0.594)\right] = 0.313
\end{aligned}
$$

In the same way, we can get $fs_{\text{object}}(o_{A1}, o_{A3}) = 0.171 < fs_{\text{object}}(o_{A2}, o_{A3})$, which means Sylvester Stallone ($A3$) is more similar to Arnold Schwarzenegger ($A2$) than Tom Hanks ($A1$). Therefore, by utilizing the relational proximity measure to exploit the linkage structure within the relational dataset, we achieve more credible results about the object comparison.

The idea of mutual reinforcement has also been utilized recently to iteratively calculate similarity values between relational data objects. Jeh and Widom [63] studied the structural-context similarity of relational data objects. Their similarity measure, named *SimRank*, is based on the principle "two objects are similar if they are related to similar objects". Given the original relational objects represented as a usual graph G, they construct an ordered node-pair graph G^2 in which each node represents a pair of objects in G and each edge represents the transit between two pairs of objects. Then

an iterative fixed-point algorithm is used to compute the SimRank scores for the node-pairs in G^2. The authors proved in theory that the solution for the SimRank equation is existent and unique. They also showed that their SimRank model is mathematically equivalent to the expected-f meeting distance in a random surfer-pairs model. Fogaras and Rácz [44] improved the scalability of the original SimRank approach by introducing the idea of randomized Monte Carlo methods combined with indexing techniques. Xi et al. [154] proposed a Unified Relationship Matrix (URM) to represent a set of heterogeneous data objects as well as their interrelationships. Based on that, they developed a unified similarity-calculating algorithm, *SimFusion*, by iteratively computing over the URM until reaching the convergence. The theoretical analysis showed that the URM can be considered as a single step transition matrix of a Markov Chain and the iterative similarity reinforcement process of updating the Unified Similarity Matrix is equivalent to a "two random walker model". Although the effectiveness of SimRank and SimFusion are supported by some experimental results, their computational complexities are quadratic with respect to the number of data objects. Xue et al. [156] developed a Multiple Relationship Similarity Spreading Algorithm (MRSSA) to compute similarities for multiple object types in an iterative spreading fashion. They first use a graph-based model to represent the linkage structure within a relational dataset and define the relational similarity measure as a linear combination of the content similarity as well as the intra- and inter-type similarities between data objects. Because the intra- and the inter-type similarity mutually affect each other, an iterative process is used to improve the precision of pairwise similarities between data objects. White and Smyth [148] define and evaluate a host of metrics to compute the similarity between a given object and the other reference objects in a graph. The above reviewed measures do not always satisfy the property of Triangle Inequality in Table 2.2, which means they are not metrics. To address this issue, some relational distance metrics like [123] or [122] have been proposed.

2.1.3 Evaluation Criteria

In this section we consider another important question: *If the proximity measure has been chosen, which criterion should be used to guide the cluster analysis and evaluate the derived results*? Clustering can be regarded as a search problem with each node corresponding to a certain partitioning [112], but exhaustively evaluating all possible partitions to find the optimal one is infeasible in practice due to the high computational complexity. Therefore some heuristic methods are utilized in the design of clustering algorithms to accelerate the search. It has been pointed out that the procedure of cluster analysis is considerably subjective in nature: the target data objects are partitioned into "a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity (i.e., chosen subjectively based on its ability to create interesting clusters)" [8]. Hence, the evaluation criterion plays an important role for the cluster analysis. It guides the search direction in the partitioning space as well as quantitatively evaluating different partitions derived by clustering algorithms to find the optimal one.

Some criteria for propositional datasets are outlined in [35]. Among which *Sum of Squared Error* (SSE) is the most widely used for clustering. Let K be the number of derived clusters, N_k be the number of data instances in cluster C_k and $\overline{x}_k$ be the center of these data. The SSE criterion is defined as:

$$E = \sum_{k=1}^{K} \sum_{x \in C_k} fd^2(x, \overline{x}_k) \tag{2.10}$$

The SSE criterion is appropriate when the clusters are compact and well separated from each other. However, when the number of data objects in the optimal partition vary greatly, the cluster result with minimum SSE may not reveal the true underlying data structure, because a partition splitting large clusters is more favorable under such circumstance. It means those clustering algorithms designed based on the principle of minimizing the SSE criterion, e.g. k-means, tend to generate the clusters of the equal

size. To address this issue, another criterion, *Related Minimum Variance*, might be used:

$$E = \frac{1}{2}\sum_{k=1}^{K} N_k \overline{s}_k \tag{2.11}$$

where

$$\overline{s}_k = \frac{1}{N_k^2}\sum_{x\in C_k}\sum_{x'\in C_k} fs(x,x') \quad \text{or} \quad \overline{s}_k = \min_{x,x\in C_k} fs(x,x')$$

The SSE or Related Minimum Variance criterion can be applied in the functions $F_{intra}(C_k)$ and $F_{inter}(C_k, C_{k'})$ of Definition 2.1 to guide the clustering procedure or evaluate the derived result. For example, in k-means we iteratively adjust the membership of data instances in each cluster in order to reduce the SSE value over all the clusters.

The SSE criterion, also named as the *Within-cluster Scatter Matrix* (denoted as S_W) in multiple discriminant analysis, can be used to evaluate the intra-cluster distances. Additionally, the criteria of *Between-Cluster Scatter Matrix* and *Total Scatter Matrix* (denoted as S_B and S_T respectively) are used to evaluate the inter-cluster distances and the scattering extent of the whole dataset respectively:

$$S_B = \sum_{k=1}^{K} N_k \cdot fd^2(\overline{x}_k, \overline{x}) \quad \text{and} \quad S_T = \sum_{\overline{x}\in D} fd^2(x, \overline{x}) \tag{2.12}$$

where $\overline{x}_k$ is the center for all data objects in cluster C_k as before, $\overline{x}$ is the center for the whole dataset. In propositional clustering, the center of a cluster is the mean of all data in that cluster: $\overline{x}_k = \frac{1}{N_k}\sum_{x\in C_k} x$ when cluster $C_k \subset \mathbb{R}^m$. Similarly we have $\overline{x} = \frac{1}{N}\sum_{x\in \boldsymbol{D}} x$. However, in relational clustering the center of a cluster is usually determined by the medoid of all data objects in that cluster, so the constraint $S_T = S_W + S_B$ that is valid in propositional datasets will be invalid for relational clustering. Additionally, the operation of determining the medoid within a cluster has quadratic computational complexity. Such disadvantage heavily restricts the application of relational clustering algorithms that are designed to minimize the SSE criterion of a partition for a relational dataset. This issue will be investigated thoroughly in the Section 2.2.1.

To evaluate the quality of the cluster result, it is possible to examine the homogeneity within the clusters and the heterogeneity between the clusters when the class labels of data objects are available. The Jaccard Coefficient [141] is suitable for this purpose, which is computed by the number of pairs of objects in same cluster and with same class label over that of pairs of objects either in same cluster or with same class label. In Table 2.3 we set the parameters $b = 0$ and $w = 1$ to get the Jaccard Coefficient.

Alternatively, we can evaluate the quality of clusters based on the idea of entropy. Entropy was first introduced in thermodynamics to measure the system's thermal energy. Being obtained from the disordered molecular motion, entropy reflects the molecular disorder in the thermodynamic system [51]. Later, entropy was extended to measure the uncertainty associated with a random variable in information theory [133]. Recently the entropy is used to evaluate the disorder or impurity of clusters [145]. Formally, given the class labels of data objects in a cluster C_k, C_k's entropy is computed by:

$$E(C_k) = -\sum_{h} P_{h,k} \log_2 P_{h,k} \tag{2.13}$$

where $P_{h,k}$ is the proportion of data objects of class h in the cluster C_k. The total entropy is defined as:

$$E = \sum_{C_k} E(C_k) \tag{2.14}$$

Generally speaking, smaller entropy values indicate higher accuracy of cluster result.

2.1.4 Determining the Number of Clusters

In this section we will briefly discuss another issue in clustering: how to determine an appropriate number of clusters that should be generated? Obviously the quality of final cluster result is heavily dependent on such number: too many derived clusters will break linkages between similar data instances and thus cause the loss of pattern information; while too few derived clusters will hide the pattern within noise and make the result difficult to interpret. Unfortunately, many clustering algorithms ask the number of

clusters as an input parameter and in most cases such number has to be estimated based on background knowledge or exclusive analysis on the dataset itself. So determining the number of clusters is considered as "the fundamental problem in cluster validity" [34].

Xu and Wunsch summarized four approaches in [155] to estimate the appropriate number of clusters for propositional clustering algorithms:

- Visualizing datasets: When the dataset can be mapped to points in a 2D or 3D space, we will use some graphical tools to visualize the distribution of data instances and thus estimate the optimal number of clusters. However, the applicability of this approach is restricted because of the complexity data structure in most real dataset.

- Constructing certain indices (stopping rules): These indices are generally based on evaluating the compactness of derived clusters, e.g. the squared error, the intra-cluster similarity and inter-cluster distance or some synthesized criteria, to determine the optimal value of K. One disadvantage is that the success of these indices is data dependent, which means the good performance of an index upon one dataset does not guarantee it works well on other datasets [41].

- Optimizing some criterion function under probabilistic mixture-model framework; The clustering problem can also be solved using EM algorithm, i.e. using data to estimate parameters of probabilistic models for a given range of K and then select the optimal K that maximize some predefined criterion. Many criteria based on various ideas in statistics, Bayesian theory or information theory have been proposed [150][117][115].

- Utilizing other heuristic approaches: For example, eigenvalue decomposition on the kernel matrix in the feature space has been investigated to determine K [49].

Ideally we expect the clustering algorithms can automatically adjust the number of derived clusters rather than use a pre-specified and fixed parameter. In propositional

clustering the above goal has been achieved more or less by utilizing the above approaches. However, in relational clustering the problem is more sophisticated because cluster results of different data types are inter-related, which means skewed clusters of one type caused by an inappropriate K will propagate along the linkage structure to influence partition of other data types and thus deteriorate the global cluster results. To the best of our knowledge, there are no systematic studies on how to determine the optimal number of clusters in relational cluster analysis.

2.2 Relational Clustering Algorithms

Relational clustering algorithms can be categorized as proximity-based, reinforcement, model-based, graph theoretic and constraint-based approaches. Among them the reinforcement clustering is the only one not rooted in propositional clustering approaches. We now discuss each of these categories in turn.

2.2.1 Proximity-based Clustering

As discussed in the previous section, proximity-based approaches can be distinguished as flat clustering and hierarchical clustering.

Flat clustering approaches assign data into a set of disjoint clusters without hierarchical relationship between each other. Since finding the global optima is usually infeasible in practice, some heuristic techniques are incorporated to accelerate the procedure. Based on the proximity measures introduced in Section 2.1.2, the clustering procedure begins at a random partition and then approximates the optimal solution by minimizing a specific criterion (Section 2.1.3) in each iteration. When the error falls under an acceptable threshold or the assignment of data to the clusters no longer change from one iteration to another, which is regarded as "convergence", the partition will be output as the final result. The most well-known partitional clustering algorithms are k-means and k-medoids. More sophisticated ideas are introduced in recent years to amend the cluster quality (e.g. PAM, CLARA+ [70] and CLARANS [112]), to achieve

global search and faster convergence (e.g. genetic k-means algorithm (GKA) [79] and global k-means algorithm [95]), to find arbitrary cluster shapes (e.g. DBSCAN [39] and OPTICS [5]), etc.

Hierarchical clustering approaches organize all data using a hierarchical structure, called a "dendrogram". The root node represents the whole dataset and each leaf node is regarded as a data object. Each intermediate node in the dendrogram corresponds to a subset of the dataset, in which the data are grouped according to their pairwise proximity. The ultimate cluster result is obtained by cutting the dendrogram at an appropriate level. Hierarchical clustering approaches can be performed divisively or agglomeratively. The former methods start by assuming the entire dataset belongs to the same cluster and successively divides it into sub-clusters. The agglomerative methods begin with each data object in a distinct (singleton) cluster and successively merges them together. Various linkage criteria, for example single-linkage, complete linkage, average linkage, median/centroid linkage or Ward's method [155], can be used to compute the pairwise proximity between clusters. The common disadvantages of classic hierarchical clustering approaches include the high computational complexity and the sensitivity to outlier data. Hence, new algorithms such as BIRCH [162], CURE [52] and Chameleon [68] were developed to address these issues.

Since the flat and the hierarchical clustering approaches are both based on proximity measures, they might be extended to process relational datasets by incorporating the relational proximity measure introduced in Section 2.1.2. Kirten and Wrobel developed RDBC [72] and FORC [71] as the extensions of propositional hierarchical agglomerative and k-partitional algorithms respectively. Both of them adopt the measure RIBL2 [73] to evaluate the distance between data objects. However, since objects in the relational dataset are normally not additive or divisible as numeric vectors, RDBC and FORC have to use the medoids to represent the center of clusters instead of using the means. The medoid of a cluster is defined as the data object that has the maximum average similarity (or minimum average of distance) with the other objects in that clus-

ter. To achieve this, all pairs of data objects in the given cluster have to be compared. Therefore, RDBC and FORC are not suitable for clustering very large datasets as they have quadratic time complexity.

2.2.2 Reinforcement Clustering

The principal of reinforcement clustering approaches comes from the observation that, since all data objects of different types are inter-related to each other, the cluster result of one data type might be propagated along the relationship structure to improve that of other data types. Anthony and desJardins summarized this idea as *inter-cluster relation signature* [6]: First, data objects of a certain type, assume c_i, are clustered based on their attributes using some propositional clustering methods; Then for other objects of type c_j referencing (or being referenced by) c_i, the inter-cluster relation signature is constructed as a K-dimensional vector, where K is the number of clusters obtained in the first step. The value of each dimension in the inter-cluster relation vector v_k is the number of edges that an object of type c_j has when being linked to objects of type c_i in cluster k. Hence, the data objects of type c_j are clustered based on the above inter-cluster relation vector in addition to their propositional attributes. The above iterative procedure continues until the clusters of all data types become stable.

The relational clustering algorithm motivated by the idea of mutual reinforcement was firstly implemented by Zeng et al. [160] in the scenario of clustering heterogeneous web objects, such as web-pages/users or queries/documents. They analyzed several cases of mutual reinforcement in clustering, and concluded that "when the links among nodes are dense enough and contain mostly correct information", the cluster results of one data type will improve that of other related data types. Additionally, they use a hybrid similarity function, a weighted sum of two similarity measures based on content features and link features, to compare data objects during the clustering procedure. The ReCoM framework proposed by Wang et al. [145] further developed this idea by incorporating the importance of data objects to improve the cluster quality. Besides

being used to group data objects in an iterative reinforcement fashion, the relationship structure is also used to differentiate the importance of data objects. More important objects, just as authoritative and hub nodes as produced by the HITS algorithm [75], can have more influence on the clustering procedure. When clustering data objects of a certain type, both of the above frameworks will transform the information of relationship structure into the link feature vector of the current objects according to the cluster result of other data types. By this means, the relational clustering task is propositionalized and thus propositional clustering algorithms could be easily embedded into each iterative clustering step to improve the efficiency.

Yin et al. [157] proposed another relational clustering approach, LinkClus, which is also performed by using reinforcement. Instead of using the numeric vectors to represent the content and linkage features of relational objects, in LinkClus they designed a hierarchical data structure *SimTree* to simplify the calculation of similarities between objects, and then use LinkClus to improve the SimTree iteratively. One drawback in LinkClus is that they only utilized frequent pattern mining techniques to build the initial SimTrees and use path-based similarity in one SimTree to adjust the similarity values and structures of other SimTrees, so information contained in the property attributes of data objects are not used in their clustering framework. Ignoring such information would definitely degrade the accuracy of the final clustering result.

2.2.3 Model-based Clustering

From the viewpoint of probability theory, data objects are assumed to be generated by a set of parametric probability models. The derived clusters, which are expected to reflect the natural structures hidden in the dataset, should match the underlying models. These probability distributions might be of different types or have the same density function but with different parameters. If the distributions are known, finding clusters within a given data set is equivalent to the problem of estimating parameters for underlying models. The Expectation-Maximization (EM) algorithm is a popular solution for this kind of

problem [17]. In propositional datasets, mixtures of multivariate Gaussian distributions are often used due to its well-developed theoretical foundation [42][45].

To analyze the co-occurrence data and discover the latent relationship between them, Hofmann and Puzicha developed the Separable Mixture Model (SMM) as a unified framework for statistical inference and clustering. By following the maximum likelihood principle, they used an improved EM algorithm to determine the model parameters. The grouping structure obtained by SMM can be regarded as a soft partitioning of the data space, in which the conditional probabilities are interpreted as class memberships of objects to a set of clusters. The basic SMM model might be extended into Asymmetric Clustering Model (ACM), Symmetric Clustering Model (SCM), and Hierarchical Clustering Model (HCM) [58]. The original SMM model can only handle one type of data co-occurrence, Bao et al. addressed this issue by explicitly adding a type variable into SMM so that their extended Typed Separable Mixture Model (TSMM) can exploit different types of co-occurrences [12].

Taskar et al. [142] proposed a general class of models for classification and clustering in relational domains. All relational instances are modeled by the framework of Probabilistic Relational Models (PRMs), in which the attributes of an instance are, based on a conditional probability distribution, determined by the related attributes of its parent instances. The parameters of PRMs are learned from data by utilizing the EM algorithm. Since the networks would be fairly complex, they have to adopt the strategy of belief propagation as the approximation scheme.

Liu et al. [97] extended the reinforcement relational clustering framework proposed by Zeng et al. [160]. Different from the numeric vector spaces used in the previous work, the new approach introduced two latent clustering layers, which serve as two mixture probabilistic models of the features. The content and the link features of each data object are generated by a content-specific probabilistic distribution (e.g. the Naïve Bayes model for web-pages) and a multi-nominal distribution respectively. Given the prior probabilities of all latent components, the probability of generating a data

object is determined by the combination of its content and link probability. The EM algorithm is used to estimate the parameters of the mixture models in each layer. And since the data are correlated between layers, the cluster result in one layer is propagated along the links to improve the calculation within the other layer. The algorithm performs iteratively until it reaches convergence.

2.2.4 Graph-based Clustering

Data clustering and graph partitioning, a sub domain in graph theory [147], share many common traits, and so we can easily use concepts and techniques of graph partitioning to define and solve the problem of data clustering: A weighted graph $G = \{V, E\}$ will be constructed to describe the target dataset D, where vertices V correspond to all data objects in D and the weight of edges E reflect the proximities between each pair of data points. The problem of clustering data objects in D is then equivalent to finding highly connected sub-graphs in G so that some prescribed properties are optimized, for example minimizing the sum of edge weights on the cut or maximizing the sum of edges weights within the partitioned sub-graphs. Actually graph partitioning techniques have been adopted in many propositional clustering approaches, such as Chameleon [68], CLICK [134], CAST [13] etc.

In the scenario of relational clustering, Neville et al. [110] provided some preliminary work of adapting graph-based techniques to incorporate both linkage structure and attribute information. In their approach, the similarity of a pair of related objects is determined by the number of common attributes they share. Objects that are not directly related in the linkage structure have zero similarity regardless of their attribute values. This similarity measure is used to weight edges of the graph G. Then three algorithms, Karger's Min-Cut [66], MajorClust [137] and spectral clustering with normalized-cuts criterion [135], are used to partition the graph G and thus generate the clustering result for the original dataset D. One disadvantage of this approach is that only the information contained in the first degree of linkage structure is exploited in clustering. It does

not consider the pairwise similarity between data objects that are linked via more than one intermediate object.

Han et al. [54] proposed a hyper-graph model to represent a relational dataset D. A hyper-graph extends the concept of a graph in the sense that each hyper-edge may connect more than two vertices. In their approach, an association rule algorithm such as Apriori [2] is firstly used to mine the frequent item sets in D. Each frequent item set is represented by a hyper-edge in the hyper-graph whose weight is equal to the average confidence of the association rules. Then the HMETIS [67] system is adopted to partition the hyper-graph and hence generate the cluster results.

In [99], a multi-type relational dataset is first formulated as a k-partitional graph, from which a Relation Summary Network (RSN) is constructed to find hidden data structures within the relational dataset. The quality of the derived RSN model is evaluated by the closeness of its linkage structure to the original k-partitional graph. Given a certain distance function between two graphs, it turns out to be an optimization problem of minimizing an objective criterion defined by a set of matrices, which represent the linkage information in the k-partitional graph and the RSN graph respectively. Since the above optimization problem is NP-hard, the authors use an iterative algorithm to find the local optimal RSN graph with the Bregman divergence [11]. Additionally, they provide a unified view of several different clustering algorithms, including bipartite spectral graph partitioning and k-means clustering, using their RSN model.

2.2.5 Spectral Clustering

Spectral clustering was proposed in the 1990s [118][53] and flourished very quickly after that. Rooted in spectral theory [139], spectral clustering approaches need to analyze the eigenvectors of an affinity matrix (also called as "similarity matrix" or "adjacent matrix") derived from the dataset. The problem of optimally partitioning the original dataset is usually converted into solving a set of algebraic equations [135] or performing the propositional clustering procedure in a new feature space spanned by the eigenvectors

[111]. Finally it is necessary to interpret the cluster indices for the original dataset from the computational result. Weiss reviewed several typical spectral clustering algorithms within a unified framework [146]. Zha et al. reformulated k-means clustering as a trace maximization problem associated with the Gram matrix of the data vectors. Then they showed that a relaxed trace maximization problem has the global optimal solution that can be obtained by computing a partial eigen-decomposition of the Gram matrix [161]. Since spectral clustering can handle non-spherical clusters and is easy to implement, it has been successfully applied in the field of speech separation, image segmentation, bio-data classification, etc.

Long et al. presented a general framework for multi-type relational clustering in [100]. Based on the assumption that the hidden structure of a data matrix can be explored by its factorization, the relational clustering is converted into an optimization problem: approximate the multiple relation matrices and the feature matrices by their corresponding collective factorization. Under this model a spectral clustering algorithm for multi-type relational data is derived, which updates one intermediate cluster indicator matrix as a number of leading eigenvectors at each iterative step until the result converges. Finally the intermediate matrices have to be post-processed to extract the meaningful cluster structure.

2.2.6 Fuzzy Clustering

Different from hard (crisp) clustering we introduced above in which each data objects belongs to only one cluster, soft (fuzzy) clustering specifies the membership degree of each object to each cluster. One of the most famous propositional fuzzy clustering algorithm is Fuzzy C-Means (FCM) [15]. It differs from the standard k-means by adding a fuzzification parameter to control the level of cluster fuzziness. Later, numerous FCM variants have been proposed to accommodate more proximity measures, optimize the performance or improve the drawback of FCM [59].

The idea of fuzzy clustering has also been extended to analyze relational datasets.

Three most popular algorithms are Fuzzy Non-metric Model (FNM) [128], Assignment Prototype Model (AP) [149] and Relational Fuzzy C-Means (RFCM) [57]. All these models require the availability of relation matrix that is composed of dissimilarities between every pair of data instances and RFCM further requires all data are in the Euclidean space. To ease the latter restriction, β-spread transformation was introduced to transform non-Euclidean relations into Euclidean ones. Krishnapuram et al. improved the efficiency of FCM and RFCM in [81] by reducing the number of data objects to be examined when updating the cluster prototypes. The objective function was designed based on the idea of Least Trimmed Squares [129] to achieve high robustness. Their algorithms, FCMdd and RFCMdd, performed well in the application of web mining. A recent progress was reported by Corsini et al. in [29]: assuming all relationships between data types are numerical, each data object can be represented as a vector of its relation strengths with other objects, and then the clustering procedure is similar to that of FCM, i.e. minimizing the Euclidean distance between each data object and the prototype of clusters.

2.2.7 Constraint-based Clustering

Until now all the relational clustering approaches we have introduced belong to unsupervised learning, which means they are performed under the assumption that there is no preliminary knowledge about the categorical information of the dataset and no constraints are specified that the discovered patterns should satisfy. In many applications, the incorporation of constraints can lead to a more efficient learning procedure as well as guide the search for patterns that are of most interest to users [19].

Železny et al. adopted a propositionalization approach, based on adapting rule learning and first-order feature construction, to Relational Subgraph Discovery (RSD) [144][84]. Initially, a set of first-order features are identified within a relational database by analyzing local variables (structural predicates) and generating new variables (utility predicates). Those derived features comply with user-defined constraints. In parallel the

traditional rule learner CN2 is improved by a weighted covering algorithm in which the weight of each data object is inversely proportional to the number of times it is covered by multiple rules. The weighted relative accuracy is used as the criterion in the improved CN2 algorithm to learn rules and thus formulate subgraphs. The syntactic and semantic constraints are exploited heavily in the pruning mechanism to reduce computational complexity.

Recently user-guided clustering also attracted research interest, where users can provide information or opinion to influence the clustering procedure. It is quite different from classification or semi-supervised learning [74][16], since the user's knowledge might be inaccurate or incomplete. Hence, it is necessary that the clustering algorithms have the capabilities of automatically evaluating the pertinence of the given information and utilizing it appropriately. Yin et al. developed a user-guided clustering algorithm CrossClus in [158]. If the user knows some features are very informative for the relational clustering task, he can specify such knowledge as an input of the clustering procedure. The algorithm will navigate the relationship structure to find other pertinent features that best match the user's requirement. All these pertinent features are associated to the target tuples by the technique of Tuple ID Propagation. When the feature selection is finished, each target tuple is transformed into a propositional vector that compounds the set of pertinent features. Hence the originally relational-clustering problem is propositionalized and propositional approaches, such as CLARANS [112] or k-means, are used to cluster the derived compound vectors.

Chapter 3

Representative Objects

The classic k-means algorithm is sensitive to outliers, because "an instance that is extremely far from others in the same cluster may substantially distort the mean of the cluster. This effect is particularly exacerbated due to the use of Sum-Of-Squared-Error criterion" [55]. To tackle this issue the means are replaced by the medoids in k-medoids, which are the data instances that have the maximum average similarity (or minimum average of distance) to the others in the same cluster. These medoids can be considered as a kind of representative objects. Another clustering approach that adopted the idea of representative objects is CURE (Clustering Using REpresentatives) [52]. The algorithm uses a set of well-scattered points to represent each cluster. The distances between these representatives control the clustering procedure and determine which sub-clusters should be merged into super-clusters.

Representatives are also applied in other data mining fields. In the classic k-NN method [31], every new instance is classified based on the votes of its k nearest neighbours. For fast nearest neighbour searching, the algorithm known as Approximating and Eliminating Search (AESA) [143] was introduced, which uses a set of *prototypes* to classify the new instance x. Later in [107] the base-prototypes were proposed to further alleviate the computational cost. Since all the distances from the base-prototypes to other instances have been pre-computed, the lower bound distance from x to proto-

types can be estimated using the triangle inequality and thus the k nearest neighbours are picked out for predicting the category of x. In [124] the author proposed a unified framework for instance selection, in which prototyping is an important step after sampling and clustering. Prototypes are assumed to be able to represent information of an entire subset of tuples. The author also distinguished two types of approaches: prototype selection and prototype construction.

This chapter is organized as follows: the definition of Representative Objects together with several basic RO-selection strategies are formulated in Section 3.1. Then two important extensions of the RO-selection strategies are proposed in Section 3.2. Some experimental results are provided in both sections.

3.1 Representative Objects

Generally speaking, any data instance that summarizes some features of the whole dataset can be considered as the *Representative Object* (RO). Reinartz described this idea by using word "prototype" in [124], which is regarded as condensed descriptions to represent information of the entire subset of tuples. According to the above definition, different sets of ROs might be selected from the same dataset to serve for different purposes of learning tasks. For example, in k-medoids the medoids can be used to represent their respective clusters, because a medoid is the data instance that has the maximum average similarity (or minimum average distance) with the other instances in the same cluster, or in other words the medoid approximates the mathematical expectation of data distribution in that cluster. In LAESA [107], the set of base-prototypes are regarded as ROs because they maximally spread in the dataset and are hence used (together with the triangle inequality) to estimate the lower bound distance between new data instances to the previously stored prototypes for classification. Formally we generalize the definition of Representative Objects follows:

Definition 3.1. The Representative Objects (ROs) are a set of instances in the dataset

$\boldsymbol{D}$ that can together provide a useful surrogate for $\boldsymbol{D}$ with respect to the data mining problems. For example, it is common for learning algorithms to use a random sample of data instances as the standard surrogate for the data space.

In the scenario of cluster analysis, we expect that the selected ROs can: (1) credibly reflect the geometric shape of data instances in the data space $\boldsymbol{S}$ and distinguish them from other subsets of data in $\boldsymbol{S}$; (2) be determined with low computational cost; (3) be helpful for facilitating the cluster analysis. One single RO (e.g. a medoid) does not satisfy the above requirement, because it cannot reflect the geometric shape of the dataset and, more importantly, the operation of identifying medoids requires the calculation of the similarity values between each pair of data instances in the given cluster, which leads to the quadratic computational complexity for the cluster analysis. Such computational inefficiency also exists in the agglomerative clustering approaches when they merge the pair of closest sub-clusters into one super-cluster and then try to identify the medoid of this new cluster. Similarly, the divisive hierarchical clustering approaches need to decide whether a cluster should be preserved or divided further. The criterion for division is usually the diameter of the cluster, which is calculated by the distance of two data instances in the cluster that are the farthest away from each other. Obviously using one single medoid to represent a cluster does not satisfy the requirements of ROs for our cluster analysis.

Before designing our algorithm of identifying ROs, we first consider another issue: how to measure the spread of a dataset $\boldsymbol{D}$ in the whole data space $\boldsymbol{S}$? Intuitively it should be measured by the *diameter* of $\boldsymbol{D}$:

Definition 3.2. The diameter of a dataset $\boldsymbol{D}$ is the largest distance between pairs of data instances in $\boldsymbol{D}$, i.e.

$$diam(\boldsymbol{D}) = \max_{\mathbf{x},\mathbf{y}\in\boldsymbol{D}} fd(\mathbf{x},\mathbf{y}) \tag{3.1}$$

However, computing the diameter of $\boldsymbol{D}$ has the quadratic complexity with respect to the size of $\boldsymbol{D}$ again, because the distances between every possible pair of data

instances in $\boldsymbol{D}$ have to be evaluated. Instead, assume we have obtained a set of ROs that well represent the data instances in $\boldsymbol{D}$, then evaluating the dataset spread based on these ROs would bring the advantage of great computational efficiency.

Definition 3.3. The variance of a dataset $\boldsymbol{D}$ is the largest distance between pairs of ROs in $\boldsymbol{D}$, i.e.

$$var(\boldsymbol{D}) = \max_{\mathbf{x},\mathbf{y}\in\{RO_i\}} fd(\mathbf{x},\mathbf{y}) \tag{3.2}$$

We use the word "variance" throughout this thesis to denote the largest distance between ROs, so it has a different meaning from its common usage in statistics. Later in this chapter we will see that $var(\boldsymbol{D})$ can approximate $diam(\boldsymbol{D})$ when the ROs are identified properly. A small value of $diam(\boldsymbol{D})$ or $var(\boldsymbol{D})$ means all the data instances reside in a certain compact part of $\boldsymbol{S}$ and thus are more close to each other. The difference between $diam(\boldsymbol{D})$ and $var(\boldsymbol{D})$ is that the former requires the comparison between all the data instances in $\boldsymbol{D}$ while the latter only requires the computation of proximity between ROs and, because their pairwise distances have been pre-computed, the total computational complexity will be greatly reduced when the number of ROs is far less than that of the data instances in $\boldsymbol{D}$.

Now the problem is: *How can a set of ROs* $\{\mathbf{ro}_i\}$ *(*$1 \leq i \leq r$ *where* r *is the user-specified number of ROs) be determined to best represent the distribution of the dataset*? Basically there are two types of approaches: prototype selection and prototype construction [124]. We follow the former routine and develop several strategies to identify ROs based on different concerns of maximal proximity[1]. They are named as *Max-One*, *Max-Sum* and *Max-Min* respectively and introduced in the follows.

After a start instance $\mathbf{x}_s$ is randomly chosen, the first RO is identified by:

$$\mathbf{ro}_1 = \arg\max_{\mathbf{x}\in D} fd(\mathbf{x},\mathbf{x}_s) \tag{3.3}$$

[1]In this and the following chapters, we assume an appropriate distance (or similarity) function has been defined for the data space, and so the type of the distance (or similarity) function will not be specified in the formula.

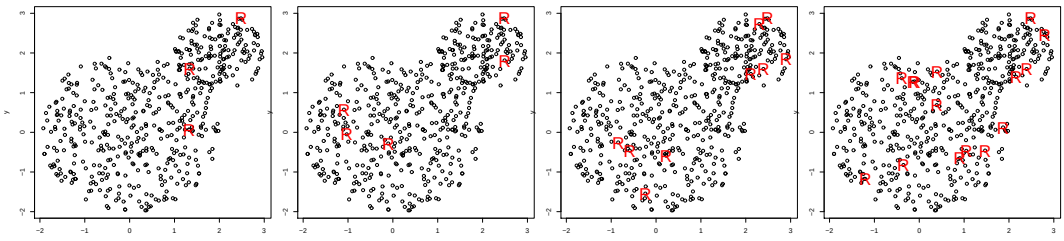
Table 3.1: Basic Strategies of RO-Selection

Algorithm	**Formula** $(2 \leq i \leq r, 1 \leq j < i\ )$
Max-One	$\mathbf{ro}_i = \underset{\mathbf{x} \in D - \{\mathbf{ro}_j\}}{\arg\max}\ fd(\mathbf{x}, \mathbf{ro}_j)$
Max-Sum	$\mathbf{ro}_i = \underset{\mathbf{x} \in D - \{\mathbf{ro}_j\}}{\arg\max} \left(\sum_j fd(\mathbf{x}, \mathbf{ro}_j) \right)$
Max-Min	$\mathbf{ro}_i = \underset{\mathbf{x} \in D - \{\mathbf{ro}_j\}}{\arg\max} \left(\min_j fd(\mathbf{x}, \mathbf{ro}_j) \right)$

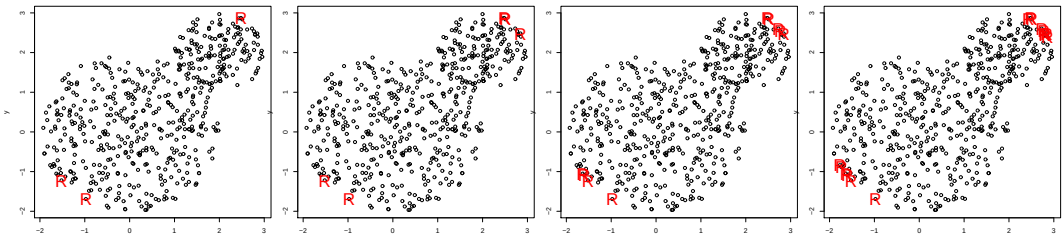
The reason we do not use $\mathbf{x}_s$ as an RO is: $\mathbf{x}_s$ is randomly selected among all the data instances and generally does not satisfy the criteria listed in Table 3.1 that maximize the proximities between ROs.

After $\mathbf{ro}_1$ is determined, the other $\mathbf{ro}_i$ $(2 \leq i \leq r)$ are identified by using different criteria to maximize the RO pairwise distances, which are summarized in Table 3.1. Although the *Max-One* strategy is simpler than the other two because only the farthest one from previously-selected ROs is considered in each iteration of the calculation, the derived ROs will oscillate between the endpoints of the longest direction within the dataset. In contrast, the *Max-Sum* and the *Max-Min* strategies consider all the previous selected ROs and are able to describe the distribution of the dataset more credibly. *Max-Sum* considers the accumulated distances from the current data instance to all the previously derived ROs, and so all the determined ROs are arranged along the border of the dataset. In contrast, *Max-Min* aims at maximizing the minimum distance between the current data instance to all the previously selected ROs, and so all the derived ROs will be located more uniformly in the data space.

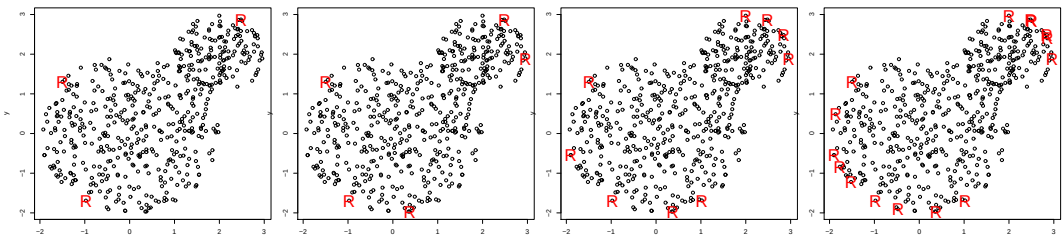
The whole procedure as well as the computational complexity of each step are summarized in Table 3.2: The start point $\mathbf{x}_s$ is randomly selected with the time com-



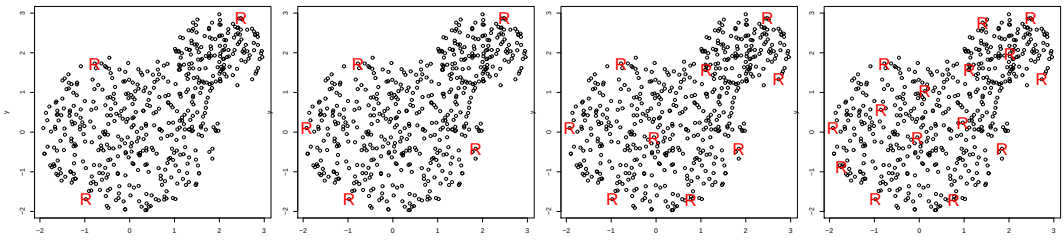
(a) Random: $r = 3$ (b) Random: $r = 5$ (c) Random: $r = 9$ (d) Random: $r = 15$



(e) Max-One: $r = 3$ (f) Max-One: $r = 5$ (g) Max-One: $r = 9$ (h) Max-One: $r = 15$



(i) Max-Sum: $r = 3$ (j) Max-Sum: $r = 5$ (k) Max-Sum: $r = 9$ (l) Max-Sum: $r = 15$



(m) Max-Min: $r = 3$ (n) Max-Min: $r = 5$ (o) Max-Min: $r = 9$ (p) Max-Min: $r = 15$

Figure 3.1: Comparison of RO-selection strategies

Table 3.2: Basic Strategies of RO-Selection

Procedure	Complexity	
	Time	Space
Randomly select a start instance $\mathbf{x}_s$	$O(1)$	0
Determine $\mathbf{ro}_1$ using Equation 3.3	$O(N)$	0
Determine the other $r-1$ ROs using either formula in Table 3.1	$O(rN)$	$O(rN)$

plexity of $O(1)$. We then determine $\mathbf{ro}_1$ by scanning the whole dataset to find the instance that is farthest away from $\mathbf{x}_s$. No extra information needs to be stored at this stage, and so the time and space complexities are $O(N)$ and 0 respectively. After that, the other ROs are determined by using either of the formulae in Table 3.1. To determine $\mathbf{ro}_2$, the pair-wise distances between $\mathbf{ro}_1$ and all the non-RO instances will be computed and stored in the memory. Iteratively, to determine $\mathbf{ro}_i$ the pair-wise distances between $\mathbf{ro}_{i-1}$ and all the non-RO instances will be computed and the pair-wise distances between $\mathbf{ro}_j$ $(1 \leq j \leq i-2)$ and all the non-RO instances are retrieved from the memory, so the time and space complexities are both $O(rN)$. In conclusion, the total time and space complexities for all the strategies discussed here are $O(rN)$.

Some empirical studies were conducted to validate our claims of these RO-selection algorithms. All the strategies were examined using a 2-D point dataset. As a baseline comparison, the *Random* strategy that selected a random sample of data instances as the set of ROs was also evaluated. Figure 3.1 show the results. As we expect, the ROs determined by the *Random* strategy cannot accurately reflect the distribution of the dataset when the parameter r is small; the *Max-One* strategy tends to aggregate ROs at the endpoints of the longest direction within the dataset. In contrast, the *Max-Min* strategy uniformly distributes the ROs in the datasets while *Max-Sum* prefers to arrange the ROs along the boundary of the dataset.

3.2 Extension of RO-Selection Strategies

The basic strategies of determining ROs listed in Table 3.1 can be extended by considering more issues. Here are two possible extensions:

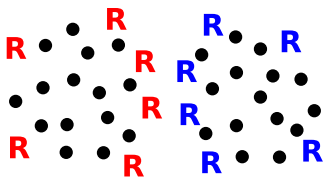


Figure 3.2: RO-Selection Depending on Other Clusters

Table 3.3: Enhanced Strategies of RO-Selection

Algorithm	**Formula** ($2 \leq i \leq r, 1 \leq j < i, 1 \leq j' \leq r$)
Max-One	$\mathbf{ro}_i = \underset{\mathbf{x} \in D - \{\mathbf{ro}_j\}}{\arg\max} \left(\alpha \cdot fd(\mathbf{x}, \mathbf{ro}_j) + (1-\alpha) \cdot fs(\mathbf{x}, \mathbf{ro}'_{j'}) \right)$
Max-Sum	$\mathbf{ro}_i = \underset{\mathbf{x} \in D - \{\mathbf{ro}_j\}}{\arg\max} \left(\alpha \cdot \sum_j fd(\mathbf{x}, \mathbf{ro}_j) + (1-\alpha) \cdot \sum_{j'} fs(\mathbf{x}, \mathbf{ro}'_{j'}) \right)$
Max-Min	$\mathbf{ro}_i = \underset{\mathbf{x} \in D - \{\mathbf{ro}_j\}}{\arg\max} \left(\alpha \cdot \min_j fd(\mathbf{x}, \mathbf{ro}_j) + (1-\alpha) \cdot \max_{j'} fs(\mathbf{x}, \mathbf{ro}'_{j'}) \right)$

3.2.1 Discriminate different data clusters

In some scenarios, the procedure of determining the ROs should be impacted by not only the data instances within the same cluster but also those from other clusters. When processing two neighbouring clusters, we wish to select ROs along the borders in order to better discriminate their member instances respectively. As shown in Figure 3.2, the selected ROs (denoted as letter "R" in the figure) identify the gap between two clusters. The results are very useful for the classification and clustering analysis because they can help us to distinguish the membership of data instances for each cluster.

To achieve this, each strategy in Table 3.1 needs to be adjusted by appending a factor that reflects the influence from other clusters. Such influence is computed based on the distance between ROs in two clusters: the new derived RO is expected to be

assigned at a location that maximizes its distance to all the previously selected ROs in the same cluster and at the same time minimizes its distance to all the ROs from different clusters. The latter constraint can be reiterated as maximizing the similarity between the current (candidate) RO and all the ROs from different clusters. Based on this idea, the strategies listed in Table 3.1 are respectively modified as those in Table 3.3 to process the case of only two clusters in total.

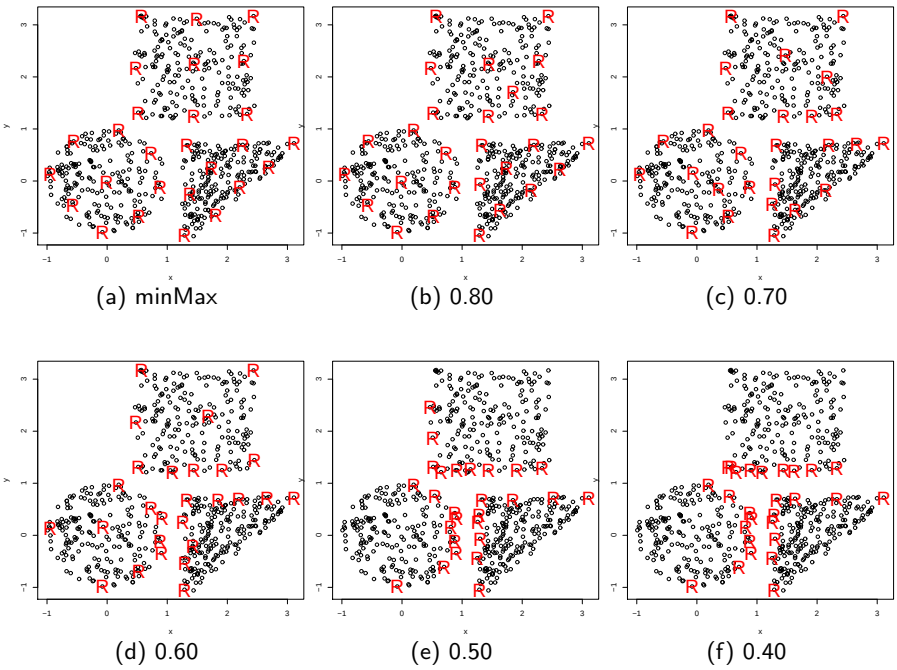
Given $\{\mathbf{ro}'_{j'}\}$ $(1 \leq j' \leq r)$ are the ROs in the neighbouring cluster, the factor of distances from the same clusters as well as that of distances from the other cluster are linearly combined by the parameter α, which is used to adjust the relative importance between these two factors:

- when $\alpha = 1$, the ROs from other clusters will not impact the identification of ROs for the current cluster, so the formulae in Table 3.3 are transformed back to those in Table 3.1, which means the procedure of determining ROs for each cluster is completely independent from that of the other clusters;

- when $\alpha = 0$, the previously selected ROs of a cluster will not impact the identification of subsequent ROs in the same cluster, so the procedure of determining ROs for a cluster is completely dependent on the distribution of ROs in the other clusters;

- when α is between 0 and 1, the procedure of identifying the successive ROs is impacted by both the previously selected ROs in the same cluster and the distribution of ROs in the other clusters.

The whole procedure is launched in an iterative fashion: In the first iteration, the RO set for each cluster is determined independent from the others by utilizing one of the formulae in Table 3.1. From the second iteration and afterwards, the distribution of ROs in other clusters will be considered in the identification of ROs for the current cluster. This procedure continues iteratively until all the ROs in all the clusters become stable (no changes in the ROs).



(a) minMax (b) 0.80 (c) 0.70

(d) 0.60 (e) 0.50 (f) 0.40

Figure 3.3: Determine ROs to discriminate the clusters

All the equations in Table 3.3 can be generalized to process the case of K clusters. We define the set $\{\mathbf{ro}_{k,i}\}$ as all the ROs of the cluster k, where $1 \leq k \leq K$. The Max-Sum strategy is demonstrated in Algorithm 1 [2]. We also give a brief complexity analysis for the above algorithm here, because it is very similar to that in Section 3.1. Assuming cluster k has N_k instances in total, it can be deducted easily that the time and space complexities are $O(rK\sum_k N_k)$ and $(r\sum_k N_k)$ respectively, or $O(rKN)$ and (rN) respectively because $\sum_k N_k = N$.

To show the effectiveness of our new strategies proposed above, some experiments were conducted upon a 2-D dataset with three clusters. The value of parameter α was decreased from 1.0 to 0.4 (here we ignore the results when $\alpha < 0.4$ because they are the same as that when $\alpha = 0.4$). According to the formulae given in Table 3.3, the impact of ROs from other clusters will be more and more critical in the procedure

[2]We can change the lines 5, 7, 13 and 15 in Algorithm 1 to get the other RO-selection strategies in Table 3.3.

Algorithm 1: Iterative RO-Selection Procedure Using the Max-Sum Strategy

Input: clusters $\{D_k\}$, number of ROs r, variance threshold v, weight α
Output: RO set $\{\mathbf{ro}_{k,i}\}$

begin
  `// The first iteration`
  Set $iter \leftarrow 1$;
  foreach *cluster* $D_k \in \{D_k\}$ **do**
    Start point $\mathbf{x}_{k,s} \leftarrow$ randomly selected an instance in D_k;
    $\mathbf{ro}_{k,1}^{iter} \leftarrow \arg\max_{\mathbf{x} \in D_k} fd(\mathbf{x}, \mathbf{x}_{k,s})$;
    for $i \leftarrow 2$ **to** r **do**
      $\mathbf{ro}_{k,i}^{iter} \leftarrow \arg\max_{\mathbf{x} \in D_k - \{\mathbf{ro}_{k,j}^{iter}\}} \sum_j fd(\mathbf{x}, \mathbf{ro}_j)$, where $1 \leq j < i$;
    end
  end
  `// Iteratively update the RO set`
  repeat
    $iter \leftarrow iter + 1$;
    foreach *cluster* $D_k \in \{D_k\}$ **do**
      $\mathbf{ro}_{k,1}^{iter} \leftarrow \arg\max_{\mathbf{x} \in D_k} \left(\sum_j fs(\mathbf{x}, \mathbf{ro}_{k',j}^{iter-1}) \right)$;
      for $i \leftarrow 2$ **to** r **do**
        $\mathbf{ro}_{k,i}^{iter} \leftarrow$
        $\arg\max_{\mathbf{x} \in D_k - \{\mathbf{ro}_{k,j}^{iter}\}} \left(\alpha \cdot \sum_j fd(\mathbf{x}, \mathbf{ro}_{k,j}^{iter}) + (1-\alpha) \cdot \sum_{j'} fs(\mathbf{x}, \mathbf{ro}_{k',j'}^{iter-1}) \right)$;
      end
    end
  until $\{\mathbf{ro}_{k,i}^{iter}\} = \{\mathbf{ro}_{k,i}^{iter-1}\}$;
  return $\{\mathbf{ro}_{k,i}^{iter}\}$.
end

of determining ROs for the current cluster as α becomes smaller. At the first phase of the experiment, the strategy (for example *Max-Min*) is applied upon each cluster to generate its own RO set; then each of the derived RO sets is iteratively updated based on the impacts from other RO sets until they are stable. The results are shown in Figure 3.3. As we expected, the derived ROs gradually move to identify the boundaries between clusters as α decreases.

3.2.2 Determine ROs incrementally

All the RO-selection strategies introduced earlier are only suitable for processing static (non-incremental) datasets. As analyzed in the previous sections, their computational complexity of determining ROs is linear to the size of the dataset D. However, in the case that new data objects are incrementally added into D, if we utilize these strategies in a brute-force manner, i.e. launching them from scratch at each time a new data object is inserted, the whole computational complexity will become quadratic. Hence, more efficient RO-selection algorithms are necessary for the incremental learning scenario.

Actually, the search of ROs can be viewed as an optimization problem. Here we use the *Max-Min* strategy in Table 3.1 as an example to explain our idea and then extend the discussion to other strategies. Given the dataset D of which the set $\{\mathbf{ro}_i\}$ $(1 \leq i \leq r)$ have been selected, the max-min criterion for ROs is equivalent to maximizing the objective function:

$$L = \sum_{i=1}^{r} \left(\min_{\substack{1 \leq j \leq r \\ j \neq i}} fd(\mathbf{ro}_i, \mathbf{ro}_j) \right) \tag{3.4}$$

The existing ROs will be incrementally updated if a new instance $\mathbf{x}$ is added: All the $\mathbf{ro}_i$ $(1 \leq i \leq r)$ are in turn examined to see which one of them might be replaced by $\mathbf{x}$ so that the new RO set can maximally increase the function L for the new dataset $D \cup \{\mathbf{x}\}$. Theoretically there are r possibilities to be examined within the set $\{\mathbf{ro}_i\}$. To reduce duplicated computation, all the pairwise distances between existing ROs could be stored in the memory. We only need to compare $\mathbf{x}$ with $\mathbf{ro}_i$ $(1 \leq i \leq r)$ and retrieve

other distance values from the memory, so the time and space complexities are $O(r)$ and $O(r^2)$ respectively.

Deeper investigation shows that the space complexity can be reduced to $O(r)$ as well. Without the loss of generality, assume that $\mathbf{ro}_R$ is replaced by $\mathbf{x}$, denoted as $\mathbf{x} \rightarrow \mathbf{ro}_R$. We use **RO** to represent the whole RO set, i.e. $\mathbf{RO} = \{\mathbf{ro}_i\}$ $(1 \leq i \leq r$. $NB(\mathbf{ro}_i)$ refers to the RO that is nearest to $\mathbf{ro}_i$: $NB(\mathbf{ro}_i) = \arg\min_{\mathbf{ro}_j} fd(\mathbf{ro}_i, \mathbf{ro}_j)$. Then:

$$
\begin{aligned}
& L_{\mathbf{x} \rightarrow \mathbf{ro}_R} \\
&= \min_{\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\}} fd(\mathbf{x}, \mathbf{ro}_i) + \sum_{\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\}} \left(\min_{\substack{\mathbf{ro}' \in \mathbf{RO} - \{\mathbf{ro}_R\} \\ \cup \{\mathbf{x}\}}} fd(\mathbf{ro}_i, \mathbf{ro}') \right) \\
&= \min_{\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\}} fd(\mathbf{x}, \mathbf{ro}_i) \\
&\quad + \sum_{\substack{\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\} \\ NB(\mathbf{ro}_i) = \mathbf{ro}_R}} \min \Big(fd(\mathbf{ro}_i, \mathbf{rn}_{i,2}), fd(\mathbf{ro}_i, \mathbf{x}) \Big) \\
&\quad + \sum_{\substack{\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\} \\ NB(\mathbf{ro}_i) \neq \mathbf{ro}_R}} \min \Big(fd(\mathbf{ro}_i, \mathbf{rn}_{i,1}), fd(\mathbf{ro}_i, \mathbf{x}) \Big)
\end{aligned}
$$

where $\mathbf{rn}_{i,1}$ and $\mathbf{rn}_{i,2}$ represent the first and second closest ROs to $\mathbf{ro}_i$ respectively. The first item in the above result is the minimum distance between $\mathbf{x}$ and all the current ROs except $\mathbf{ro}_R$. The second item considers those ROs of which the closest RO is $\mathbf{ro}_R$. Because $\mathbf{ro}_R$ will be replaced by $\mathbf{x}$, the value of L is determined by the comparison of $fd(\mathbf{ro}_i, \mathbf{rn}_{i,2})$ and $fd(\mathbf{ro}_i, \mathbf{x})$. In contrast, the third item considers the other ROs of which the closest RO is not $\mathbf{ro}_R$, so L is determined by the comparison of $fd(\mathbf{ro}_i, \mathbf{rn}_{i,1})$ and $fd(\mathbf{ro}_i, \mathbf{x})$.

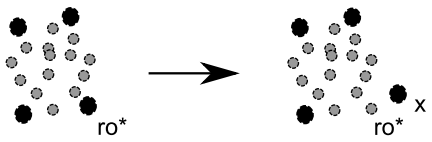


Figure 3.4: Dynamically update ROs

The change of L after $\mathbf{ro}_R$ is replaced by $\mathbf{x}$ is:

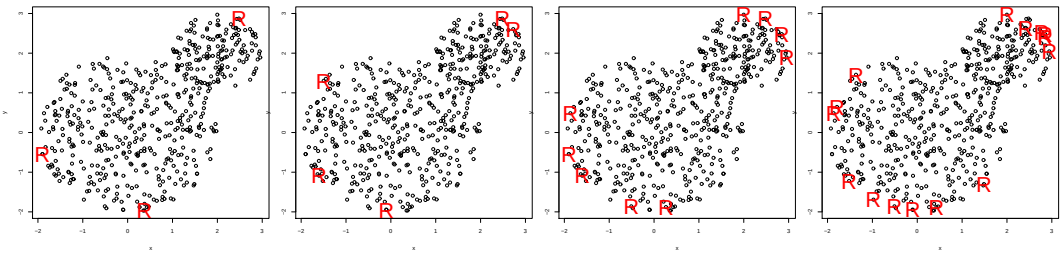
$$
\begin{aligned}
&\Delta L_R \\
&= L_{\mathbf{x}\rightarrow\mathbf{ro}_R} - L \\
&= \min_{\mathbf{ro}_i \in \mathbf{RO}-\{\mathbf{ro}_R\}} fd(\mathbf{x}, \mathbf{ro}_i) - fd\Big(\mathbf{ro}_R, NB(\mathbf{ro}_R)\Big) \\
&\quad + \sum_{\substack{\mathbf{ro}' \in \mathbf{RO}-\{\mathbf{ro}_R\} \\ NB(\mathbf{ro}_i)=\mathbf{ro}_R}} \Big[\min\Big(fd(\mathbf{ro}_i, \mathbf{rn}_{i,2}), fd(\mathbf{ro}_i, \mathbf{x})\Big) - fd(\mathbf{ro}_i, \mathbf{ro}_R)\Big] \\
&\quad + \sum_{\substack{\mathbf{ro}' \in \mathbf{RO}-\{\mathbf{ro}_R\} \\ NB(\mathbf{ro}_i)\neq\mathbf{ro}_R}} \Big[\min\Big(fd(\mathbf{ro}_i, \mathbf{rn}_{i,1}), fd(\mathbf{ro}_i, \mathbf{x})\Big) - fd(\mathbf{ro}_i, \mathbf{rn}_{i,1})\Big] \qquad (3.5)
\end{aligned}
$$

If $\Delta L_R > 0$, we select $R^* = \arg\max_R \Delta L_R$ and replace $\mathbf{ro}_{R^*}$ by $\mathbf{x}$ to form the new RO set, otherwise no replacement happens. According to Equation 3.5, we can store the first and second closest ROs for each RO as well as the corresponding distance values into memory. In order to compute $\mathbf{ro}_{R^*}$ that is replaced by $\mathbf{x}$, we only need to calculate the distances between $\mathbf{x}$ and existing ROs and retrieve the distance value from the memory, so the space and time complexities are both $O(r)$. In summary, the computational complexity is $O(r)$ when only one data instance is added into the existing dataset. If the incremental strategy is used to process the dataset D, we can assume that D is empty at first and gradually appended by N data instances, so the above incremental strategy is iteratively applied as data instances are added, so the total computational complexity is $O(rN)$.

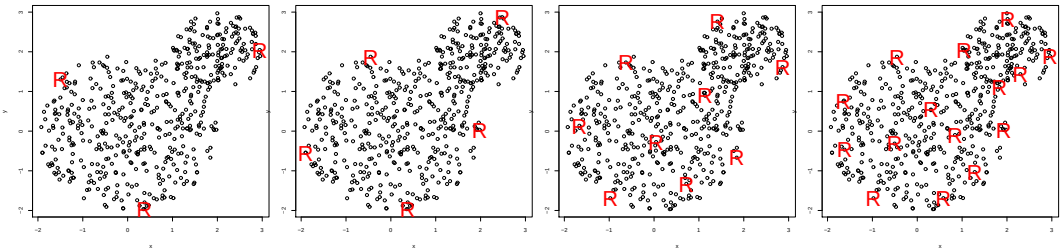
Figure 3.4 is an example to show the idea of updating ROs: the original $\mathbf{ro}^*$ is replaced by the new data object $\mathbf{x}$, because $\mathbf{x}$ can better represent the shape of the new dataset, or equivalently $\Delta L_R > 0$ based on the Max-Min criterion.

Table 3.4: Incremental Strategies of RO-Selection

Algorithm	$\Delta\, L_R = L_{\mathbf{x}\rightarrow \mathbf{ro}_R} - L$
Max-One	$\max\limits_i fd(\mathbf{x}, \mathbf{ro}_i) - \max\limits_i fd(\mathbf{ro}_R, \mathbf{ro}_i)$ $+ \sum\limits_{\substack{i \\ FAR(\mathbf{ro}_i)=\mathbf{ro}_R}} \Big[\max\Big(fd(\mathbf{ro}_i, \mathbf{rf}_{i,2}), fd(\mathbf{ro}_i, \mathbf{x})\Big) - fd(\mathbf{ro}_i, \mathbf{ro}_R)\Big]$ $+ \sum\limits_{\substack{i \\ FAR(\mathbf{ro}_i)\neq\mathbf{ro}_R}} \Big[\max\Big(fd(\mathbf{ro}_i, \mathbf{rf}_{i,1}), fd(\mathbf{ro}_i, \mathbf{x})\Big) - fd(\mathbf{ro}_i, \mathbf{rf}_{i,1})\Big]$ where: $\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\}$, $FAR(\mathbf{ro}_i)$ represents $\mathbf{ro}_i$'s farthest RO, $\mathbf{rf}_{i,1}$ and $\mathbf{rf}_{i,2}$ are the 1st and 2nd farthest ROs to $\mathbf{ro}_i$ respectively.
Max-Sum	$\sum\limits_{\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\}} \Big[fd(\mathbf{x}, \mathbf{ro}_i) - fd(\mathbf{ro}_R, \mathbf{ro}_i)\Big]$
Max-Min	$\min\limits_i fd(\mathbf{x}, \mathbf{ro}_i) - \min\limits_i fd(\mathbf{ro}_R, \mathbf{ro}_i)$ $+ \sum\limits_{\substack{i \\ NB(\mathbf{ro}_i)=\mathbf{ro}_R}} \Big[\min\Big(fd(\mathbf{ro}_i, \mathbf{rn}_{i,2}), fd(\mathbf{ro}_i, \mathbf{x})\Big) - fd(\mathbf{ro}_i, \mathbf{ro}_R)\Big]$ $+ \sum\limits_{\substack{i \\ NB(\mathbf{ro}_i)\neq\mathbf{ro}_R}} \Big[\min\Big(fd(\mathbf{ro}_i, \mathbf{rn}_{i,1}), fd(\mathbf{ro}_i, \mathbf{x})\Big) - fd(\mathbf{ro}_i, \mathbf{rn}_{i,1})\Big]$ where: $\mathbf{ro}_i \in \mathbf{RO} - \{\mathbf{ro}_R\}$, $NB(\mathbf{ro}_i)$ represents $\mathbf{ro}_i$'s nearest RO, and $\mathbf{rn}_{i,1}$ and $\mathbf{rn}_{i,2}$ are the 1st and 2nd closest ROs to $\mathbf{ro}_i$ respectively.



(a) Max-Sum: $r = 3$ (b) Max-Sum: $r = 5$ (c) Max-Sum: $r = 9$ (d) Max-Sum: $r = 15$



(e) Max-Min: $r = 3$ (f) Max-Min: $r = 5$ (g) Max-Min: $r = 9$ (h) Max-Min: $r = 15$

Figure 3.5: Select ROs using incremental Max-Sum and Max-Min strategies

We can also define the function L for the *Max-One* and *Max-Sum* strategies as:

$$\text{Max-One: } L = \sum_{i=1}^{r} \left(\max_{1 \le j \le r} fd(\mathbf{ro}_i, \mathbf{ro}_j) \right) \tag{3.6}$$

$$\text{Max-Sum: } L = \sum_{i=1}^{r} \left(\sum_{1 \le j \le r} fd(\mathbf{ro}_i, \mathbf{ro}_j) \right) \tag{3.7}$$

and then deduce the incremental formula for them in a similar way. Table 3.4 summarizes all three incremental strategies. They are developed based on the same idea but differ from the criteria of updating ROs when new data are available. When applying these strategies to determine the ROs for the dataset D, each data instance is sequentially scanned. If the total number of scanned data instances is less than the required number of ROs, the current data instance x is added as the next RO; otherwise x is compared with all the previously chosen ROs using one of the strategies listed in Table 3.4 to see whether x can replace one of the existing ROs.

Table 3.5: Average Distance between ROs

	Basic	**Incremental**
Max-Sum	0.4944 ± 0.0079	0.4965 ± 0.0091
Max-Min	0.3766 ± 0.0075	0.3797 ± 0.0083

The incremental *Max-Sum* and *Max-Min* strategies in Table 3.4 were tested and the experimental results are shown in Figure 3.5. We can see that these incremental algorithms can obtain similar results as their non-incremental versions (in Figure 3.1).

Finally, we examined the impact of randomness upon our strategies. The randomness comes from two aspects: (1) the dataset itself is input into the algorithm in a random order; (2) the start instance $\mathbf{x}_s$ is randomly selected in Equation 3.3. Hence, we set two random number generators to simulate the above random effects respectively. Our algorithms were repeated for 100 times with different random seeds and the experimental results are reported in Table 3.5. The first column (with title "Basic") evaluates the randomness upon the basic RO-selection strategies in Table 3.1 and the second one (with title "Incremental") evaluates the incremental strategies in Table 3.4. We can see that the average pair-wise distances between ROs are very stable, which means the impact of both kinds of randomness are effectively eliminated from our algorithms.

3.3 Summary

In this chapter we introduce the idea of Representative Objects and proposed several strategies to identify the ROs for a given dataset. The ROs are identified based on the maximization of some proximity measures such as Max-Sum or Max-Min, so they can reflect the distribution of data instances. More important, our algorithms have linear computational complexity and are therefore applicable for analyzing very large datasets. We also extend the basic strategies in cases of discriminating multiple data clusters and incremental updating ROs. Several datasets, including synthetic and real ones, are used to evaluate our algorithms. More empirical studied will be performed in the next section to show the application of ROs in relational clustering.

Chapter 4

DIVA Clustering Framework

As discussed in Chapter 2, the more domain information is exploited in the cluster analysis, the more credible the results will be generated, but at the same time the higher computational cost we have to afford. Since in many cases the similarity/distance measures are pre-specified by the application requirements, we aim to develop a general clustering framework that accommodates all kinds of measures and is therefore suitable for processing propositional as well as relational datasets. The efficiency gain of our framework is achieved by reducing the total number of similarity/distance computations between data instances and the cluster prototypes (i.e. representative objects introduced in Chapter 3) during the cluster analysis.

In this chapter we discuss in details our multi-phase clustering framework named *DIVA* (DIVision and Agglomeration), which is based on the idea of Representative Objects that has been introduced in Chapter 3. The framework is summarized in Algorithm 2, consisting of the divisive step (Section 4.1) and the agglomerative step (Section 4.2). Their computational complexity are analyzed in Section 4.3. Finally, we conducted comprehensive experiments to prove the effectiveness and efficiency of our algorithms. The experimental results are studied Section 4.4 and Section 4.5.

Algorithm 2: Main Framework of DIVA

Input: dataset D, number of ROs r, variance threshold v
Output: cluster result

```
1 begin
2     micro-clusters {C_k} ← call DivStep (D, r, v);
3     dendrogram T ← call AggStep ({C_k});
4     Construct the cluster result based on T;
5 end
```

4.1 Divisive Step

The divisive step is designed to partition the whole dataset D into a number of micro-clusters so that data instances in each micro-cluster are of high homogeneity. Since the concept of *variance* (Definition 3.3) is used to measure the spread of a cluster, i.e. smaller values of variance means data instances are located closer to each other in the data space and thus are more homogeneous, we set the variance threshold v as a parameter of the algorithm to control the homogeneity of the micro-clusters derived by the divisive step.

Given the non-incremental and the incremental RO-selection strategies introduced in Sections 3.1 and 3.2.2, we propose two implementations for the divisive step:

4.1.1 Recursive Approach

The recursive implementation (summarized in Algorithm 3) starts by assuming all the data instances belong to the same cluster. We use C_0 to denote the initial cluster that corresponds to the whole dataset D. An appropriate RO-selection strategy[1] from Table 3.1 is applied to find the RO set for C_0 and the variance $var(C_0)$ is calculated based on the Definition 3.3. If $var(C_0)$ is greater than the pre-specified variance threshold v, the division procedure is launched: Two ROs (assumed as $\mathbf{ro}_1$ and $\mathbf{ro}_2$) that are farthest away from each other, i.e. the pair of ROs determining the variance of C_0, are used

[1]Since we use the binary partition in our divisive step which only depends on the farthest pair of ROs in the data, all the RO-selection strategies will generate similar results here.

Algorithm 3: RecDivStep

Input: dataset D, number of ROs r, variance threshold v
Output: micro-clusters $\{C_k\}$

begin
  Initialize the cluster set: $C_0 = D$;
  foreach *newly added cluster* C_k **do**
    `// Generate the set of ROs` $\{\mathbf{ro}_i^{(C_k)}\}$
    begin
      Start object $\mathbf{x}_s^{(C_k)} \leftarrow$ randomly select an instance from C_k;
      `// Determine` $\mathbf{ro}_i^{(C_k)}$ $(1 \leq i \leq r)$
      $\mathbf{ro}_1^{(C_k)} \leftarrow$ select the instance $\mathbf{x} \in C_k$ that is farthest away from the start point $\mathbf{x}_s^{(C_k)}$;
      for $i \leftarrow 2$ **to** r **do**
        $\mathbf{ro}_i^{(C_k)} \leftarrow$ select the instance $\mathbf{x} \in C_k$ that maximizes the accumulated distance from itself to the existing ROs $\mathbf{ro}_j^{(C_k)}$ $(1 \leq j < i)$, as described in Section 3.1;
      end
    end

    `// Assume the farthest pair of ROs in` $\{\mathbf{ro}_i^{(C_k)}\}$ `are` $\mathbf{ro}_1^{(C_k)}$ `and` $\mathbf{ro}_2^{(C_k)}$
    $var(C_k) \leftarrow fd(\mathbf{ro}_1^{(C_k)}, \mathbf{ro}_2^{(C_k)})$;
    if $var(C_k) > v$ **then**
      Create two new clusters $C_{k'}$ and $C_{k''}$, using $\mathbf{ro}_1^{(C_k)}$ and $\mathbf{ro}_2^{(C_k)}$ as the absorbent objects for $C_{k'}$ and $C_{k''}$ respectively;
      Allocate the other instances $x \in C_k$ into either $C_{k'}$ or $C_{k''}$ based on the comparison of $fd(\mathbf{x}, \mathbf{ro}_1^{(C_k)})$ and $fd(\mathbf{x}, \mathbf{ro}_2^{(C_k)})$;
      Use $C_{k'}$ and $C_{k''}$ to replace C_k in the cluster set;
    end
  end
  return *clusters* $\{C_k\}$
end

as the absorbent objects for two sub-clusters respectively. Since we have known the distances between all pairs of ROs, the effort of identifying $\mathbf{ro}_1$ and $\mathbf{ro}_2$ is trivial. Other data instances are allocated to their closest sub-cluster based on their distance to the absorbent objects. Finally, the original cluster C_0 is replaced by its derived sub-clusters in the cluster set. Since the distance (or similarity) values of all the non-RO instances to all the ROs have been persisted in the memory when determining $\{\mathbf{ro}_i^{(C_0)}\}$, the division operation for C_0 can be performed without extra computational effort.

The above division process is performed recursively on sub-clusters of which the variance is greater than the threshold v. Finally we obtain a set of micro-clusters $\{C_k\}$ ($\bigcup_k C_k = C_0,\ C_{k_1} \bigcap C_{k_2} = \emptyset$). The variance of each micro-cluster is less than or equal to v. These micro-clusters are then used as the input of the agglomerative step in Section 4.2.

It is worth noting that our cluster results are not sensitive to the value of r because we use the binary split here to divide clusters and $(\mathbf{ro}_1, \mathbf{ro}_2)$ are in general the largest pair of ROs among the others. According to Algorithm 3, the randomly selected start object $\mathbf{x}_s$ is likely to be located at the center region of cluster C_k, then $\mathbf{ro}_1$ is identified along the longest direction that C_k spreads in the data space and $\mathbf{ro}_2$ is at the direction opposite to $\mathbf{ro}_1$. Therefore, the distance between $\mathbf{ro}_1$ and $\mathbf{ro}_2$ approximates the diameter of C_k, and they are chosen as the absorbent objects to construct $C_{k'}$ and $C_{k''}$ respectively. Other ROs might be utilized if multiple-split instead of binary-split is adopted in our divisive step, but such discussion is out of our scope in this thesis. Our above claim is supported by the experimental results shown in Section 4.5. When generating clusters for a set of sampled data instead of the whole dataset, the cluster results should not be sensitive to the value of v, if we assume the distribution of any sampled data is the same as that of the whole dataset.

Algorithm 4: IncDivaStep

Input: dataset D, number of ROs r, variance threshold v, branching factor B
Output: micro-clusters $\{C_k\}$

```
begin
    Initialize the Search Tree S = ∅;
    foreach x ∈ D do
        Call IncBuild (S, B, r, v, x).
    end
    Construct the cluster result {C_k} based on the leaf nodes of S;
    return clusters {C_k};
end
```

4.1.2 Incremental Approach

In contrast, the incremental implementation (summarized in Algorithm 4) sequentially scans the dataset D by comparing each data instance with the current set of micro-clusters. Given that $\{\mathbf{ro}_i^{(C)}\}$ is the RO set for the micro-cluster C, the distance between the instance $\mathbf{x}$ and the cluster C is defined by:

$$fd(\mathbf{x}, C) = \max_{1 \leq i \leq r} fd(\mathbf{x}, \mathbf{ro}_i^{(C)}) \tag{4.1}$$

Similar to the recursive implementation, the variance threshold v is used to control the homogeneity of the derived micro-clusters, so an instance $\mathbf{x}$ is assigned into the cluster C only when they satisfy the following constraint:

$$fd(\mathbf{x}, C) \leq v \tag{4.2}$$

The above constraint together with Eq. 4.1 guarantees the previously assigned data instances as well as the new instances x, which together constitute the updated cluster C', satisfy the variance requirement. Instead, if we use the minimum or the average function in Eq. 4.1, a possible case is: the distance between x and its nearest RO is small than v while the distance between x and its furthest RO is greater than v, which

Algorithm 5: IncBuild

Input: search tree S, branching factor B, number of ROs r, variance threshold v, data instance $\mathbf{x}$

begin
  if $S = \emptyset$ **then**
    Create $C_0 \leftarrow \{\mathbf{x}\}$;
    Set $\texttt{root}(S) \leftarrow C_0$;
  else
    Find $C^* \in S$ for $\mathbf{x}$ based on Eq.4.3 ;
    Set $C_p \leftarrow \texttt{parent}(C^*)$;
    // Assume $\{\mathbf{ro}_i^{(C^*)}\}$ is the set of ROs for C^*
    if $fd(\mathbf{x}, C^*) \leq v$ **then**
      Assign $\mathbf{x}$ into the cluster C^*;
      if $|\{\mathbf{ro}_i^{(C^*)}\}| < r$ **then**
        $\mathbf{x}$ is added into $\{\mathbf{ro}_i^{(C^*)}\}$ as a new RO;
      else
        $\mathbf{ro}_{R_M}^{(C^*)}$ is replaced by $\mathbf{x}$ in $\{\mathbf{ro}_i^{(C^*)}\}$;
      end
    else
      Create new cluster $C' \leftarrow \{\mathbf{x}\}$;
      Set $\texttt{children}(C_p) \leftarrow \texttt{children}(C_p) \cup C'$;
    end
    Update the ROs of C_p and its super-nodes recursively;
    // Adjust the structure of S when necessary
    if $|\texttt{children}(C_p)| > B$ **then**
      Find the farthest pair of ROs in C_p. Without loss of generality, assume they are $\mathbf{ro}_1^{(C_p)}$ and $\mathbf{ro}_2^{(C_p)}$;
      Divide $\texttt{children}(C_p)$ into two groups based on their distance to $\mathbf{ro}_1^{(C_p)}$ and $\mathbf{ro}_2^{(C_p)}$;
      Link these two groups to new parents $\widetilde{C}_1, \widetilde{C}_2$ respectively;
      Set $\texttt{parent}(\widetilde{C}_1) \leftarrow C_p$, $\texttt{parent}(\widetilde{C}_2) \leftarrow C_p$ and $\texttt{children}(C_p) = \{\widetilde{C}_1, \widetilde{C}_2\}$;
    end
  end
end

means $var(C') > v$ and C' has to be split. It is similar to the unbound problem found in the First Leader algorithm [56], i.e. with more data assigned into a cluster, the cluster center might be moved gradually or the cluster shape becomes skewed, which makes its radius unbounded. Therefore, to make sure the variance of the derived cluster keeps being less than v, we use the maximum function in Eq. 4.1.

In case that more than one candidate clusters satisfy the constraint, the closest cluster C^* is used to absorb $\mathbf{x}$:

$$C^* = \arg\min_{C} fd(\mathbf{x}, C) \tag{4.3}$$

To facilitate the search for an appropriate cluster that the new data should be assigned into, the micro-clusters $\{C_k\}$ are indexed by a balanced search tree S. The leaf nodes in S correspond to micro-clusters and the non-leaf nodes are the union of their sub-nodes[2]. Each node in S has r ROs. The ROs of a non-leaf node are selected from the union of its children's ROs using the strategies introduced in Chapter 3. Additionally the parameter B is used to control the maximum number of sub-nodes that a non-leaf node may have. The algorithm for inserting a new instance x into a micro-cluster is as follows:

1. Identify the closest leaf node in S: Starting from the root node, the instance x traverses S from top to bottom by recursively selecting the closest sub-node at each level, using Equation 4.1, until the leaf node C^* is found.

2. Modify the node: If $fd(\mathbf{x}, C^*) \leq v$, the instance x will be absorbed into the cluster C^*; Otherwise, a new leaf node C' is created as the sibling node of C^* to contain x. In the latter case, we examine whether the number of C^*'s sibling nodes is less than B; if not, the parent node will be split by using two farthest ROs of the parent node as the absorbent seeds and re-distribute all the sub-clusters.

3. Update the ROs of the modified node: if x is absorbed into cluster C^*, we check whether the size of ROs within C^* is less than r: if so, x becomes a new RO

[2]We use the words "cluster" and "node" interchangeable here.

in C^*; otherwise, each of the existing ROs will be further examined, using the strategies described in Section 3.2.2, to see whether it can be replaced by x.

Step 3 is performed recursively from bottom to top to update the leaf nodes until the root node is reached. Algorithm 5 summarizes the incremental-build algorithm.

4.2 Agglomerative Step

Although micro-clusters derived by either the recursive or the incremental divisive step have sufficient intra-cluster homogeneity, they do not preserve the information of *closeness* (or distance) between clusters [68]. The closeness is in essence a criterion to evaluate the inter-cluster homogeneity. There are three common techniques to compute the distance between clusters, defined as follows [35]:

$$\text{Single-Linkage: } fd(C, C') = \min_{i,j} fd(\mathbf{x}_i^{(C)}, \mathbf{x}_j^{(C')}) \tag{4.4}$$

$$\text{Complete-Linkage: } fd(C, C') = \max_{i,j} fd(\mathbf{x}_i^{(C)}, \mathbf{x}_j^{(C')}) \tag{4.5}$$

$$\text{Average-Linkage: } fd(C, C') = \frac{1}{|C||C'|} \sum_{i,j} fd(\mathbf{x}_i^{(C)}, \mathbf{x}_j^{(C')}) \tag{4.6}$$

where data instances $\mathbf{x}_i^{(C)}$ and $\mathbf{x}_j^{(C')}$ belong to clusters C and C' respectively. These three techniques are based on different focuses: *single-linkage* relies on the nearest pair of data instances within two clusters, *complete-linkage* considers the furthest pair of data instances within two clusters, and *average-linkage* that balances the first two techniques. When being used to merge clusters, they have different effects: *single-linkage* is capable of finding elongated clusters while *complete-linkage* tends to find spherical clusters, and *average-linkage* can achieve a good compromise between them [35].

Unfortunately, all the above three techniques have the quadratic computational complexity because they need to consider every pair of data instances within two clusters. To reduce the computational burden, because each cluster is represented by a set of ROs

in our DIVA framework, we re-define the single-linkage, the complete-linkage and the average-linkage techniques as:

$$\text{Single-Linkage: } \widehat{fd}(C, C') = \min_{i,j} fd(\mathbf{ro}_i^{(C)}, \mathbf{ro}_j^{(C')}) \tag{4.7}$$

$$\text{Complete-Linkage: } \widehat{fd}(C, C') = \max_{i,j} fd(\mathbf{ro}_i^{(C)}, \mathbf{ro}_j^{(C')}) \tag{4.8}$$

$$\text{Average-Linkage: } \widehat{fd}(C, C') = \frac{1}{r^2} \sum_{i,j} fd(\mathbf{ro}_i^{(C)}, \mathbf{ro}_j^{(C')}) \tag{4.9}$$

where $\{\mathbf{ro}_i^{(C)}\}$ are the set of ROs for cluster C and $\{\mathbf{ro}_j^{(C')}\}$ are the set of ROs for cluster C'. When the RO sets well represent the distribution of their respective clusters, the RO-based Equations 4.7, 4.8 and 4.9 approximate Equations 4.4, 4.5 and 4.6 respectively but reduce their complexity to $O(r^2)$.

Given the above techniques of evaluating the proximity between clusters, we build a dendrogram T in a bottom-up fashion using the Hierarchical Agglomerative Clustering algorithm. The micro-clusters $\{C_k\}$ obtained from the divisive step constitute the leaf nodes of the dendrogram. In each iteration, the closest pairwise clusters (sub-nodes) are merged to form a new super-cluster (parent node). Without loss of generality, we assume that the super node t_p is formed based on two sub-nodes t_k and $t_{k'}$, then the top-r maximum-spread ROs in $\{\mathbf{ro}_i^{(t_k)}\} \cup \{\mathbf{ro}_j^{(t_{k'})}\}$ are chosen as the ROs for t_p.

The agglomerative step is summarized in Algorithm 6. It is worth noting that constructing the hierarchy in this step is not a reverse reproduction of the divisive step. Because in the divisive step we use the intra-cluster homogeneity as the criterion to partition the clusters, while in the agglomerative step we consider the inter-cluster homogeneity given the clusters are internally homogeneous. It has been proven that the agglomeration can remedy the inaccurate partitioning derived by the divisive step, as shown in BIRCH [162]. But in comparison with our DIVA algorithm, BIRCH is only capable to process propositional datasets because it depends on the linear and squared sum of data vectors, which are not available within relational datasets.

Algorithm 6: AggStep

Input: micro-clusters $\{C_k\}$
Output: dendrogram T

begin
  `// Initialize the dendrogram` T
  foreach C_k **do**
    Construct a leaf node t_k in T;
  end
  repeat
    Compute the distance between nodes in T that have no parent node, among which the pair (assuming t_k and $t_{k'}$) with the smallest distance value are chosen;
    Construct node t_p as the parent node for both t_k and $t_{k'}$, which equals to create a new super-cluster C_p to merge C_l and $C_{l'}$;
    Determine the ROs for t_p;
    Store t_p into T;
  until $K-1$ *times, where* K *is the size of* $\{C_k\}$;
  return T.
end

After building the dendrogram T, we need to determine the appropriate level in T and use the corresponding nodes to construct the cluster result. A common strategy is to select the level at which the variance of each node equals or is greater than v. Alternatively, we can record the variance of newly generated node for each level, find the largest gap between variances of two neighboured levels and use the lower level as the basis to construct clusters [35]. If the number of clusters is pre-specified, we will select the level which contains the required number of nodes to construct clusters.

4.3 Complexity Analysis

We now analyze the computational complexities of the recursive and the incremental implementations of the divisive step, given the size of dataset D is N and the number of ROs for each cluster is r:

- Recursive division: In each iteration, assuming the number of instances belonging

to the cluster C_k is N_k, the computational complexity of determining ROs is $O(rN_k)$ (as explained in Section 3.1), and so the total complexity is $O(\sum_k rN_k) = O(rN)$. Then each cluster of which the variance exceeds v will be bi-partitioned by using the farthest pair of ROs as the absorbent objects and re-allocating all the instances in that cluster accordingly. Here the distances between pairs of instances are not required to be calculated again. Therefore, the total complexity is $O(rRN)$, where R is the number of the recursive iterations.

- Incremental division: All the data instances in D are sequentially scanned and the micro-clusters $\{C_k\}$ are created incrementally. In order to insert the data instance $\mathbf{x}$ into the cluster result, $\mathbf{x}$ will be compared with all the nodes on the search path of the search tree S, so the number of comparisons for $\mathbf{x}$ is $O(r \log K)$, assuming K is the number of micro-clusters in $\{C_k\}$. Therefore, the total complexity for processing all instances in D is $O(rN \log K)$. It is worth noting that the search tree S utilized during the incremental divisive step is always balanced while the division tree derived by the recursive implementation does not guarantee such a balanced structure, so R would be expected to be greater than $\log K$.

In summary, the computational complexities of partitioning D using either the recursive or the incremental implementation of the divisive step is linear in N, assuming that v is set appropriately to make sure $R \ll N$ and $K \ll N$ for them respectively.

For the agglomerative step, the computational complexity of building the dendrogram is $O(r^2 \cdot K^2)$, given K is the number of micro-clusters derived by either the recursive or the incremental division. The value of K is controlled by the variance threshold v: smaller value of v leads to the generation of more clusters. When $v \to 0$, the divisive step will generate many tiny clusters, each of which contains the RO only. In this extreme case, the agglomerative step will behave like the pure agglomerative approach RDBC with quadratic complexity. Nevertheless, by choosing moderate values for r and v to keep $rK \ll N$, the computational complexity of our algorithm would be linear to N. In Section 4.4 and Section 4.5, comprehensive experimental results are

analyzed to see how to determine appropriate values for r and v.

Generally speaking, the incremental division is more preferred because the whole dataset will be scanned only once, so it is unnecessary to pre-load the whole dataset into the memory. In contrast, each data instance might be accessed many times in the recursive division, resulting in expensive disk I/O operations if the whole dataset cannot be contained in the memory. Due to this reason, the incremental implementation is more suitable for processing very large datasets as well as data streams.

Next, we will conduct comprehensive experiments to evaluate the effectiveness and efficiency of our DIVA framework. After being tested upon some propositional datasets of sophisticated distributions in Section 4.4, our framework is compared with several well-known clustering algorithms upon relational datasets in Section 4.5.

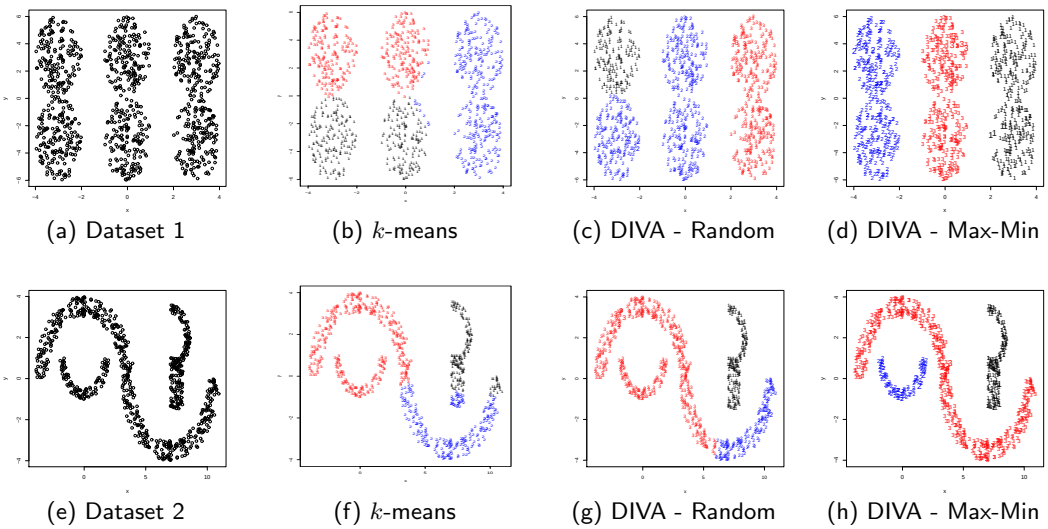
4.4 Experiments on Propositional Datasets

Although our DIVA framework is mainly developed for the purpose of clustering relational datasets, it can also be used upon propositional datasets, because the latter ones may be regarded as a special case of the former where all the data are of the single type and without relationships between them. In this section, we evaluate DIVA's performance in comparison with several classic propositional clustering algorithms using two datasets.

As explained in Section 2.1.3, if the class labels of data instances are available, we use an entropy-based measure [145] to evaluate the uniformity or purity of a cluster. Formally, the entropy $E(C_k)$ is defined by:

$$E(C_k) = \begin{cases} -\sum_h P_{h,k} \log_2 P_{h,k}, & \text{If each data instance has single class label.} \\ -\sum_h P_{h,k} \log_2 \frac{P_{h,k}}{Q_{h,k}}, & \text{If each data instance has multiple class labels.} \end{cases} \tag{4.10}$$

where $P_{h,k}$ and $Q_{h,k}$ are respectively the proportion of data instances of class h in cluster C_k. Then the total entropy is the sum of $E(C_k)$ over all clusters. Otherwise, the Intra-cluster Similarity can be used to measure the average similarity between pairs



(a) Dataset 1 (b) k-means (c) DIVA - Random (d) DIVA - Max-Min

(e) Dataset 2 (f) k-means (g) DIVA - Random (h) DIVA - Max-Min

Figure 4.1: Cluster Results Using Different Strategy

of data instances in the same cluster [35]:

$$S_{intra} = \frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{N_k^2}\sum_{x\in C_k}\sum_{x'\in C_k} fs(x,x')\right] \tag{4.11}$$

where N_k is the number of data instances in cluster C_k. Generally speaking, larger intra-cluster similarity and smaller entropy values indicate higher quality of cluster results.

4.4.1 Arbitrary-shaped Dataset

Propositional clustering has been thoroughly investigated over years and many algorithms, from the classical PAM, k-Means and HAC to the more recent BIRCH, DBScan and CLARANS, have been developed (as reviewed in Chapter 2). One great challenge in propositional clustering is how to effectively discover the data clusters of sophisticated shapes, which is especially important in spatial or geographical applications. Generally speaking, in these datasets the shape of the potential clusters might be spherical, elongated, curved or even arbitrary. Figures 4.1a and 4.1e are two examples of such sophisticated datasets.

Table 4.1: Execution Time of HAC, DBScan and DIVA (sec)

	k-means	HAC	DBScan	RecDiva	IncDiva
Dataset 1	0.320 $\pm$ 0.044	2.183 $\pm$ 0.038	0.995 $\pm$ 0.013	0.203 $\pm$ 0.011	0.507 $\pm$ 0.014
Dataset 2	0.315 $\pm$ 0.062	1.350 $\pm$ 0.025	0.700 $\pm$ 0.024	0.133 $\pm$ 0.005	0.373 $\pm$ 0.007

Figure 4.1 also provides the cluster results obtained by our DIVA algorithms, given the parameters $r = 4$ and $v = 0.9$ (that are the number of ROs for each cluster and the variance threshold respectively). Different digits with different colors in the figures represent the membership of data instances to different clusters. As shown in Figures 4.1c and 4.1g, the final cluster results were not correct when the RO sets were randomly selected. In contrast, the Max-Min strategy used by either the recursive or the incremental implementation of DIVA (denoted as "RecDiva" and "IncDiva" respectively) could discover the correct clusters in both datasets. Here we used the single-linkage strategy in the agglomerative step because it is generally better than the complete-linkage strategy to find the clusters with arbitrary shapes.

Furthermore, we compare our DIVA algorithm with k-means, HAC and DBScan [39] that has been implemented in the open-source Data Mining software *WEKA* [151]. All the algorithms were executed for 10 times with 10 different random seeds to shuffle the dataset. Unfortunately k-means cannot discover the correct clusters for both datasets, because it tends to generate spherical clusters. The results derived by k-means are shown in Figures 4.1b and 4.1f. On the other hands, the correct clusters are obtained when we utilize HAC, but DIVA is more scalable than HAC because micro-clusters instead of individual data instances are used as the leaf nodes when building dendrogram, which would greatly accelerate the clustering procedure. This conclusion is supported by the experimental results in Table 4.1. For DBScan, we tuned the parameter ϵ carefully and finally set as 0.13 and 0.07 for two datasets respectively, which allowed DBScan to find the correct clusters in the shortest time. The results in Table 4.1 indicate that DIVA spent substantially less time than DBScan did for both datasets.

Table 4.2: Experimental Results on UCI Chess Dataset

	k-means	HAC	DBScan	RecDiva	IncDiva
Entropy	62.93 $\pm$ 0.02	-	-	57.70 $\pm$ 0.81	61.77 $\pm$ 0.28
Time Spent	19.23 $\pm$ 8.14	-	374.68 $\pm$ 17.00	18.32 $\pm$ 0,98	24.72 $\pm$ 4.18

4.4.2 UCI Chess Dataset

Another propositional dataset comes from the UCI Machine Learning Repository [7]. We choose the chess (King-Rook vs. King) dataset [3], which has already been cited and studied in many papers. This dataset has 6 numeric attributes: white king file (column), white king rank (row), white rook file, white rook rank, black king file and black king rank. They were used as the input of out clustering algorithms. The attribute used to evaluate the cluster quality in our experiment is the depth-of-win for white in 0 to 16 moves or draw otherwise.

In total this dataset contains 28056 instances, for which the number of derived clusters is 18 (based on the cardinality of attribute depth-of-win). Like the experimental setup in the previous section, all the algorithms were executed for 10 times with 10 different random seeds to shuffle the dataset. We still compare our DIVA algorithms with k-means, HAC and DBScan. Unfortunately HAC does not work for this dataset because computing the similarity matrix between every pair of data instances, which in this case is composed of $28056^2 \approx 7.9 \times 10^8$ values, is infeasible. For DBScan, we found it is very sensitive to the user-specified parameters: When $\epsilon \leq 0.142$, DBScan generate lots of unit clusters, i.e. only one data instance in a cluster, which is not useful. On the other hand, all instances are assigned into the same clusters when $\epsilon \geq 0.143$, which is meaningless as well. In addition, DBScan spent too much time to process this dataset.

For DIVA, we set $r = 3$ and $v = 0.1$. The experimental results are presented in Table 4.2. Here RecDiva spent comparable time as k-means but IncDiva spent more time because it needs to create and maintain the search structure. RecDIVA generated better results than that of k-means with respect to the entropy criterion, which means

[3] http://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King)

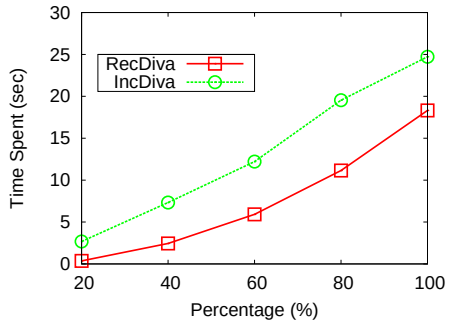


Figure 4.2: UCI Chess Dataset - Time Spent

the derived clusters are more accordant with the data classes.

We also tested the scalability of our DIVA algorithms by randomly sampling the whole dataset with a uniform distribution. The sampling percentage was ranged from 20% to 100% and repeated for 50 times. The experimental results are shown in Figure 4.2. We can see that time spent by RecDiva and IncDiva keep approximately linear growth with respect to the increase of data amount, which supports our complexity analysis in Section 4.3.

4.5 Experiments on Relational Datasets

We now compare RecDiva and IncDiva with the following relational clustering approaches: (1) FORC [71], which is the natural extension of k-Medoids in the field of relational clustering by utilizing the relational similarity measure. (2) ReCoM [145], which exploits the relationships among data instances by an iterative reinforcement fashion. In our experiments, k-Medoids is utilized as the meta clustering algorithm for ReCoM. (3) LinkClus [157], which develops a hierarchical structure, named *SimTree*, to represent the similarity values between pairwise data instances and hence facilitate the iterative clustering process. (4) RndDiva, which implements the DIVA framework but using the random strategy to select the RO set. Our experiments were conducted upon three datasets:

Figure 4.3: Ontology of the Synthetic Amazon dataset

4.5.1 Synthetic Amazon Dataset

This dataset simulates the users' visit to the Amazon website. It is generated by the following steps, given the data ontology as in Figure 4.3:

1. The product taxonomy was retrieved from Amazon's website *www.amazon.com*. It contained 11 first-level categories, 40 second-level categories and 409 third-level categories in the taxonomy. We generated 10,000 virtual products and randomly assigned them to the third-level categories. The category information was the only content feature defined for these products.

2. We randomly generated 2,000 users and uniformly distributed them into 100 groups. For each group, we constructed a probability distribution to simulate the users' preferences on the third-level categories (obtained in Step 1), which defined the likelihood of a user in that group to browse a product in a certain category. Each group of users had 4 interest categories: one major interest, two intermediate interests and one minor interest. Their corresponding probabilities were assigned as 0.5, 0.2 and 0.1 respectively.

3. Each browsing action of the users was generated according to the information of users, groups, categories and products: (i) randomly select a user and get his group; (ii) based on the probabilities of group interests, select an interest and get the related product category; (iii) randomly select a product that belongs to this category; (iv) create a browsing action between the user and the product obtained in step (i) and (iii). In total 100,000 browsing actions were created.

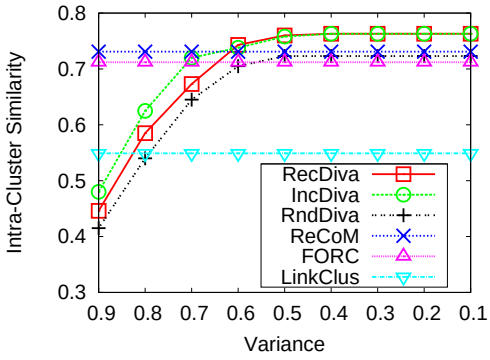
4. In order to test the robustness of different clustering approaches, we also generated some noise data: for each user, (i) uniformly choose four noise interests; (ii)

randomly select a product that belongs to one of the four noise interests of the current user; (iii) create a noise browsing action between the user and the product. We would examine how these clustering approaches perform in the case of different noise ratios, which is the percentage of noise data in the whole dataset.

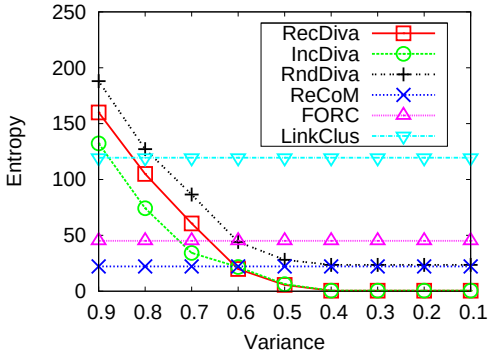
The users were assigned into 100 groups using different clustering algorithms. LinkClus was executed with a series of c values and the best cluster result was used in the comparison with other approaches. Since FORC, LinkClus and ReCoM launch an iterative clustering procedure, we set the maximum number of iterations to be 10, which appeared to be sufficient for these algorithms to converge in our experiments.

First of all, Figures 4.4a and 4.4b show the change of the cluster quality with respect to the parameter variance v in DIVA. We ranged v from 0.9 to 0.1 and fix $r = 3$. In general, the quality of the cluster result derived by DIVA improved as v increases. When $v \leq 0.6$, RecDiva and IncDiva outperformed all the other algorithms. The random strategy for RO selection, as we expected, could not derive good cluster result. We also found that the accuracy of LinkClus was far worse than those of the other approaches. The reason is that LinkClus builds the initial SimTrees by applying Frequent Pattern Mining, which only exploits the link structure of the relational dataset. The content features of the data instances, for example the category information for product, are completely ignored in the procedure of clustering. Due to such information loss LinkClus cannot generate clusters of high quality. Hence, we did not test RndDiva and LinkClus in the future experiments for brevity.

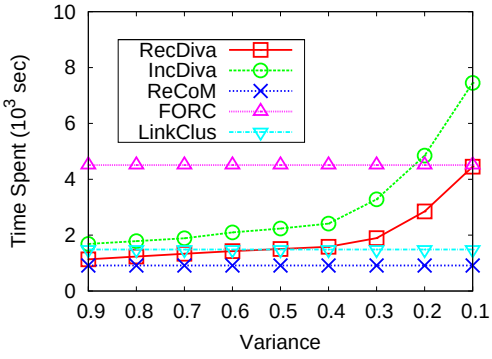
Figure 4.4c shows the execution time of the FORC algorithm is far more than that of the others. This result is not surprising since its computational complexity is $O(N^2)$. On the other hand, time spent by ReCoM, LinkClus and DIVA are comparable. Generally, as v decreases, DIVA will spend more time to generate smaller clusters in the divisive step and combine them again in the agglomerative step. When $v < 0.3$ in this case, time spent by DIVA sharply increased because many single-object clusters were produced. As discussed in Section 4.3, the use of such low variance threshold reduced



(a) Intra-cluster Similarity



(b) Entropy



(c) Time Spent

Figure 4.4: Synthetic Amazon Dataset - Clustering users w.r.t. different variance v

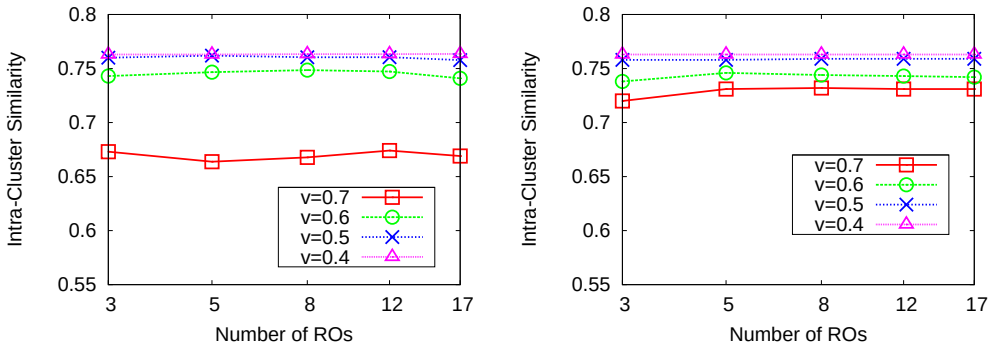
DIVA into RDBC with quadratic complexity. Another fact is that IncDiva generally spent more time than RecDiva did because it needed some extra effort to maintain the data structure S used for allocating new instances, and for small value of v it will be worse than FORC.

Next we examined the parameter r, i.e. the number of ROs for each cluster. r was varied from 3 to 17 increased by $\sqrt{2}$. Figures 4.5a to 4.5d show cluster results evaluated by the criteria of intra-cluster similarity and entropy. We found that these results are not sensitive to different values of r, because the binary split that only depends on the largest pair of ROs is used in our divisive step, as we claimed in Section 4.1. Figures 4.5e and 4.5f show that the running time grows while the accuracy are not substantially improved for both RecDiva and IncDiva. Therefore, a fairly small value of r is enough for providing high accuracy as well as keeping short running time for cluster analysis. Based on the above observation, we keep setting $r = 3$ in our following experiments.

Figure 4.6 illustrates the robustness of all approaches under different noise ratios of browsing actions, ranging from 20% to 100%. The parameters for DIVA were set as: $v = 0.5$ and $r = 3$. Generally, the accuracy of all approaches are reduced as the noise ratio increases. When evaluated by the criterion of intra-cluster similarity, the cluster results derived by RecDiva, IncDiva and ReCoM were comparable. When evaluated by the entropy-based criterion, both RecDiva and IncDiva outperformed ReCoM and FORC when the noise ratio was below 80% and their performance were very close when the noise ratio was above 80%. Here we think the entropy-based criterion is more meaningful because it is calculated only based on the class labels of the data instances while the computation of the intra-cluster similarity is impacted by the noise data.

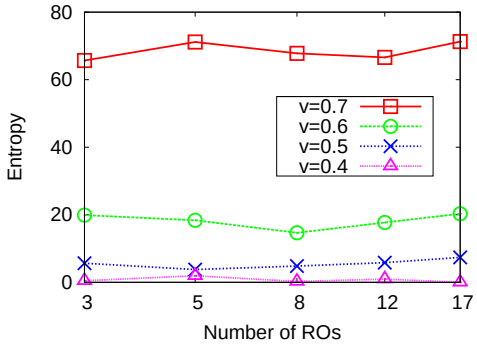
4.5.2 DBLP Dataset

This dataset was downloaded from the DBLP website, including authors, papers and venues (i.e. conferences and journals). The database schema is shown in Figure 4.7.

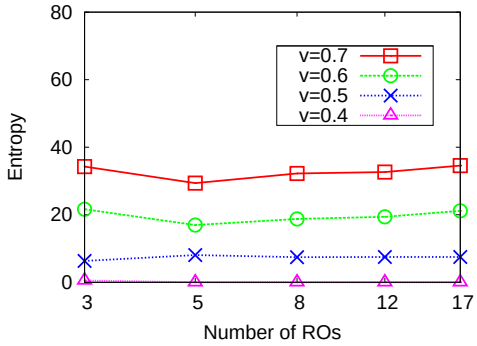


(a) RecDiva: Intra-cluster Similarity

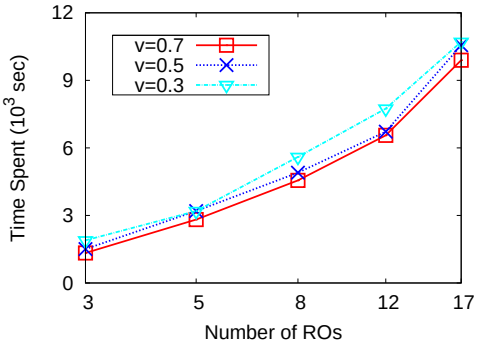
(b) IncDiva: Intra-cluster Similarity



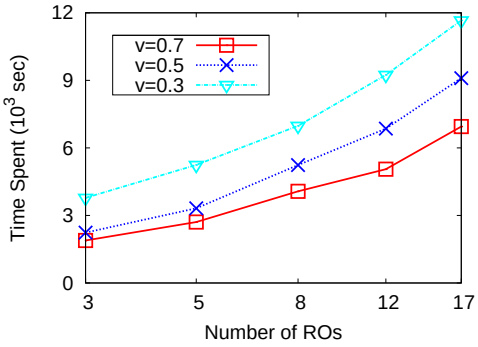
(c) RecDiva: Entropy



(d) IncDiva: Entropy

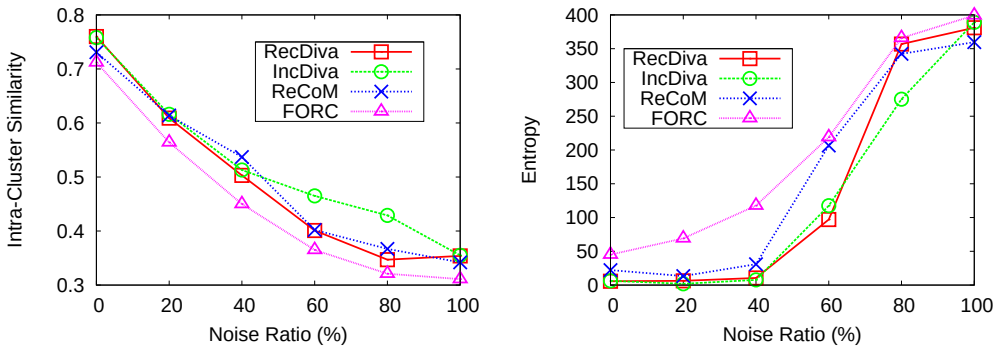


(e) RecDiva: Time Spent



(f) IncDiva: Time Spent

Figure 4.5: Synthetic Amazon Dataset - Clustering users w.r.t. different number of ROs



(a) Intra-cluster Similarity (b) Entropy

Figure 4.6: Synthetic Amazon Dataset - Clustering users w.r.t. different noise ratio

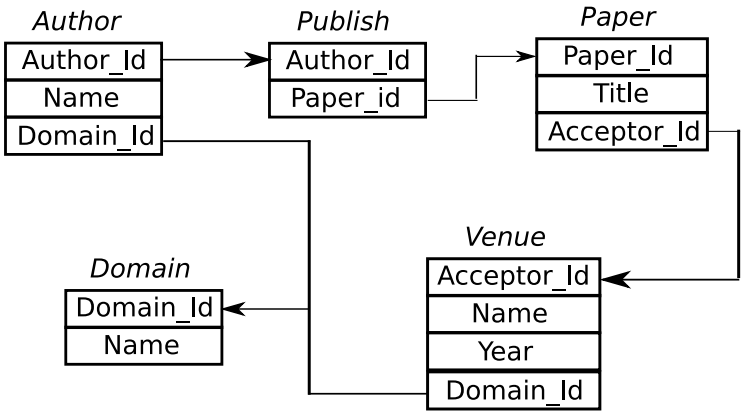


Figure 4.7: Schema of the DBLP Database

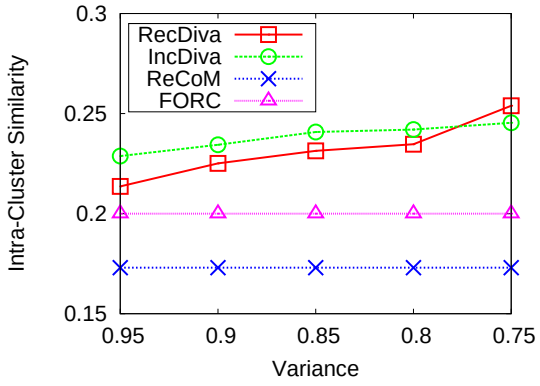
We selected 509 most productive authors and 718 well-known venues for our experiments, which were related to 94,217 papers. The total number of links between authors and papers is 2,941,276 and the number of papers published by an author individually or cooperatively ranges from 146 to 526. We can quantitatively evaluate the linkage sparseness of a dataset by using the ratio of the real and the potential numbers of links. For the DBLP dataset here, the ratio is $\frac{2941276}{509\times94217} = 0.0613$. In comparison, the linkage sparseness of the Synthetic Amazon dataset is $\frac{100000}{2000\times1000} = 0.005$, which mean the linkage structure in the DBLP dataset is more dense than that of the Synthetic Amazon dataset. Additionally the linkage structure is more complex here because the information of venues should also be considered in the clustering procedure. We manually assigned these authors into 15 research areas based on the information retrieved from their homepages, which were only used to evaluate the cluster quality. As each author might be related to more than one research field, the entropy measure based on the Kullback-Leibler Divergence was used to evaluate the relative purity of the clusters. We use ReCoM, FORC and DIVA to group the authors into 15 clusters.

Figure 4.8 reports our experimental results[4]. We can see in Figure 4.8a that both RecDiva and IncDiva outperformed FORC and ReCoM with regard to the intra-cluster similarity. Figure 4.8b shows that the cluster purity derived by RecDiva and IncDiva are better than that of FORC when $v < 0.9$. Furthermore, both DIVA implementations are more efficient than the other two algorithms, as shown in Figure 4.8c. Because the number of objects to be clustered in this dataset is far less than that of the Synthetic Amazon dataset, IncDiva spent comparable time as RecDiva due to the lower cost of maintaining the SearchTree.

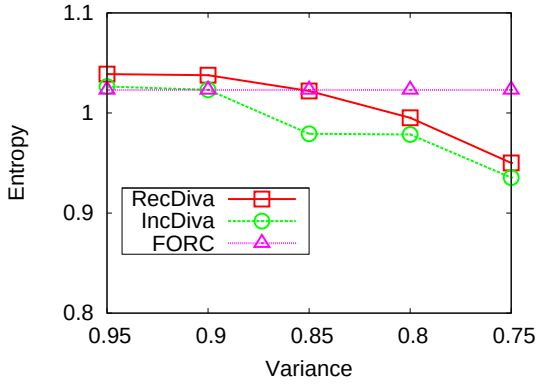
4.5.3 Movie Dataset

The clustering approaches were also evaluated on a real movie dataset defined by the ontology in Figure 2.3. This dataset was provided by a DVD retailer in UK. The movie

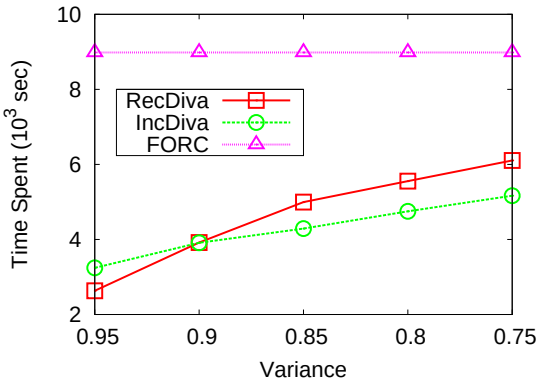
[4]Because the entropy of cluster result derived by ReCoM is 1.93 and the time spent is more than 10^4 seconds, they are not included in Figures 4.8b and 4.8c for brevity.



(a) Intra-cluster Similarity



(b) Entropy

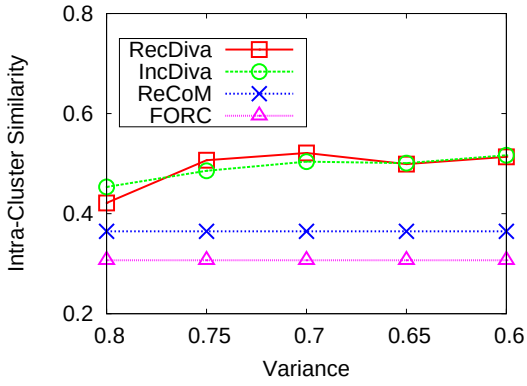


(c) Time Spent

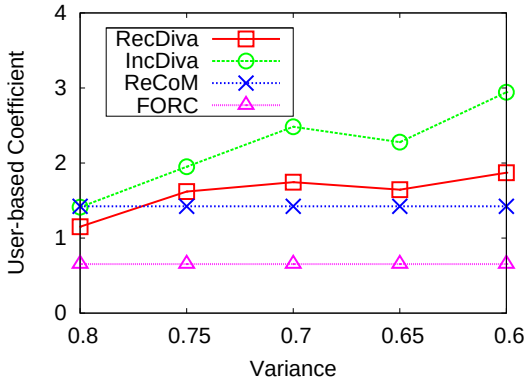
Figure 4.8: DBLP Dataset - Clustering authors w.r.t. different variance v

data were collected by spidering the retailer's website. All the rating data were derived from the web server log collected over a period of 3 months. After data pre-processing, there are 62,955 movies, 40,826 actors and 9,189 directors. The dataset also includes a genre taxonomy of 186 genres. Additionally, we have 542,738 browsing records included in 15,741 sessions from 10,151 users. The number of sessions made by different users ranges from 1 to 814. We use the 5,000 top-rated movies in our experiments for the cluster analysis.

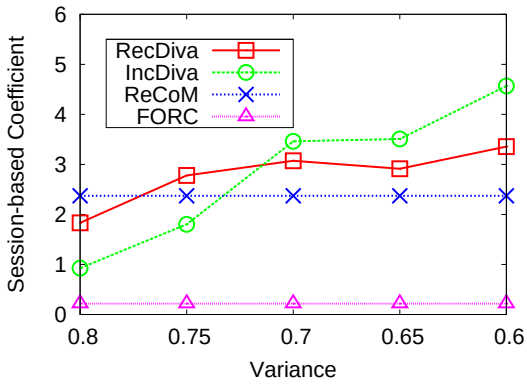
The entropy-based criterion can not be applied in this case, because no pre-specified or manually-labelled class information is available for movies. We have to utilize the visit information from users to evaluate the cluster results indirectly. The motivation behind this evaluation criterion stems from the research of recommender systems, which will be investigated thoroughly in Part III. Traditional collaborative filtering algorithms construct a user's profile based on all items he/she has browsed across sessions, then the profile of active user u_a is compared with those of other users to form u_a's neighbourhood and the items visited by the neighbourhoods but not by u_a are returned as the recommendation [77]. Hence, two items are "labelled" into the same category if they are co-browsed by the same user, which reflects the partitioning of the dataset from the viewpoint of users. Accordingly, we can construct the evaluation criterion as in [157]: two objects are said to be correctly clustered if they are co-browsed by at least one common user. The accuracy of clustering is defined as a variant of Jaccard Coefficient: the number of objects pairs that are correctly clustered over all possible object pairs in the same clusters. Another criterion is similar but of higher granularity, based on the assumption that a user seldom shifts his/her interest within a session, so two objects are said to be correctly clustered if they are included in at least one common session. Therefore we have two new criteria for evaluating the accuracy of clustering: user-based and session-based coefficient. As discussed in [157], higher accuracy tends to be achieved when the number of clusters increases, so we set to generate 100 clusters for all approaches.



(a) Intra-cluster Similarity



(b) User-based Coefficient



(c) Session-based Coefficient

Figure 4.9: Movie Dataset - Clustering movies w.r.t. different variance v

The evaluation results based on the intra-cluster similarity are shown in Figure 4.9a, in which both RecDIVA and IncDiva perform better than ReCoM and FORC. Figure 4.9b and 4.9c are the experimental results evaluated by the above coefficient criteria. RecDIVA and IncDiva still outperform ReCoM and FORC when $v < 0.7$, indicating the clusters generated by both of the DIVA implementations are more accordant with the user browsing patterns, i.e. the partitioning of the dataset derived by DIVA is more acceptable by users. It indicates that our DIVA implementations are more valuable than ReCoM or FORC when being applied to recommender systems. In Part II we will continue investigating the extension of our DIVA framework to generate hierarchical taxonomies with more details.

4.6 Summary

This chapter develops the multi-phase relational clustering framework *DIVA* based on the idea of Representative Objects. Our approach can obtain highly qualified cluster results and at the same time achieve high efficiency, because the divisive step partitions the dataset into micro-clusters of sufficient intra-cluster homogeneity, and the agglomerative step remedies the inaccurate partitions by considering inter-cluster homogeneity. The utilization of ROs facilitates the proximity computation for both the divisive step and the agglomerative step. Additionally we conduct comprehensive experiments covering propositional and relational datasets to evaluate our approaches. In the next part we will discuss an advanced topic in data mining, i.e. how to build and optimize the hierarchical taxonomy based on the cluster analysis in order to facilitate people and computers better exploiting the dataset.

Part II

Automated Taxonomy Generation

A taxonomy provides an efficient mechanism for people to navigate and browse a large amount of data by organising these data into a hierarchy of clusters [163] [83]. There are many examples of manually constructed taxonomies for describing the real world: In biology we have the taxonomy composed of seven levels (from general to specific): *kingdom*, *phylum*, *class*, *order*, *family*, *genus*, *species*. Each kind of animal on the earth is uniquely located in this taxonomy to indicate their evolutionary relationships to other animals. On the Internet, some taxonomies such as Yahoo! Directory (`http://dir.yahoo.com`) or Open Directory Project (`http://www.dmoz.org`) have been created to organize web-pages according to their content based on some global topical structures.

However, such manually maintained taxonomies are only feasible for small or static datasets. In contrast, Automated Taxonomy Generation (ATG) techniques have the ability to process huge and dynamic datasets and so has attracted considerable research interests in recent years. At present ATG techniques are mainly used to organize a large collection of documents [108] [85] [26] or web-pages/images [22] [86] [83] [23]. These applications usually apply Natural Language Processing or Information Retrieval techniques to extract a bag of words from the text as the linguistic features of the documents. Then all the documents are grouped based on these linguistic features using some hierarchical clustering techniques and a hierarchical taxonomy is created accordingly. Given the derived taxonomy, users can browse the whole document collection by following the taxonomic structure and then only focus on their interested topics. A successful application of the ATG technique on the Web is the search engine Clusty (`http://www.clusty.com`), which automatically groups the retrieved web-pages into a topic hierarchy as the response of user queries, so users can more easily find their interested articles.

We wish to extend these ATG approaches into more general scenarios of organizing relational datasets. The derived taxonomy is expected to reflect the internal distribution of the dataset as well as preserve important pairwise similarities between

data instances. Such generalized ATG approaches are especially valuable because they can not only facilitate people's navigation in the dataset but also improve the efficiency of many information systems. In Part III we will explore the application of an automatically derived taxonomy to improve the efficiency of recommender systems.

To the best of our knowledge the utility of ATG techniques upon relational datasets has not been investigated to any great extent until now. In order to address this issue, we propose an effective approach to automate the construction of taxonomies for representing and organizing relational datasets. Our ATG approach is based on the two-phase clustering framework DIVA proposed in Part I. The taxonomic structure is determined based on the homogeneity of the data instances belonging to the same taxonomic node as well as the heterogeneity of the data instances belonging to different nodes. The node labels are assigned by analyzing the divergence of the data attributes. We also propose a synthesized criterion to quantitatively evaluate the quality of the derived taxonomy by considering both the intra-node homogeneity and inter-node heterogeneity. In addition, we compare the robustness of our algorithm with others under different noise ratios, which is not studied in previous works.

The rest of this part is organized as follows: Some related research on ATG algorithms is reviewed in Chapter 5 together with the discussion of how to quantitatively evaluate the derived taxonomy. In the following chapters, we will focus on the utilization of the ATG techniques to analyze relational datasets. More specifically, in Chapter 6 we will develop the ATG algorithm based on the DIVA clustering framework and also discuss the issue of optimizing the taxonomic structure. Then in Chapter 7 we will explain our novel strategies of finding labels for the taxonomic nodes based on the Kullback-Leibler Divergence. Some experiments are conducted to evaluate the effectiveness of our approaches, and the experimental results together with the analysis are provided in Chapters 6 and 7 respectively.

Chapter 5

Review of Taxonomy Generation Algorithms

The Internet's open and decentralized nature encourages people to publish various textual and multimedia data to others, making the Internet as one of our major information sources today, but at the same time surfing on the web to retrieve useful information is not easy. To tackle this problem, there have been some attempts to create a static hierarchical categorization and then use it to organize webpages. For example, the Open Directory Project (`http://www.dmoz.org`) is the largest human-edited directory of the web maintained by more than 80,000 volunteer editors (up to Aug 2009). However, the ODP directory only covers less than 5% of the entire web. In addition, the categorical structure cannot be adjusted automatically to track the dynamic changes of the web and the cost of manually updating the categories is considerably high.

In recent years, Automated Taxonomy Generation (ATG) has gradually become an attractive research topic because it is very suitable for processing huge and dynamic datasets. Currently ATG techniques are mainly used to organize, represent and discover knowledge within a large collection of documents by means of generating the hierarchical categories. In this chapter, we will review some important issues relating to ATG algorithms.

5.1 Fundamentals of the ATG algorithms

The main purpose of building a taxonomy is "to provide a meaningful navigational and browsing mechanism by organizing large amount of information into a small number of hierarchical clusters" [163]. Kummamuru et al. suggested in [83] that the taxonomy built for a document collection should have six desirable properties. In order to process more general datasets, we reiterate these properties with appropriate extension as follows:

1. *Coverage*: Ideally each data instance should be contained in at least one node in the taxonomy. That is to say, for each instance, we can find a path in the taxonomy which starts from the root node and ends at the node that the instance belongs to. The derived taxonomy covering more data instances is considered to be better.

2. *Compactness*: This property restricts the size of the taxonomy. Since ATG techniques are used for the purpose of summarizing and navigating datasets, taxonomies with too many levels or too many branches in each level are not encouraged.

3. *Sibling Node Distinctiveness*: At any level of the hierarchy, the sibling nodes should be as different as possible from each other.

4. *Node Label Predictiveness*: Labels are used to summarize the contents or the characteristics of their corresponding nodes. A taxonomy with good node labels can help users to find data of their interests with minimum efforts.

5. *Reach Time*: The average time spent by users to find data of their interests is important. This criterion can be qualified by the size of the taxonomy, because both the depth and the width of the taxonomy will impact user's reach time.

6. *General-to-Specific Order*: The node labels should be selected to follow the general-to-specific relationship within the hierarchical taxonomy from top to bottom.

Krishnapuram and Kummamuru provided an overview in the field of automatic taxonomy generation and raised three questions [80]:

1. How to identify documents that have similar content;
2. How to discover the hierarchical structure of the topics and subtopics;
3. How to find appropriate labels for each of the topics and subtopics;

We will examine each of these three issues below.

The document similarities are mainly computed based on the keywords occurring in the documents. Usually, techniques in the fields of Natural Language Processing (NLP), Text Mining or Information Retrieval (IR) are utilized as the first step to extract some linguistic features from the content of these documents. Then document i is represented as the feature vector $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$, where $x_{i,j}$ is the weight of term w_j in the document i. There are many ways of calculating the value of x_i, among which the TF-IDF measure [9] is the most widely used one because it can effectively evaluate the importance of a word to a document by considering the term frequency (TF) as well as the inversed document frequency (IDF), where IDF is used to remove the bias of the common words in the corpus. Based on the above feature vector model, the similarity measure between two documents can be computed by the cosine similarity value or the Pearson's Correlation Coefficient of their corresponding feature vectors [9]. In addition, our linguistic knowledge can be exploited to facilitate the syntactic and semantic analysis on the documents.

Given the vector space model of the documents, traditional divisive or agglomerative hierarchical clustering algorithms can be used to build the hierarchical taxonomy. Steinbach et al. investigated the possibility of using the bisecting k-means clustering algorithm, a variant of standard k-means, to analyze the document collection [138]. The bisecting k-means algorithm selects a cluster (based on some criteria such as the largest size or the least overall similarity) and splits it into exactly two sub-clusters by using the standard k-means. The algorithm continues iteratively until K clusters are obtained.

It has been shown that the bisecting k-means often outperforms the standard k-means and the agglomerative clustering algorithms. Boley developed a divisive partitioning algorithm PDDP in [18], which conducts binary divisions in each cluster iteratively until the scatter value of the derived clusters are less than a user-defined threshold. Here the scatter value is calculated by a linear discriminant function which is dependent on the first principal component of the covariance matrix. In contrast, Cheng and Chien extended the agglomerative clustering algorithm in [22] to generate topic hierarchies. To make the hierarchy more feasible for browsing, they examined the intra-cluster and the inter-cluster distances at each level in the hierarchy and then merged those levels of which the clusters are not cohesive. The application of hierarchical clustering to build query taxonomies can also be found in [25]. In case of semi-automated taxonomy generation where suggestions from users are available, Kashyap et al. used a series of thresholds to select nodes and extract a taxonomy from the dendrogram [69].

As opposed to the above approaches that use the dendrogram as the basis to generate taxonomies, some other approaches directly build the taxonomies from scratch. Clerkin et al. [27] applied the incremental conceptual clustering algorithm COBWEB [43] to construct the class hierarchies. COBWEB conducts a hill-climbing search in the space of possible taxonomies and uses *Category Utility* [50] to select the best categorizations. The algorithm is performed in an incremental fashion. When classifying a new instance, the algorithm evaluates the CU of the derived taxonomy and then performs one of the following operations: classify the instance into the existing cluster, create a new cluster, divide or merge the existing clusters. Finally the output of COBWEB is translated into a class hierarchy. Lawrie and Croft [86] proposed the use of a set of topical summary terms for taxonomy construction. These topical terms are selected by maximizing the joint probability of their topicality and predictiveness, which is estimated by the statistical language models of the document collection. Kummamuru et al. [83] developed an incremental learning algorithm DisCover to maximize the coverage as well as the distinctiveness of the taxonomy. The criterion used to pick out concepts

is the weighted combination of two functions reflecting document coverage and sibling distinctiveness respectively.

Furthermore, Wnek proposed to use the Latent Semantic Indexing (LSI) techniques to construct taxonomies. The introduction of LSI effectively reduced the mismatching of words and thus improved the quality of the derived taxonomy [152]. Sie and Yeh combined the human-provided knowledge network and ATG techniques to generate a semantic hierarchy in the digital library environment. They exploited the meta-data schema of the digital library and considered the path of the knowledge network in the similarity computation, so the derived hierarchy is more human-readable and semantic-oriented [136]. Kashyap provided an experimentation framework for automated taxonomy construction in [69]. They evaluated some of the NLP and clustering techniques introduced above and identified their parameters. However, none of the above approaches can easily be extended to process relational datasets. For example, PDDP requires the availability of covariance matrix and probabilistic models beyond propositional domains, which are usually undefined in the relational datasets.

The next issue is to assign appropriate labels for the derived taxonomic nodes. If the taxonomy is built for document collections, the keywords or phrases extracted by the NLP/IR techniques were usually used as the node labels [25][22]. Krishnapuram and Kummamuru suggested to use a set of words with a high frequency of occurrence in the nodes as labels [80]. In the algorithm DisCover [83], meaningful nouns, adjectives or noun phrases (with necessary pre-processing such as stemming, stop-word elimination or morphological generalization) extracted from the documents were used as the labels. Boley proposed the use of the "most significant words" as the node labels. These words should be the most distinctive ones to the current cluster or distinguish the cluster from its neighbours, and so they provide good indicators of the cluster topics. The author used the cosine value between the principal direction vectors (computed by the PDDP algorithm) of the current node and its parent to select the most significant words. In [69], the authors proposed to assign some potential labels for the taxonomic nodes and

then use the noun phrase replacement and the label propagation techniques to refine the taxonomy labels. Nevertheless, how to find labels for the taxonomic nodes built for relational datasets has not been investigated to date.

5.2 Evaluation Criteria

Another non-trivial question is how to evaluate the quality of the derived taxonomy. Kashyap et al. classified the criteria as the categories of evaluating the taxonomic contents or evaluating the taxonomic structures [69]. The former measures the overlap of the assigned labels while the latter measures the structural validity of these labels. Both types of criteria assume the availability of the standard taxonomy, so the precision and recall of the assigned labels can be compared with the standard taxonomy. Chuang and Chien applied the F-measure that combines precision and recall as the evaluation criterion for their derived taxonomy, because the class information of the data were also available in their experiments.

Krishnapuram and Kummamuru in [80] summarized three scenarios when evaluating the derived taxonomies:

1. When a standard taxonomy with ground truth is available, we can compare the derived taxonomy with it. Two commonly used criteria are *accuracy* and *mutual information*. The accuracy can be computed by assigning a class label for each node in the standard taxonomy and then counting how many data instances in the derived taxonomy are correctly clustered. In contrast, mutual information [98] considers the probability distribution of all the class/cluster labels within the standard and derived taxonomies, and then computes their mutual dependence as follows:

$$MI(\mathcal{K}, \mathcal{K}') = \sum_{k_i \in \mathcal{K}} \sum_{k'_j \in \mathcal{K}'} p(k_i, k'_j) \log \frac{p(k_i, k'_j)}{p(k_i)p(k'_j)} \tag{5.1}$$

   where $\mathcal{K}$ and $\mathcal{K}'$ are the set of class-labels in the standard taxonomy and the set of cluster-labels in the derived taxonomy respectively, $p(k_i)$ and $p(k_i)$ are the

marginal probability distribution of class- and cluster-labels respectively, $p(k_i, k'_j)$ is their joint probability distribution.

2. When the ground truth is not available, one possible way to evaluate the derived taxonomy is to see how well the assigned labels in the taxonomy can predict the content of the documents in each cluster, which can be measured by the *Expected Mutual Information Measure* (EMIM) [85]. Given the set of label words as $\mathcal{W}$ and the set of meaningful words in the document collections as $\mathcal{W}'$, EMIM is computed as follows:

$$EMIM(\mathcal{W}, \mathcal{W}') = \sum_{w \in \mathcal{W}} \sum_{w' \in \mathcal{W}'} p(w, w') \log \frac{p(w, w')}{p(w)p(w')} \tag{5.2}$$

   Another useful criterion is *reachability*, which is determined by the percentage of documents that are covered by the derived taxonomy and the total number of nodes we need to navigate before finding all the relevant documents.

3. When comparing two taxonomies that are built by different algorithms, we can count the total number of parent-child pairs in common between these two taxonomies or compute the edit distance between them.

In this thesis we focus on analyzing unlabelled datasets. Since there is no ground truth available for us to build the standard taxonomy, we examine our derived taxonomy with respect to six properties reviewed in Section 5.1 and evaluate the labels of our derived taxonomy using the reachability criterion introduced above. More specifically, the procedure of user navigation will be simulated and hence the quality of the derived taxonomy be evaluated based on the success ratio and the path length that relevant data instances are discovered. More details can be found in Chapters 6 and 7.

Chapter 6

Taxonomy Generation for Relational Datasets

In this chapter we will present our general approach of automatically generating taxonomies as well as evaluating the derived taxonomy based on the desired properties introduced in Chapter 5. In essence, taxonomy generation can be understood from the viewpoint of artificial intelligence as a search through a hypothesis space composed of all possible hierarchies upon the given dataset, and so in Section 6.1 the search model is briefly explained. Then our ATG approaches are discussed in Section 6.2 and their complexities are analyzed in Section 6.3. Finally an empirical evaluation on the proposed methods is presented in Section 6.4 to show the effectiveness of our approaches.

6.1 Search Model for Learning

From the viewpoint of artificial intelligence, many learning problems can be generalized as a *search* through a hypothesis space [104]. For example the algorithm ID3 [119] is regarded as searching in the space of possible decision trees for the best one that fits the training examples. ID3 performs a hill-climbing search through this space in a general-to-special order: From an empty tree, ID3 progressively elaborates the hypothesis by

appending more constraints into the tree, which is equivalent to moving from one node to another in the hypothesis space, until all the examples are correctly classified. In each step the criterion guiding the search is the information gain measure [103].

There are a few dimensions to be considered when a learning problem is characterized as a search [33]. One of them is the *control strategy* which means the selection between the exhaustive strategies (like depth-first or width-first search) and the heuristic ones (like hill-climbing or beam-search). A second dimension, the *search direction*, means that the nodes in the hypothesis space are ordered by their generality, and so a learner may use either the generalization or the specialization operation to search the space. Another important dimension is the *search criterion*, which determines whether the current hypothesis fits the data and (if not) which candidate hypothesis should replace the current one in the next iteration of search.

By viewing learning as a search problems, we can easily examine and compare different strategies for searching the hypothesis space, especially those algorithms that are efficiently searching very large or infinite hypothesis spaces, to find the learning model that best fits the given data [103].

6.2 Taxonomy Construction

The agglomerative step developed in Section 4.2 uses all the micro-clusters generated by the divisive step as the leaf nodes of the dendrogram and iteratively merges them using either of the three strategies: single-linkage, complete-linkage or average-linkage. The obtained dendrogram T is a binary-split hierarchical structure in which each super-node contains all the data instances belonging to its sub-nodes. Let us evaluate the agglomerative step using six desired properties introduced in Section 5.1. Since each data instance is contained in one of the leaf nodes, Property *Coverage* is always satisfied. Properties *Node Label Predictiveness* and *General-to-Specific Order* are about the labels of the taxonomic nodes. They are originally proposed for document collections, in which the nodes are labeled by parsing the textual content of the documents. Because

it is difficult to evaluate the goodness of labels for summarizing taxonomic nodes, labeling taxonomic nodes is essentially very subjective. We postpone the discussion about labeling taxonomy until the next chapter. Therefore, we put our focus on optimizing the hierarchical structure of the derived taxonomy to satisfy Properties *Compactness*, *Sibling Node Distinctiveness* and *Reach Time* in this section together with the analysis of comprehensive experimental results to evaluate our algorithm. Obvious, the binary-split structure of the dendrogram T does not satisfy the requirement of Property *Compactness*. We need to adjust the structure of T by reducing the total number of levels in it. Moreover, we expect the optimized hierarchy can best reflect the natural distribution of the dataset, i.e. all the taxonomic nodes located at the same level are approximately of the same granularity and all the sibling nodes belonging to the same parent node are as distinctive as possible.

Some heuristic algorithms have been developed for this purpose. Chuang and Chien developed a top-down algorithm to merge the dendrogram structure for organizing the query terms. The determination of a cutting level depends on the quality of the cluster set at that level, which is the product of intra- and inter-cluster similarities as well as the size of the cluster set [25]. This algorithm is not suitable for other datasets because it is sensitive to the perceptual terms in the query. Later, Cheng and Chien used the distance between two adjacent levels, which is the ratio of the change of intra-cluster distance to that of inter-cluster distance in two levels, as the cutting criterion in [22], but the merging operation after determining the cutting levels is not clear and the authors did not conduct experiments to compare their algorithm with other ATG approaches. In addition, these algorithms are not efficient in processing the relational datasets due to their high computational complexity in computing pairwise data similarities.

In this section we will investigate the use of our RO-based DIVA clustering framework to achieve high efficiency gains for the taxonomy generation. As explained in Section 6.1, the problem of optimizing the taxonomy can be characterized as a search in the hypothesis space that is composed of possible taxonomic structures upon the given

dataset. It is easy to see that such hypothesis space is complete because every possible taxonomy can be represented by a node in the hypothesis space. Now we consider three dimensions of the search: control strategy, search direction and search criterion. The basic idea of our approach is to find a set of cutting levels in the dendrogram of which the corresponding clusters are qualified and other unqualified clusters are removed accordingly. The quality will be evaluated using some measures based on cluster homogeneity. As the algorithms proposed in [25][22], our approach is in essence a heuristic search since the hypothesis space is too large for the exhaustive search to be performed. The search begins from the initial node, i.e. the dendrogram of which the hierarchical structure is binary split and each leaf node represents a micro-cluster derived by the clustering procedure. Then in each step some taxonomic nodes are merged, corresponding to a move from a node to another in the hypothesis space. Based on the criterion of identifying the cutting levels and the strategy of merging nodes, we develop two algorithms for optimizing the taxonomic structure, explained in the following sections respectively.

6.2.1 Global-Cut

For a dendrogram built by the HAC algorithm, Duda et al. suggested that the node-pair similarity used in each agglomerative iteration indicates whether the formed clustering is natural or forced: an unusually large gap within a series of node-pair similarities means a natural partitioning of the dataset. They applied this idea to obtain the flat cluster result from a dendrogram [35]. However, this is only a "rule of thumb" because they did not provide any formula to quantitatively measure the similarity gap.

We extend their idea to construct the taxonomy based on a dendrogram. Our approach, named "Global-Cut", iteratively identifies multiple layers of natural partitions in the dataset and organizes them as a taxonomy. First of all, we formally define the intra-node distance $g_{intra}(l)$ and inter-node distance $g_{inter}(l)$ as the basis for the cutting criterion. Since each node is represented by a set of ROs, $g_{intra}(l)$ and $g_{inter}(l)$ can be obtained with low computational expense: Assuming all the levels in the dendrogram T

are numbered as $1, 2, \ldots, L$ from top to bottom, we have:

$$g_{intra}(l) = \frac{1}{|\{t_k^{(l)}\}|} \sum_k var(t_k^{(l)}) \tag{6.1}$$

where $\{t_k^{(l)}\}$ are all nodes located at the l-th level of T and

$$g_{inter}(l) = \frac{2}{|\{t_k^{(l)}\}| \times (|\{t_k^{(l)}\}| - 1)} \sum_{k \neq k'} fd_{node}(t_k^{(l)}, t_{k'}^{(l)}) \tag{6.2}$$

The function $|\cdot|$ means the cardinality of the set. One special case is the calculation of $g_{inter}(1)$, because the first level only contains the root node and thus its inter-node distance is undefined. To solve this problem we simply set $g_{inter}(1) = 0$.

The changes of intra- and inter-node distances between two neighbouring levels are combined in:

$$\begin{aligned} G(l) &= \Delta g_{intra}(l) - \Delta g_{inter}(l) \\ &= \Big[g_{intra}(l-1) - g_{intra}(l)\Big] - \Big[g_{inter}(l-1) - g_{inter}(l)\Big] \end{aligned} \tag{6.3}$$

where $l \geq 2$. Thus, we develop our rule to identify the global cutting levels:

Rule 6.1. *Level l is regarded as a cutting level only when $G(l) > 0$.*

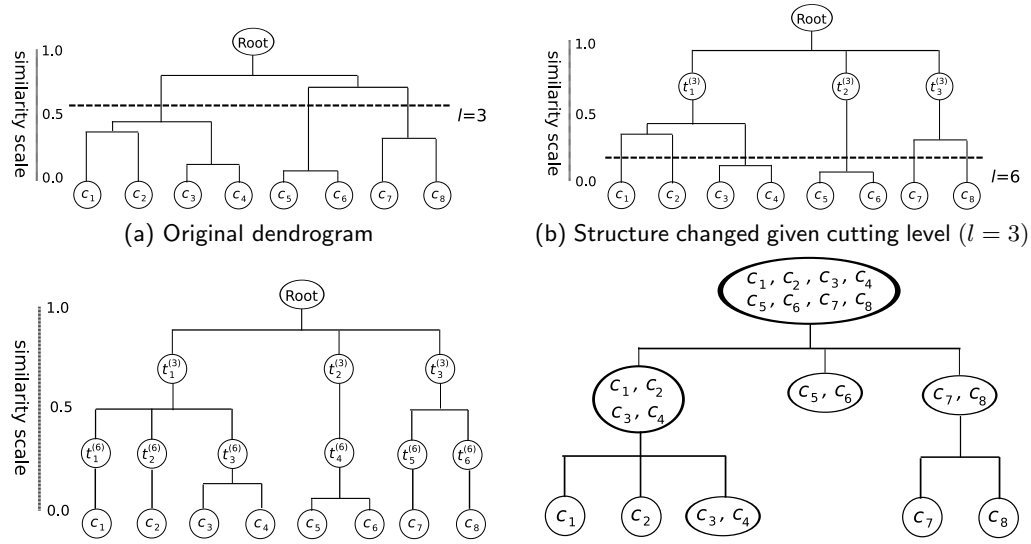
Here is an intuitive explanation for this criterion: if the change of intra-cluster distance between level $l-1$ and l is greater than the change of inter-cluster distance between them, which means the data grouped by the nodes of level l is more cohesive or "natural" than that of level $l-1$, it is reasonable to set a cutting level between $l-1$ and l (which is equivalent to preserve the nodes at level l into the final taxonomy); otherwise, levels $l-1$ and l would be merged.

The cutting criterion defined in Equation 6.3 is utilized iteratively: in each iteration the whole dendrogram is scanned to identify one cutting level. Then the hierarchical structure is re-organized by merging all the nodes between the previously and the cur-

Algorithm 7: Iterative Optimize the Taxonomic Structure

Input: Dendrogram T
Output: Optimized taxonomy T'

begin
  Retrieve the root node, calculate $g_{intra}(1)$;
  $g_{inter}(1) \leftarrow 0$;
  for $l \leftarrow 2$ **to** $l \leq L$ **do**
    Retrieve all nodes in levels l, denoted as $\{t_k^{(l)}\}$ respectively;
    Calculate $g_{intra}(l)$ and $g_{inter}(l)$;
    Calculate $G(l)$ using Equation 6.3;
    if $G(l) > 0$ **then**
      `// Set a cutting level between` $l-1$ `and` l`;`
      Store $\{t_k^{(l)}\}$ as a natural partition of the dataset;
      CONTINUE the **For** loop;
    else
      `// Merge the nodes at level` l`;`
      Remove $\{t_k^{(l)}\}$ from T;
      Link all sub-nodes of $t_k^{(l)}$ to $t_k^{(l)}$'s parent directly;
    end
  end
  Organize the remaining nodes in T according to their linkage relationship to obtain the optimized taxonomy T';
end



(a) Original dendrogram

(b) Structure changed given cutting level ($l = 3$)

(c) Structure changed given cutting level ($l = 6$)

(d) Final taxonomy

Figure 6.1: Example of generating taxonomy

rently identified cutting levels: when a super-node t_p and its sub-node $\{t_k\}$ are together located between two neighboring cutting levels or below the lowest cutting level, all t_k nodes will be removed and their sub-nodes will be linked to t_p directly. The above iteration continues until no more cutting levels can be found in the dendrogram. The whole procedure is summarized in Algorithm 7.

A simple example is given here to demonstrate our algorithm: Figure 6.1a is a binary dendrogram T built by the agglomerative step of the DIVA algorithm, which is the starting point of our approach for taxonomy generation (for the sake of brevity, we do not draw any non-leaf nodes in this graph). In the first iteration, we found a cutting level at $l = 3$ where $G(3) > 0$. Therefore the node set $\hat{L}_3 = \{t_1^{(3)}, t_2^{(3)}, t_3^{(3)}\}$ is regarded as a natural partition upon the dataset. The dendrogram structure is changed accordingly by removing the right subnode of the root that is above the cutting level, resulting in a new hierarchy shown in Figure 6.1b. In the second iteration, we found another cutting level at $l = 6$ where $G(6) > 0$, which means another natural partition composed of nodes $\hat{L}_6 = \{t_1^{(6)}, t_2^{(6)}, t_3^{(6)}, t_4^{(6)}, t_5^{(6)}, t_6^{(6)}\}$, and so we preserve these nodes and remove other nodes between the node sets $\hat{L}_3$ and $\hat{L}_6$, as in Figure 6.1c. Finally,

since no more cutting levels can be identified, the remaining nodes are removed from the hierarchy. In addition, node $t_4^{(6)}$ is also removed because it contains the same content as node $t_2^{(3)}$. The final taxonomy T' is shown in Figure 6.1d.

6.2.2 Local-Cut

The Global-Cut approach explained previously identifies the cutting-levels by considering all the sub-trees in the dendrogram. Another approach, named "Local-Cut", identifies the cutting levels in a recursive fashion for each of the sub-trees. Hence, we re-define the intra- and the inter-node distances as follows:

$$\hat{g}_{intra}(t_k) = var(t_k) \tag{6.4}$$

and

$$\hat{g}_{inter}(t_k) = \frac{1}{|\{t_{k'}\}|} \sum_{t_{k'}} fd_{node}(t_k, t_{k'}) \tag{6.5}$$

where the set $\{t_{k'}\}$ are t_k's sibling nodes. Similar to the Equation 6.3, we combine the changes of intra- and inter-node distances in:

$$\begin{aligned} \hat{G}(t_k) &= \Delta\hat{g}_{intra}(t_k) - \Delta\hat{g}_{inter}(t_k) \\ &= \left[\hat{g}_{intra}(t_k) - \frac{1}{|\{t_s\}|}\sum_{t_s}\hat{g}_{intra}(t_s)\right] \\ &\quad - \left[\frac{1}{|\{t_{k'}\}|}\sum_{t_{k'}}\hat{g}_{inter}(t_{k'}) - \frac{1}{|\{t_s\}|}\sum_{t_s}\hat{g}_{inter}(t_s)\right] \end{aligned} \tag{6.6}$$

Then whether node t_k should be split into a set of sub-nodes $\{t_s\}$ depends on the following rule:

Rule 6.2. *Taxonomic node t_k should be split into a set of sub-nodes $\{t_s\}$ only when $\hat{G}(t_k) > 0$.*

Here is an intuitive explanation: when $\hat{G}(t_k) > 0$, which means the split of node

t_k is natural to reflect the distribution of all the data instances belong to t_k, and so t_k's sub-nodes $\{t_s\}$ will be preserved in the resulting taxonomy. Otherwise, $\{t_s\}$ will be removed and their sub-nodes are linked to t_k directly and t_k will be re-examined. The above procedure is summarized in Algorithm 8. We expect such an optimization strategy can generate taxonomies of higher quality than Global-Cut. The experimental results discussed in Section 6.4 prove our assumption.

In addition, we can extend the Global-Cut and the Local-Cut formula by combining the intra-node and the inter-node distances with different coefficients, as follows:

$$G'(l) = \alpha\Big[g_{intra}(l-1) - g_{intra}(l)\Big] - (1-\alpha)\Big[g_{inter}(l-1) - g_{inter}(l)\Big] \tag{6.7}$$

and

$$\begin{aligned}\hat{G}'(t_k) &= \alpha\left[\hat{g}_{intra}(t_k) - \frac{1}{|\{t_s\}|}\sum_{t_s}\hat{g}_{intra}(t_s)\right] \\ &\quad -(1-\alpha)\left[\frac{1}{|\{t_{k'}\}|}\sum_{t_{k'}}\hat{g}_{inter}(t_{k'}) - \frac{1}{|\{t_s\}|}\sum_{t_s}\hat{g}_{inter}(t_s)\right]\end{aligned} \tag{6.8}$$

where the user-tuned parameter α $(0 \leq \alpha \leq 1)$ is used to adjust the relative importance of the changes of the intra-node and the inter-node distances to the decision of merging subtrees. In the following, for the sake of simplicity we do not differentiate the importance between them, which means $\alpha = 0.5$.

6.3 Complexity Analysis

Compared with traditional ATG approaches, our approach is far less expensive to determine the cutting levels.

Since each node in T is represented by a $\mathbf{ro}_i$ $(1 \leq i \leq r)$, the complexities of calculating $g_{intra}(l)$ and $g_{inter}(l)$ are not dependent on the size of the whole dataset D: Given the number of nodes $\{t_k\}$ that are located at level l is K_l, to compute

Algorithm 8: Recursively Optimize the Taxonomic Structure

Input: Dendrogram T
Output: Optimized taxonomy T'

begin
  Initialize the processing queue Q_1 and the finished queue Q_2;
  $Q_1 \leftarrow$ root node of T;
  while $Q_1 \neq \emptyset$ **do**
    $t_k \leftarrow \texttt{GetHead}(Q_1)$;
    $Q_1 \leftarrow Q_1 - \{t_k\}$;
    if t_k *is leaf* **then**
      $Q_2 \leftarrow Q_2 \cup \{t_k\}$;
      CONTINUE the **while** loop;
    else
      Retrieve t_k's sub-nodes $\{t_s\}$;
    end
    Retrieve t_k's sibling nodes $\{t_{k'}\}$;
    Calculate $\hat{g}_{intra}(t_k)$, $\hat{g}_{intra}(t_s)$, $\hat{g}_{inter}(t_k)$ and $\hat{g}_{inter}(t_s)$;
    Calculate $\hat{G}(t_k)$ using Equation 6.6;
    if $\hat{G}(t_k) > 0$ **then**
      `// Set a cutting level between` t_k `and` $\{t_s\}$`;`
      $Q_2 \leftarrow Q_2 \cup \{t_k\}$;
      $Q_1 \leftarrow Q_1 \cup \{t_s\}$;
    else
      `// Merge the nodes` $\{t_s\}$`;`
      Remove $\{t_s\}$ from T;
      Link t_s's sub-nodes to t_k directly;
      Insert t_k at the head of Q_1 again;
    end
  end
  Organize all the nodes in Q_2 according to their linkage relationship to obtain the optimized taxonomy T';
end

$g_{intra}(l)$ we will compare each pair of ROs belonging to the same node, and so the total number of comparison is $O(r^2 \cdot K_l)$. Similarly the complexity of computing $g_{inter}(l)$ is $O\left(r^2 \cdot \left(\frac{K_l}{K_{l-1}}\right)^2\right)$ in a balanced hierarchy or $O\left(r^2 \cdot K_l^2\right)$ in the worst case. When the number of nodes K_l is far less than the size of D, the time spent by the optimization step of our approach is approximately constant. For our recursive optimization strategy, the above discussion is also suitable for the complexity analysis of computing $\hat{g}_{intra}(t_k)$ and $\hat{g}_{inter}(t_k)$.

In contrast, traditional ATG approaches usually need to compare each pair of data instances when determining the cutting levels, and so their computational complexities are quadratic with respect to the size of D. This conclusion is supported by our experimental results in Section 6.4.

6.4 Experiments

To evaluate our DIVA-based ATG algorithm, we compare it with two well-known taxonomy construction approaches that utilize pure binary agglomeration/division: (1) HAC-based approach [25] and (2) bisecting k-means Partitioning approach [138] (here we replace k-means by k-medoids as the micro-clustering algorithm because our experiments were conducted on relational datasets instead of numeric vectors) that is performed iteratively until every leaf node contains only one data instance. Then the derived binary-split dendrogram is merged by using the taxonomy generation algorithm proposed in [22]. For the sake of simplicity, in this chapter we use the words "Global-Cut", "Local-Cut", "HAC" and "BKM" to represent the above four approaches respectively. For the BKM approach, the number of iterations within k-medoids is set to 10 when any taxonomic nodes are split and the parameter k is fixed to 2 since the approach uses binary partitioning. For our DIVA approach, the number of ROs r is set to 3 and the variance v is 0.5, which are the same as those in Chapter 4. Such parameter settings guarantee all the leaf nodes in the derived taxonomy are homogeneous enough while the total number of nodes is far less than the size of the dataset. Both BKM and DIVA

approaches are repeated 5 times with different random seeds and the final results are the average value of their respective experimental results obtained in different times.

As outlined at the beginning of this chapter, the optimized taxonomy should maximize the intra-node-uniformity and the sibling-nodes-distinctiveness. We propose to use the following distance-based criterion to evaluate the uniformity (purity) of all nodes in the derived taxonomy. For node t_k, the intra-node distance is:

$$Dis_{Intra}(t_k) = \frac{1}{|\{x_i^{(t_k)}\}|^2} \sum_{i,j} fd_{\text{obj}}(x_i^{(t_k)}, x_j^{(t_k)}) \tag{6.9}$$

Similarly, the inter-node distance of node t_k is:

$$Dis_{Inter}(t_k) = \frac{1}{|\{t_{k'}\}|} \sum_{t_{k'}} \left[\frac{1}{|\{x_i^{(t_k)}\}| \times |\{x_j^{(t_{k'})}\}|} \sum_i \sum_j fd_{\text{obj}}(x_i^{(t_k)}, x_j^{(t_{k'})}) \right] \tag{6.10}$$

where $\{t_{k'}\}$ are t_k's sibling nodes. Generally speaking, lower intra-node distance value means taxonomical nodes are more homogeneous and higher inter-node distance means they are more distinctive to each other.

Another important factor is the size of the taxonomy, denoted as $|T|$. The derived taxonomies should not be too large or too small, but it is very difficult to appoint the accurate value for that purpose. As proposed in [25], we user the square root of $|T|$ as the expected number of the leaf nodes. Then the following criterion considers all the above factors and thus evaluate the quality of the whole taxonomy:

$$QualityDis(T) = \frac{\sum_{t_k} Dis_{Inter}(t_k)}{\sum_{t_k} Dis_{Intra}(t_k)} \times \left(1 - \frac{\left| \, |T_L| - \sqrt{|T|} \, \right|}{|T|} \right) \tag{6.11}$$

where $|T_L|$ is the actual number of leaf nodes in T. According to the definition of $IntraDis(t_k)$ and $InterDis(t_k)$, we prefer the derived taxonomy with higher value of $QualityDis(T)$.

The experiments were conducted on two relational datasets: (1) the Synthetic

Amazon dataset introduced in Section 4.5.1 and (2) the real movie dataset in Section 4.5.3.

6.4.1 Synthetic Amazon Dataset

Since the class label of each data instance in this dataset is already known, we can also use the entropy-based criteria to evaluate the quality of the derived taxonomy. For node t_k, its intra-node entropy is:

$$E_{intra}(t_k) = -\sum_h P_{h,k} \log_2 P_{h,k} \tag{6.12}$$

where $P_{h,k}$ is the proportion of data instances of class h in node t_k. Similarly the sibling-nodes distinctiveness (named as the inter-node entropy) is evaluated by the Kullback-Leibler Divergence [35]:

$$E_{inter}(t_k) = \sum_h P_{h,k} \log_2 \frac{P_{h,k}}{Q_{h,k}} \tag{6.13}$$

where $Q_{h,k}$ is the default proportion of data instances of class h, which is estimated based on the data distribution of t_k's parent node. Then $E_{inter}(t_k)$ measures the distinctiveness between the current node t_k to its sibling nodes. Generally speaking, lower intra-node entropy value means taxonomic nodes are more homogeneous and higher inter-node entropy value means they are more distinctive to each other. We have:

$$QualityEntropy(T) = \frac{\sum_{t_k} E_{inter}(t_k)}{\sum_{t_k} E_{intra}(t_k)} \times \left(1 - \frac{\left|\, |T_L| - \sqrt{|T|} \,\right|}{|T|}\right) \tag{6.14}$$

The meanings of $|T_L|$ and $|T|$ have been explained above.

Table 6.1 presents our experimental results. The first line records the time spent by all the ATG approaches we examined. The HAC approach requires to determine the medoid for each cluster by comparing all the pairs of data instances in that cluster, and

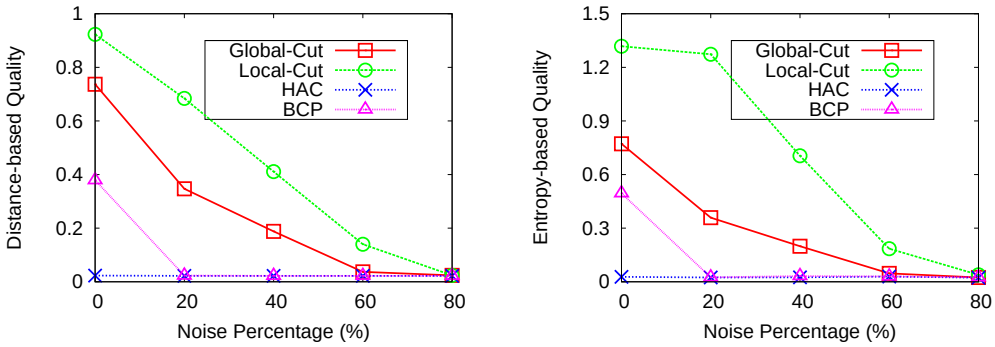
Table 6.1: Experimental results using the Synthetic Amazon dataset

	HAC	BKM	Global-Cut	Local-Cut
Time Spent (sec)	5227.4	7369.9	2127.9	2095.4
Size of Taxonomy	3963	2421	1187	441
Number of Leaves	1998	1291	594	268
Intra-node Distance	0.814	0.813	0.806	0.800
Inter-node Distance	0.818	0.818	0.819	0.821
Quality (based on Distance)	0.023	0.379	0.737	0.912
Intra-node Entropy	0.048	0.493	0.182	0.382
Inter-node Entropy	0.057	0.647	0.193	0.569
Quality (based on Entropy)	0.027	0.495	0.772	1.326

so the algorithm has quadratic computational complexity. The BKM spends even more time because it launches the k-Medoids algorithm to split every leaf node with more than one data instance. In contrast, the DIVA approach is very efficient due to the utilization of ROs to represent the clusters, which guarantees the assignment of non-RO data to be done in linear time and the operation of optimizing the taxonomic structure is linear as well.

The second and the third lines in Table 6.1 show that the size of the taxonomies generated by DIVA are substantially smaller than that of the other two approaches. In Global-Cut the taxonomy contains approximately 1200 taxonomic nodes, among which are 600 leaf nodes. In Local-Cut the taxonomy contains less than 500 taxonomic nodes and 250 leaves. These numbers are far less than those derived by HAC or BKM, which means the taxonomy generated by DIVA can better satisfy Properties *Compactness* and *Reach Time* proposed at the beginning of this chapter.

The rest of Table 6.1 are the evaluation results using the distance-based and the entropy-based criteria. We can see the taxonomy derived by HAC is far worse than the other three approaches. Our Global-Cut approach achieved 94.5% and 56.0% improvements upon BKM when evaluated by the distance-based and the entropy-based criteria respectively. Furthermore, Local-Cut is substantially better than Global-Cut with respect to the size as well as the quality of the derived taxonomy, which proves our assumption that the local optimization strategy is more effective than the global



(a) Distance-based Quality (b) Entropy-based Quality

Figure 6.2: Synthetic Amazon Dataset - Evaluate Taxonomy w.r.t noise

Table 6.2: Evaluation results using the movie dataset

	HAC	BKM	Global-Cut	Local-Cut
Size of Taxonomy	3411	4509	1184	271
Number of Leaves	2496	2983	593	223
Intra-node Distance	0.770	0.703	0.647	0.648
Inter-node Distance	0.699	0.709	0.649	0.662
Quality (based on Distance)	0.468	0.421	0.897	0.991

optimization.

Finally, Figure 6.2 illustrates the robustness of all approaches under different noise ratios of browsing actions, ranging from 0% to 80%. In general, the quality of the taxonomies derived by all approaches are reduced as the noise ratio increases. HAC performs the worst in all cases. Local-Cut performed the most stable and generated taxonomies of the highest quality under different noise ratios.

6.4.2 Real Dataset

All the taxonomy construction approaches were also evaluated using the real movie dataset. Similar to the experimental settings in Section 4.5, we use the top-5000 popular movies for generating the taxonomies. The other experimental parameters were kept the same as in the previous section.

Table 6.2 lists the experimental results of all approaches. Again the taxonomies

built by Global-Cut and Local-Cut are more compact than that of the other two approaches. Moreover, the taxonomies derived by the Local-Cut strategy is far better than the HAC- and BKM-derived taxonomies with the improvements of 111.7% and 135.4% respectively using the criterion of distance-based quality.

Figure 6.3 shows parts of the derived taxonomy. The movies corresponding to the leaf nodes in the figure are listed in the Table 6.3. We can see the derived taxonomy organizes these movies in a reasonable order: Node 1177 mainly contains the horror/thriller and crime movies; Node 1170 contains the movies of science fiction; Node 1105 is for action movies while its siblings nodes are for comedies and dramas.

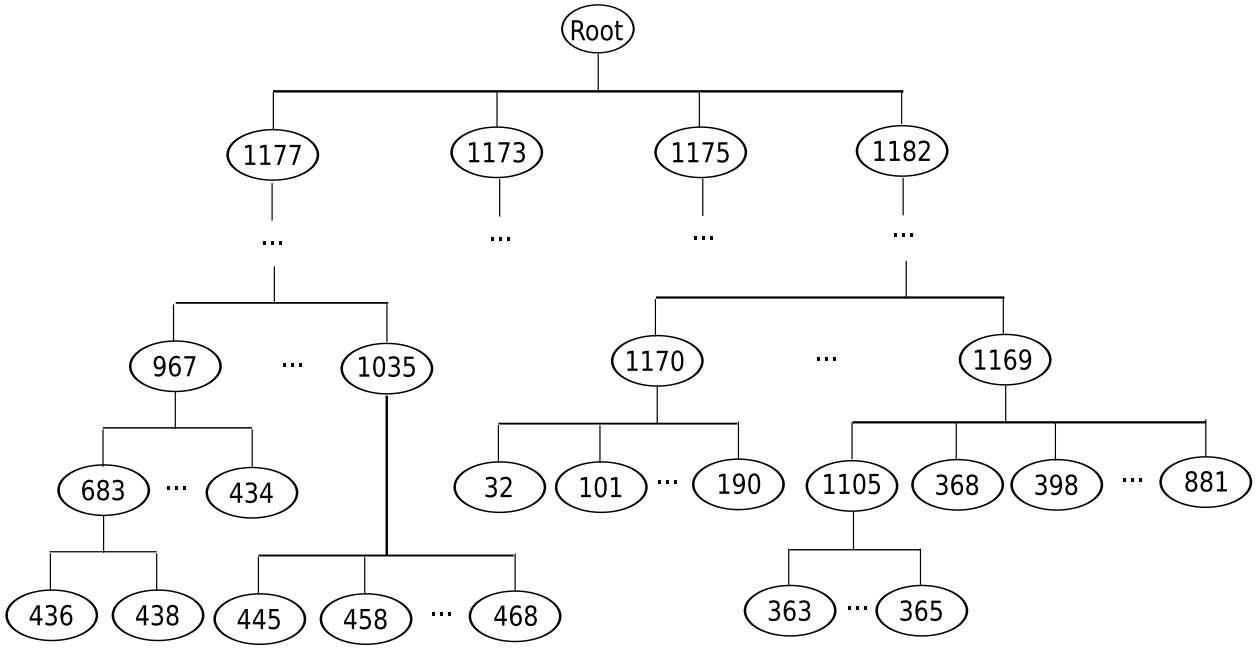


Figure 6.3: Part of the Labeled Movie Taxonomy

Table 6.3: Taxonomy Nodes in Figure 6.3 with Corresponding Movies

Id	Title	Id	Title
32	Doctor Who	101	Final Fantasy
			Star Trek
			Star Trek Voyager
			Space: 1999
			War Of The Worlds, The
190	Star Wars	363	Xena Warrior Princess
	Fifth Element, The		Right Stuff, The
	Solaris		Days Of Thunder
	Mission To Mars		Ghost In The Shell
	Men In Black		Top Gun
365	Terminator, The	368	American Pie
	Eraser		Road Trip
	True Lies		South Park
	Rambo - First Blood		Born Romantic
	Mission Impossible		Woman On Top
398	Bridget Jones's Diary	881	Notting Hill
	Dude, Where's My Car?		Puppetry Of The Penis
	But I'm A Cheerleader		Pretty Woman
	As Good As It Gets		Four Weddings And A Funeral
	Me, Myself And Irene		Shakespeare In Love
434	Kalifornia	436	Silence Of The Lambs
	Bound		Hannibal
	Rosemary's Baby		Hitchcock Collection
	Apocalypse Now		Way Of The Gun, The
	Devil's Advocate		Hole, The
438	Fifteen Minutes	445	Blow
	Leon		8MM

Continued on next page

Table 6.3 – continued from previous page

Id	Title	Id	Title
	Seven		Crash
	Gangster No. 1		Casino
	Face/Off		U-Turn
458	Stanley Kubrick	468	Godfather Trilogy
	Twin Peaks: Fire Walk With Me		Traffic
	Taxi Driver		Clockwork Orange
	Blood Of Dragon Peril		Requiem For A Dream
	Way Of The Dragon, The		True Crime

To better distinguish the content of these taxonomic nodes, we need to assign a label for each node. Intuitively, at the top few levels the movie attribute "Genre" can well distinguish a taxonomy node from its siblings. However, when navigating deeper levels in the taxonomy, the movie genres of sibling nodes become more and more homogeneous, and so other attributes such as the keywords in the title or the year of movie released or the name of actors will be used as the labels. In the next chapter we will explain our strategy of labeling the taxonomic nodes.

6.5 Summary

In this chapter we propose our automatic taxonomy generation approaches based on the relational clustering framework DIVA. Compared with the flat cluster result or binary-split dendrogram of cluster hierarchy, taxonomies are more suitable for people and computers to navigate and exploit the dataset because data instances are summarized by taxonomic nodes at different levels, reflecting the underlying data distribution of different homogeneity. Here we consider the task of constructing the taxonomic structure as the search problem in a certain hypothesis space, and develop two strategies to find the optimal

taxonomy with high efficiency. In the next chapter we will study another important problem, i.e. how to assign labels for taxonomic nodes so as to best distinguish their different content.

Chapter 7

Labeling the Automatically Generated Taxonomic Nodes

In the previous chapter we demonstrated our approach of optimizing the binary-split dendrogram to generate a well-structured taxonomy based on the Representative Objects and the DIVA clustering framework. A non-trivial issue that has not been tackled is how to label the taxonomic nodes in order to satisfy the Property *Node Label Predictiveness* discussed at the beginning of Section 5.1. More specifically, we expect the assigned labels to best summarize the content of each node and reflect their pairwise distinctiveness.

Actually this issue is rather subjective in nature. There is no widely accepted criterion to evaluate the goodness of the assigned labels. Current research mainly focuses on labeling taxonomies built for a large collection of textual documents. In such cases we can utilize the techniques of Natural Language Processing, Information Retrieval or Computational Linguistics to pre-process the text and extract keywords or concepts. Then based on these keywords/concepts the node labels are determined according to the frequency of keywords or the correlation between concepts [108][25][26].

In this chapter, we try to answer the following question: *Given a taxonomy that has been generated automatically from relational datasets using some unsupervised learning techniques, how can the taxonomic nodes be labeled appropriately so as to*

summarize the content of each node as well as reflect the distinctiveness between sibling nodes?

To explain the situation more clearly, we still use the movie dataset as an example. Given the taxonomy of movies that has been built using the algorithm introduced in Chapter 6, labels of the taxonomic nodes should now be determined to help users navigate the taxonomy. For different taxonomic nodes, the movie attributes are usually of different importance in discriminating the corresponding movie sets: When the nodes mainly consists of action movies, users often care about the leading *Actors* (e.g. Arnold Schwarzenegger, Sylvester Stallone, Jackie Chan); for taxonomic nodes of ethical films, the *Director* (e.g. Ingmar Bergman, Federico Fellini, Krzysztof Kieslowski) may better summarize the movies allocated to these nodes; while the attribute *YearOfRelease* and keywords appearing in the *Title* are two good discriminant indicators for the taxonomic nodes consisting of documentaries. The values of these attributes or related concepts constitute the labels of the corresponding taxonomic nodes.

To the best of our knowledge, the problem of automating the label assignment within the taxonomy built from relational datasets has not been investigated until now. Hence, we developed an approach to selecting the labels of taxonomic nodes by quantitatively evaluating the homogeneity of each node and the heterogeneity of its sibling nodes. Furthermore, we proved that our approach is mathematically equivalent to the Decision Tree Induction algorithm in case that the classes of data instances are already known.

7.1 Labeling Strategies

The aim of labeling taxonomic nodes is to choose some predictive labels that best summarize the content of each node as well as highlight its distinctiveness from the siblings. We first consider a simplified case in which the taxonomy is composed of one parent-node and a set of sub-nodes: Given a set of relational data instances $D = \{x_i\}$ $(1 \leq i \leq N)$ organized by a two-layered taxonomy T, as shown in Figure 7.1. t_p

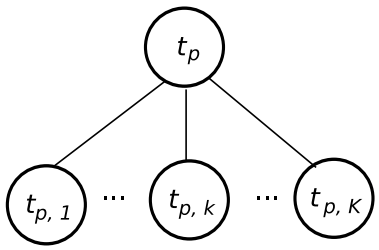


Figure 7.1: Example Taxonomy Sub-Tree

is the non-leaf node in T with K child nodes $\{t_{p,k}\}$ $(1 \leq k \leq K)$, we try to determine a label for each child node $t_{p,k}$ based on the attributes of all relational data contained in $t_{p,k}$ and its sibling nodes. If taxonomy T has multiple layers, we can divide T into a set of two-layered sub-taxonomies and then launch the algorithm for each sub-taxonomy independently.

We proposed a novel labeling strategy for solving the above problem. The key idea is to exploit the divergence of the probability distributions of data instances in different taxonomic nodes. Our algorithm is based on the following assumption: If a child node is obtained by randomly picking out data instances from the parent node, the probability distribution of the child node should be the same as that of the parent node; otherwise, the divergence between these two probability distributions emphasizes the difference between the parent and the child nodes. As being used in many research fields, the Kullback-Leibler Divergence [82] evaluates the divergence of one probability distribution from another. Hence, in this section we will investigate the applicability of KL Divergence within a taxonomy built for relational datasets. The bias of KL Divergence is then analyzed, leading to the development of a new synthesized criterion. Finally we develop two strategies to determine the node labels using KL Divergence and provide experimental results to show the effectiveness of our algorithm.

Before considering the taxonomy built for relational datasets, we first discuss the propositional case where all the data are only defined by a set of attributes $\{A_r\}$. According to our assumption, one or several attributes that best distinguish the data instances contained in node $t_{p,k}$ from those data contained in $t_{p,k}$'s sibling nodes are

chosen as the label of node $t_{p,k}$. Given an attribute A_r, assuming $p_{k,r}$ and q_r are the probability distributions defined over the domain of A_r for instances belonging to nodes $t_{p,k}$ and t_p respectively, the KL-divergence of $p_{k,r}$ from q_r is defined as:

$$KL_r(p_{k,r}, q_r) = \sum_{v \in \mathrm{Dom}(A_r)} p_{k,r}(v) \log \frac{p_{k,r}(v)}{q_r(v)} \tag{7.1}$$

where $\mathrm{Dom}(A_r)$ is the domain of A_r. Because all instances are assigned to leaf nodes, the distribution q_r is computed as the weighted sum of its child-node distributions:

$$q_r = \sum_{k=1}^{K} \frac{N_k}{N} p_{k,r} \tag{7.2}$$

where N_k is the number of data instances contained in child-node $t_{p,k}$ and $N = \sum N_k$. Because $KL_r(p_{k,r}, q_r)$ measures how important the attribute A_r is in distinguishing the members of node $t_{p,k}$ from the members of its sibling nodes, the attribute that has the maximum value of KL-divergence provides the basis for naming the node $t_{p,k}$.

The KL-divergence measure is generally unbounded. From Equation 7.1, we can estimate its upper bound in the context of the taxonomy labeling task, which is $\log \frac{N}{N_k}$. It is interesting that this upper bound is independent of the attribute domain from which the KL-divergence is computed. More details about the proof are provided in Appendix 7.4.1.

Now we consider the relational case where data instances are defined by both attributes and other related instances. The above procedure can be extended to calculate the KL-divergences for relationships as well as attributes. For example, each movie instance is related to a set of actors via the relationship *ActedBy* (through the table *Movie-Actor*). Given a set of movie instances contained in $t_{p,k}$, the size of the related actor instances is $N_{k.actor}$, which is usually greater than N_k. Then the range of KL-divergence for relationship *ActedBy* will be $\left[0, \log \frac{N.actor}{N_k.actor}\right]$ [1]. Similarly for re-

[1]The deduction procedure is very similar to that in Appendix 7.4.1, so we ignore it here for brevity.

lationship *DirectedBy*, the corresponding value range is $\left[0, \log \frac{N.director}{N_k.director}\right]$. Since the KL-divergences for different relationships have different ranges, they should be normalized to the interval $[0, 1]$ before being compared.

When being used as the criterion of determining node labels, the KL-divergence will be biased towards the preference of attributes with more unique values over those with fewer values. This phenomenon is similar to the bias in the Information Gain used in the procedure of Decision Tree induction [103]. To address this problem, we propose the KL-Ratio (KLR) measure as:

$$KLR_r(p_{k,r}, q_r) = \frac{KL_r(p_{k,r}, q_r)}{E_r(t_p)} \tag{7.3}$$

where $E_r(t_p) = -\sum_j \left(\frac{n_j}{N} \log \frac{n_j}{N}\right)$ is the entropy of the attribute A_r within the parent node t_p. Experimental results presented in Section 7.2 show that the normalization and the entropy-based adjustment together can effectively reduce the bias of the original KL-divergence measure.

In [83] the authors suggested that all the sibling nodes should use the same attribute to construct their labels, solely differing in the attribute values. To do this, we utilize the KLR criterion (Equation 7.3) to select the attribute A_r^* that has the maximum weighted sum of KLR values across all child-nodes:

$$A_r^* = \arg\max_{A_r} \sum_k \frac{N_k}{N} KLR_r(p_{k,r}, q_r) \tag{7.4}$$

It is interesting that this strategy is mathematically equivalent to the use of Gain Ratio as the criterion to determine the split attribute in Decision Tree Induction. The detailed proof is provided in Appendix 7.4.2. However, we must point out the fundamental difference between Decision Tree Induction and our ATG-based approach: the former belongs to supervised learning, i.e. the class information for all training data are already known before the learning procedure; in contrast, our approach is based on the ATG algorithms, which is in essential unsupervised and hence has no prior knowledge about

the class information. In summary, Decision Tree Induction and our ATG-based labeling aim at solving similar problems under different motivations and prerequisites.

After the most discriminative attribute A_r^* is identified by Equation 7.4, the node label is composed of the attribute name as well as the attribute values owned by the data instances in this node. Sometimes data instances in the node might contain too many different values for the attribute A_r^* (in our movie example, when the attribute "Actor" is considered as the node label, the data instances in a node may refer to tens or even hundreds of different actors). From the practical viewpoint, enumerating all possibles attributes in a node label is not a good idea, so we only use the top-10 most frequent attribute values to construct the node label.

7.2 Experimental Results

Some experiments were conducted to evaluate our algorithm presented above. We first evaluate the bias within the original KL-divergence measure and show to what extent KLR can reduce such bias. Then two strategies for labeling nodes are compared through the simulation of a user locating a given set of movies in the taxonomy. Here the real movie dataset is again used in our experiments, among which we selected 10,000 most popular movies as the dataset.

7.2.1 Bias within Kullback-Liebler Divergence

To determine whether the use of the original KL-divergence measure for selecting the most informative attribute is biased or not, we conducted the experiments upon the movie dataset as an example: three sets of 100 movies were randomly chosen to form sibling nodes t_{s1}, t_{s2} and t_{s3}, which shared the common parent node t_s. For each sub-node composed of sampled movies, we calculate the KL-divergences for all the attributes. This experiment was repeated 50 times with different random seeds. It is worth noting that the conclusions drawn in this section can be generalized to any other datasets when the number of unique attribute values are substantially different.

Table 7.1: Bias of Different Attributes

Attribute	Number of Unique Values	KL-divergence	Normalized KL-divergence	Entropy	KLR
Title	433.320±15.336	1.118 ± 0.084	0.704 ± 0.028	8.424 ± 0.048	0.084 ± 0.003
Year	16.000± 0.941	0.081 ± 0.026	0.051 ± 0.016	3.210 ± 0.065	0.016 ± 0.005
Certificate	9.500± 0.642	0.045 ± 0.020	0.028 ± 0.012	2.597 ± 0.081	0.011 ± 0.005
Genre	31.780± 2.588	0.170 ± 0.033	0.107 ± 0.021	3.883 ± 0.098	0.028 ± 0.005
Director	223.280±16.396	1.026 ± 0.068	0.647 ± 0.035	6.378 ± 0.279	0.101 ± 0.003
Actor	1070.020±67.692	1.323 ± 0.100	0.832 ± 0.021	9.731 ± 0.122	0.086 ± 0.002

Table 7.1 shows the average number of unique values for each attribute contained in the parent node t_s as well as the mean and the standard deviation of using different labeling criteria with respect to each attribute. The movie titles were processed to extract meaningful nouns, verb and adjectives using WordNet (`http://wordnet.princeton.edu/`). In Tables 7.1, the original KL-divergence values for attributes Title, Director and Actor are greater than 1 while the others are far less than 1, which proves the necessity for normalizing the KL-Divergence as suggested in Section 7.1. Furthermore, the number of unique values for different attributes vary greatly and the original KL-divergence is proportional to this number. The entropy, which acts as the penalty factor in the KLR criterion, is also impacted by that because the attributes with more unique values also have higher entropy. As can be seen from the last column of Table 7.1, KLR that synthesizes the KL-Divergence and the entropy can effectively reduce the bias.

7.2.2 Empirical Study

Figure 7.2 shows part of the labelled movie taxonomy using the strategy explained in Section 7.1. All the sibling nodes in the figure have the same decision attribute with different values as their labels. It is worth noting that, depending on the utilized techniques of taxonomy generation, some data objects belonging to two different taxonomic nodes might share the same attribute values, e.g. both Node 368 and Node 398 in Figure 7.2 use the genre value *Comedy* in their labels. To handle such overlapped node

labels, we add the second most discriminative attribute (identified by the Equation 7.4) into the node labels to better distinguish the node content. In Figure 7.2, the attribute "Title" is appended together with the first discriminative attribute "Genre" to construct the labels for Nodes 368, 398 and 881.

To quantitatively evaluate the goodness of these labels, we developed a program that simulates the procedure of a user navigating the taxonomy to find the movies satisfying his interests. For some randomly given movies, a robot will navigate the taxonomy in a top-down fashion to identify the correct leaf nodes that contain the target movies. When examining a non-leaf node, the robot will use the node label to determine which sub-node should be explored in the next iteration. If the target movie matches the labels of more than one sub-node, all the matched sub-nodes will be explored in a best-match-first order.

Our node labelling approaches are compared with the following methods of labeling the taxonomic nodes: 1) *RandomAttr*: We randomly select an attribute as the label for each taxonomic node; 2) *Fixed-Concept*: A unique attribute is used to construct the labels for all the nodes in the taxonomy. For example, "Fixed-Genre" means the movie genres are used as the node labels over the taxonomy. All the attributes were individually evaluated in our experiments.

As we explained previously, not all attribute values are enumerated in the node labels for the sake of practical brevity. Therefore, some data instances are not covered by the node label when their values of the most discriminative attribute are not included in the label. We use the criteria *success ratio* and *path length* to measure the search performance: the former is evaluated by the proportion of the movies (in the given set) that have been identified successfully in the taxonomy; the latter is the average number of nodes that the robot have to examine before reaching the correct leaf node. In our experiment, 100 randomly selected movies were used in each run and the experiment was repeated 10 times.

Table 7.2 reports our experimental results. We can see here the attributes Title,

Table 7.2: Results of Simulation Experiment

	Success Ratio	Path Length
RandomAttr	19.0 $\pm$ 4.2	44.2 $\pm$ 5.3
Fixed-Title	31.0 $\pm$ 3.6	59.2 $\pm$ 10.0
Fixed-Year	100.0 $\pm$ 0.0	251.7 $\pm$ 14.3
Fixed-Genre	93.2 $\pm$ 2.7	163.4 $\pm$ 13.9
Fixed-Director	10.0 $\pm$ 3.1	12.1 $\pm$ 2.5
Fixed-Actor	7.7 $\pm$ 2.3	15.7 $\pm$ 1.6
KLR-Based	88.8 $\pm$ 2.9	64.6 $\pm$ 3.2

Director and Actor are so specific that it hardly succeeded in finding the expected movies in the taxonomy if one of these attributes is exclusively used as the node labels. In the other extreme the attributes Year and Genre are too general: the robot has to explore nearly the whole taxonomic structure before reaching the target node, considering that the total number of nodes in the taxonomy is 271 (Table 6.2). Instead, our KLR-based approach achieves a good balance between them: the robot can identify most of the target movies (high success ratio) with low exploration cost (short exploration path). Such advantage will be very useful when the derived taxonomy is used for the recommender systems in the next part.

7.3 Summary

As the supplement of Chapter 6, we investigate another important problem in automatic taxonomy generation: how to assign meaningful labels for taxonomic nodes in order to distinguish the content of different nodes. Here we tackle the problem creatively by analyzing the probability distribution of data attributes in different taxonomic nodes and then developing a new synthesized criterion based on the Kullback-Liebler Divergence to identify the most informative attribute. In addition, we conduct empirical studies as well as simulation experiments to evaluate our taxonomy labeling approaches.

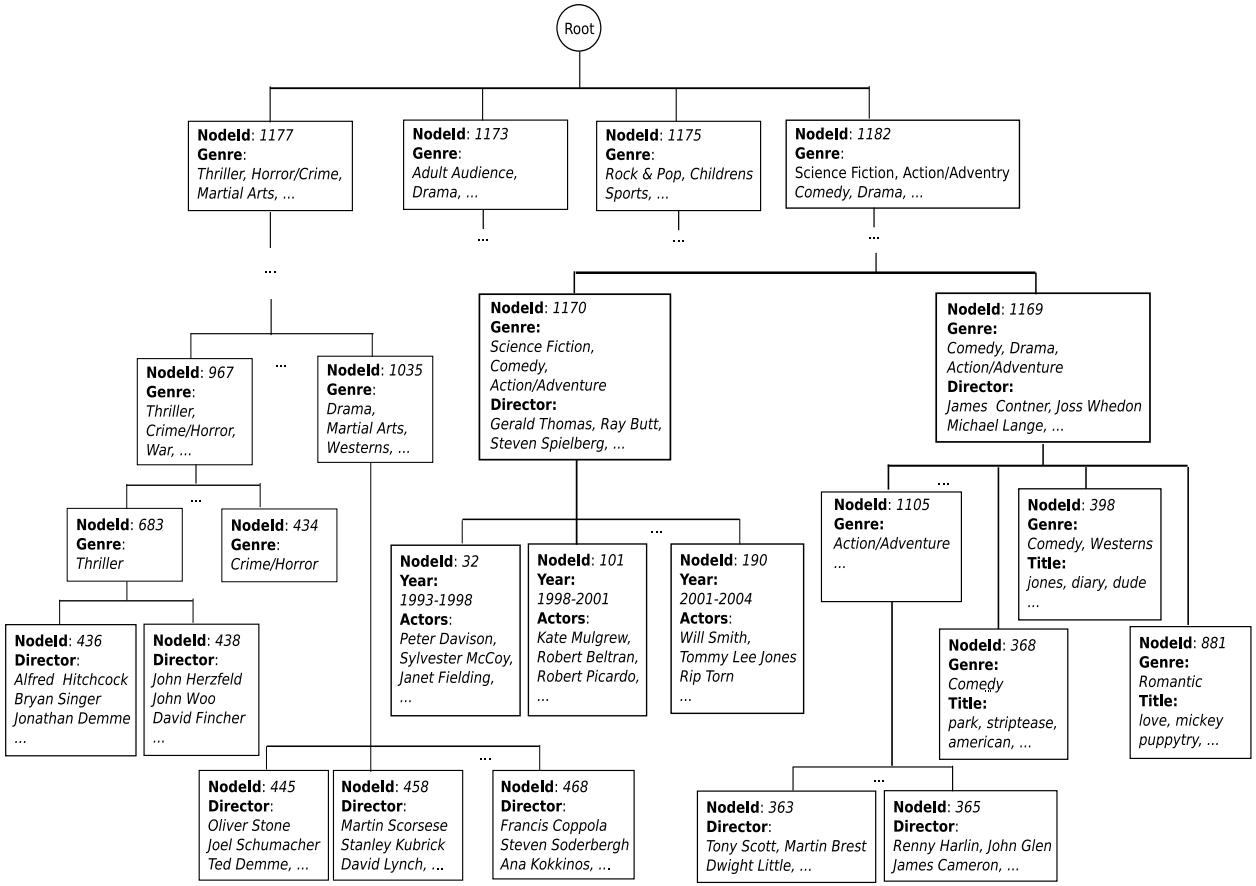


Figure 7.2: Labeling Taxonomy for Movie Dataset

7.4 Appendix

7.4.1 Proof of the KL-divergence Bounds

Proof. Given the symbols N, N_k, A_r, $p_{k,r}$ and q_r defined as in Section 7.1, it is easy to see that the KL-divergence for attribute A_r will be maximized when the distributions $p_{k,r}$ and $q_{\backslash k,r}$ have non-zero probabilities for disjoint subsets of values in A_r, where $q_{\backslash k,r} = \sum_{l \neq k} \frac{N_l}{N - N_k} p_{lr}$. The upper bound is then:

$$\begin{aligned}
\max KL_r(p_{k,r}, q_r) &= \sum_{v_j \in \mathrm{Dom}(A_r)} p_{k,r}(v_j) \log \frac{p_{k,r}(v_j)}{q_r(v_j)} \\
&= \sum_j \left(\frac{n_j}{N_k} \log \frac{\frac{n_j}{N_k}}{\frac{n_j}{N}} \right) \\
&= \sum_j \left(\frac{n_j}{N_k} \log \frac{N}{N_k} \right) \\
&= \log \frac{N}{N_k}
\end{aligned}$$

where $\frac{n_j}{N_k} \left(\sum_j \frac{n_j}{N_k} = 1 \right)$ is the frequency that the j-th value of A_r occurs in the objects contained in child-node t_{sk}. Therefore, the range of KL-divergence for node t_{sk} is $\left[0, \log \frac{N}{N_k}\right]$, which is not impacted by the pre-specified attribute A_r. □

7.4.2 Proof of the Equivalence between Information Gain and KL-based Strategy

Proof. The Information Gain used in the Decision Tree Induction is defined as:

$$
\begin{aligned}
& InfoGain_r(t_s) \\
&= E(t_s) - \sum_k \left(\frac{N_k}{N} \cdot E(t_{sk}) \right) \\
&= -\sum_j \left(\frac{n_j}{N} \log \frac{n_j}{N} \right) + \sum_k \left[\frac{N_k}{N} \cdot \sum_j \left(\frac{n_{kj}}{N_k} \log \frac{n_{kj}}{N_k} \right) \right] \\
&= -\sum_j \sum_k \left(\frac{n_{kj}}{N} \log \frac{n_j}{N} \right) + \sum_k \sum_j \left(\frac{n_{kj}}{N} \log \frac{n_{kj}}{N_k} \right) \\
&= \sum_k \sum_j \left(-\frac{n_{kj}}{N} \log \frac{n_j}{N} + \frac{n_{kj}}{N} \log \frac{n_{kj}}{N_k} \right) \\
&= \sum_k \sum_j \left(\frac{n_{kj}}{N} \log \frac{\frac{n_{kj}}{N_k}}{\frac{n_j}{N}} \right) \\
&= \sum_k \sum_j \left(\frac{N_k}{N} \cdot p_{k,r(v_j)} \log \frac{p_{k,r}(v_j)}{q_r(v_j)} \right) \\
&= \sum_k \left(\frac{N_k}{N} \cdot KL_r(p_{k,r}, q_r) \right)
\end{aligned}
$$

The Gain Ratio is defined as the ratio of Information Gain and the entropy of parent node with respect to the attribute A_r [103], which gives:

$$
\begin{aligned}
GainRatio_r(t_s) &= \frac{InfoGain_r(t_s)}{E_r(t_s)} \\
&= \sum_k \left(\frac{N_k}{N} \cdot \frac{KL_r(p_{k,r}, q_r)}{E_r(t_s)} \right) \\
&= \sum_k \frac{N_k}{N} KLR_r(p_{k,r}, q_r)
\end{aligned}
$$

This completes the proof. □

Part III

Recommender Systems

Today's information technologies allow us to produce, store and transfer a vast amount of data, but unfortunately in many scenarios it is not easy for us to find useful information from these data. For example, online retailers such as Amazon or Staples provide thousands of commodities and various choices for the same type of commodities, regarding different brands, qualities or packages. A customer has to spend substantial time and effort on finding the best one to satisfy his requirements. On the web the number of published documents has exceeded a trillion and is growing by several billions per day [2]. Even the most powerful search engines today cannot index all the web documents. The incapability of people to digest such huge volumes of data, generally referred to as the problem of *information overload*, becomes more and more critical in many applications.

Although lots of data mining and knowledge discovery algorithms have been developed, they are not always effective in practice: the standard search engines usually return a ranked list with thousands of web documents as the response of one query, but most users only have patience to read the top-20 results and give up the others even if they contain relevant information. When the search results are not satisfactory, users have to manually change the keywords in the query or adjust the search strategies, which means they need to master advanced searching skills or domain-specific knowledge. On the other hand, some algorithms generate sophisticated data models that are not intuitive for non-professional users.

Recommender systems are proposed specifically to tackle the problem of information overload. They utilize information filtering techniques to find new items that best match the interests of different users [1]. Traditionally, all the items and users are represented as numeric vectors, e.g. the keyword vectors (extracted from the content description by the techniques of text mining or information retrieval) in content-based filtering or the user rating vectors (explicitly or implicitly collected by the techniques of Web Usage Mining) in user-based collaborative filtering. Based on such vector models,

[2] `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`

the similarity between items or users, which is the key to discover the neighborhood of items and thus generate the recommendations, can easily be calculated using standard metrics, such as cosine similarity or Pearson's correlation coefficient. A bundle of candidate items that best match the active user's interest, i.e. most similar to the items he likes and dissimilar to those he dislikes, are returned as the recommendations.

Recent research focuses on exploiting the domain knowledge about items to improve the recommendation quality and address the problem of data sparseness [30] [102] [4]. However, dealing with semantic relational knowledge will greatly increase the complexity of similarity calculation and hence impair the scalability of recommender systems. Various clustering techniques have been applied as an offline pre-processing step to address the difficulty. The primary motivation behind these approaches is to group items or users and then employ traditional collaborative filtering algorithms upon the group profiles [105] or independently within each group rather than all items/users [140]. Nevertheless, these approaches are less helpful when the items are defined by sophisticated domain knowledge. For example, when investigating our movie datasets used in Chapter 2, calculating the pairwise similarities within a small set of movies is still time-consuming.

One promising research direction is to construct some compact data model for preserving the item similarities. Recommender systems can then retrieve these values directly instead of computing them online when generating recommendations. Precomputing pairwise item similarities requires a space complexity of $O(n^2)$. Improving on the memory requirement is hence a useful practical goal to make recommender systems scale to the real-world applications. In this part we propose an efficient approach based on the ATG technique to incorporate relational domain knowledge into recommender systems so as to achieve high system scalability and predication accuracy. By analyzing domain knowledge about items in the offline phase, a hierarchical data model is synthesized to simulate the pair-wise item similarities. The derived data model is then used by online recommender systems to facilitate the similarity calculation and at the

same time keep their recommendation quality comparable to the memory-based systems that exploit the domain knowledge in real-time.

Another approach is to group the candidate users based on their rated items, which can then be used to reduce the number of users that have to be examined when formulating neighbours for the active user. This is akin to indexing users in lazy learning approaches such as using kd-trees or other case indexing approaches of case-based reasoning.

This part is organized as follows: In Chapter 8 recommendation algorithms are reviewed in details. After that, we focus on exploiting the automatically generated taxonomy to improve the scalability of recommender systems in Chapter 9. Comprehensive experiments were conducted to show the effectiveness of our approaches.

Chapter 8

Review of Recommender Systems

Web personalization aims to provide users with what they want or need without requiring them to ask for it explicitly [109]. Such user-adaptive systems will automatically tailor the delivered content based on the past/present behaviors of the current user and inferences from other like-minded users [159]. As an implementation of web personalization, recommender systems utilize information filtering techniques to select items (for example movies, music or books) that best match the interests of different users [1]. In this chapter we will review the fundamental principles and some important issues in recommender systems.

8.1 Fundamentals of Recommender Systems

Based on the information employed for generating recommendations, current recommender systems can be classified into two main categories, namely *content-based* and *collaborative* filtering [1] [3]:

Content-based filtering systems [10] [116] describe the items in the form of free text on a fixed set of attribute-value pairs and make the assumption that user preferences are captured by the description of items they liked in the past. Therefore, these systems usually represent each item by a set of textual or numeric attributes extracted from the

item's content description. The profile of the current user, being composed of the above attributes, summarizes all the items he has viewed. Items not previously viewed by the user are then compared with this profile to see whether they fit the user's preference so that the top-k most similar items are recommended to the user. One common problem in these systems is that the recommended items are too similar to those the user has been visited, which we call "overspecialization".

Collaborative filtering systems [125] [114] [131] do not deal with the content of items. They assume users can be divided into different groups based on their interests and the current user will prefer those items that are also preferred by other users in the same group. According to this idea, the user profiles, composed of the user ratings upon items they have viewed, are used to identify the neighbourhood of the current user, which are other users with the same taste. Such collaborative-based systems can effectively avoid the overspecialization problem faced by content-based approaches and have more chance of recommend unexpected items, which can greatly increase user satisfaction. However, these collaborative systems need to obtain a considerable amount of user ratings before producing useful recommendations. Recently, item-based collaborative filtering techniques [130] [32] were proposed to improve the scalability of recommender systems, which first analyzes relationships between items and identifies k similar items rated by different users and then performs the similarity computation as traditional collaborative filtering but in the data space of items to accelerate the online recommendation procedure.

Generally speaking, the collaborative-based techniques are more applied in practice and have achieved some success, for example the GroupLens system used to recommend news on the Usenet [77] and the item-based collaborative filtering system by the website `www.amazon.com`. Nevertheless, both of the above techniques have their own limitations, which lead to the development of hybrid recommendations. Burke compared seven forms of hybrid recommender systems with their respective design models and performance characteristics [21].

8.2 Exploitation of Domain Knowledge to Improve the Recommendation Quality

One of the main challenges in recommender systems is how to improve their recommendation quality. It has been observed that the percentage of items rated by most users will be very smaller as the number of items increases sharply, which means the users will not have a large number of common ratings on items, and so their pairwise similarities will decline to zero. The nearest neighbour computations are hence impaired because the derived recommendation results based on such skewed similarity values are likely to be wrong. We call this problem as "data sparseness", which mainly impacts the collaborative filtering systems [3].

Recent research has proven that exploiting domain knowledge about items is valuable for addressing this issue, because the item characteristics as well as the user preferences will be better understood with the aid of such knowledge. Dai and Mobasher pointed out in [30] that pure content-based approaches are incapable of capturing complex relationships among objects at the semantic level. They proposed a general framework of integrating domain knowledge within Web Usage Mining for user-based recommendation. Semantic attributes and relations about candidate items are first extracted from webpages based on the ontology. Then "item-level" user profiles are transformed into ontology-enhanced "domain-level" aggregate profiles that capture the common interests of user groups at the domain level. Finally, the aggregated profiles together with the current user profile are used as the input to produce domain-level recommendations, based on which the real web objects are recommended to the user. Later, Mobasher et al. developed an approach for enhanced item-based collaborative filtering in [106]. Structured semantic knowledge about items, extracted automatically from the Web based on the domain-specific ontology, is used for item comparison. The combination of this semantic similarity with the rating similarity provides two advantages: 1) the semantic attributes for items are helpful to infer users' preferences; 2) in case

where little rating information is available, the system can still use semantic similarities to generate reasonable recommendations for users.

Other research achievements of applying domain knowledge in Web Personalization include: Eirinaki et al. applied a concept hierarchy (taxonomy) to semantically annotate web content in [38]. Because all the documents fall into one or more taxonomy categories, this classification enables their personalization system SEWep to recommend documents not only depending on exact keywords matching but also on semantic similarity. Additionally, they utilized the C-logs, a conceptual abstraction of the original Web usage logs based on the website's semantics, as the input of the web mining process to further improve the recommendation quality. Middleton et al. explored the use of an ontology in a recommender system to solve the cold-start problem as well as discover user interests dynamically [101] [102]. They applied the k-NN algorithm to predict the category of candidate papers, track the volatile interests of users by unobtrusively monitoring their behavior history and use ontology-based network analysis to identify the communities of users. The final recommendation result synthesizes all these three factors. Ghani and Faro developed a recommender system based on customer-built knowledge with product semantics [48]. They analyze the product descriptions as well as user behaviors, then automatically infer these semantic attributes to build abstract user models and finally facilitate the personalization by recommending items across categories. Niu et al. built customer profiles based on the product hierarchy in order to learn customer preferences [113].

Anand et al. integrated the user rating vectors with an item ontology to generate more semantic recommendations [4]. The impact factor, implicitly learning from the web usage logs, is introduced to capture the importance of a given concept within the domain ontology on the user's behavior, and so different values of impact factor reflects the user's preferences among different ontological concepts. These impact factors are then utilized within the similarity computation during the procedure of neighbourhood formulation. We will further discuss their recommendation framework in Section 9.1.

Ziegler et al. proposed an approach of exploiting the hierarchical background taxonomy to address the sparseness issue and diversify the recommendation results to better satisfy the spread of the user interests. The user profiles were composed of their interest scores upon the *topics* in the product taxonomy, not their ratings upon products (as in traditional collaborative filtering). The relationships between super-concepts and sub-concepts are exploited in the procedure of constructing the user profiles as well as the identification of a user's neighbourhood. Finally the topic diversification step amends the recommendation results to enhance their utility for the user [164] [165].

8.3 Utilization of Data Mining Techniques to Improve the System Scalability

Various data mining techniques including clustering, similarity indexing, classification and rule induction have been used to improve recommender systems [159]. A state-of-the-art survey of this field was provided in [105], in which the author distinguished the personalization procedure as three phases and explained in detail how the data mining techniques might be applied for each of these phases. Here we mainly focus on the application of clustering techniques as it is most relevant to the research presented in this thesis.

To improve the scalability of recommender systems, Chee et al. developed the data structure RecTree with a divide-and-conquer approach in [132]. Their method iteratively performs the k-means clustering upon users until the number of users in each partition is smaller than a threshold. Then the recommendation algorithm can be conducted within such reduced data space to achieve high scalability and avoid the dilution of suggestion from good users by that from a multitude of poor users. Suryavanshi et al. combined memory-based and model-based collaborative filtering techniques in [140]. They use the k-NN algorithm to identify the neighbours of the current user and generate recommendations based on such small groups of users instead of the whole dataset.

In contrast, Mobasher suggested to group items or users and then employ traditional collaborative filtering algorithms upon these aggregated profiles [105].

Nevertheless, the above approaches are less helpful for improving the system scalability when the domain knowledge about items are considered in the recommendation procedure. For example in our movie datasets, calculating the pairwise similarities within a small set of movies is still time-consuming [4]. A better solution is to construct some kind of data model for preserving item similarities. Recommender systems can then retrieve these values directly instead of computing them online when generating recommendations. Model-based approaches are more suitable for this purpose because the huge amount of candidate items often prevents the utilization of memory-based approaches in practice. In [159] the author reviewed the similarity indexing approaches that can be used to identify users with the same taste or items with the same characteristics. The inverted index (widely used in the community of information retrieval) provides an efficient index structure by mapping the users or items into state vectors, but this technique cannot efficiently resolve queries in the recommender systems.

8.4 Proximity Measure based on Taxonomy

When exploiting the taxonomy to compute the item similarities based on domain knowledge in recommender systems, an important issue is how to define an appropriate measure to evaluate the proximity between nodes (corresponding to concepts or data instances) in the taxonomy. Roughly speaking, there are two categories of proximity measure based on the taxonomic structure:

Edge-based approaches mainly consider the topological structure of the hierarchy. The most intuitive way is to use the length of shortest path between nodes as the similarity measure, which only depends on their relative location in the hierarchy [120]. Wu and Palmer took into account the absolute locations of both nodes as well as that of their Lowest Common Ancestor (LCA) to calculate the similarity [153]. Additional weights for edges might be assigned to reflect the structural characteristics of the hierar-

chy [65]. For example, the similarity value between two objects in SimTree is calculated by multiplying all weights along the path that connects both objects [157].

Node-based approaches are based on the idea of Information Content. Each of the nodes in the hierarchy, i.e. a concept in a taxonomy, contains a certain amount of information quantified by $IC(c) = -\log_2 p(c)$, where $p(c)$ is the probability of encountering an instance of concept c. The root concept has zero information content because all sub-concepts are derived from it. As one moves down the hierarchy, the probability of encountering the instances of a concept decreases, and so the concept's information content or informativeness increases monotonically. The more abstract a concept, the lower its information content and vice versa. Hence, the similarity of two concepts is determined by the information they share, i.e. the information content in their LCA concept. In a text corpus, the concept probability is calculated by maximum likelihood estimation(MLE) based on the concept frequency [126] [127].

The above categories of proximity measure evaluate the semantic similarity from different viewpoints, and so they are suitable for different situations. When an accurate taxonomy is available, the edge-based approaches work better. The node-based approaches outperform in other cases, because they ignore the topological details of the hierarchy. To overcome their individual disadvantages, several hybrid approaches have been proposed to incorporate the information content into the edge-based measure [65] [96]. In comparison, Li et al. proposed a measure that combines the length of the shortest path between two concepts and the location of their LCA within the semantic taxonomy in a non-linear manner [94]. A variant of this measure was later used [24] to personalize web search. In addition, Ganesan et al. reviewed the evolution of proximity measures upon the hierarchical domain structure, analyzed and compared these measures empirically in [46]. They also extended these measures to deal with the multiple occurrence of elements in the taxonomy.

Chapter 9

Exploitation of Taxonomy in Recommender Systems

In this chapter we will first explain a hybrid recommendation framework in Section 9.1. Two methods for improving the system scalability, i.e. building taxonomies upon items and user visits, are investigated in Sections 9.2.1 and 9.2.2 respectively. Then experimental results are analyzed in Section 9.3.

9.1 Framework of Recommender System

Given a set of users $\mathsf{U} = \{\mathsf{u}_i\}$ $(1 \leq i \leq M)$ and a set of items $\mathsf{X} = \{\mathsf{x}_j\}$ $(1 \leq j \leq N)$. In the past visits, each user $\mathsf{u}_i \in \mathsf{U}$ has rated some items $\mathsf{X}_{\mathsf{u}_i} \in \mathsf{X}$ based on his interests. These user ratings can be collected explicitly or implicitly. Explicit rating means users are required to input explicitly their ratings into the recommender system, while implicit rating means the system would track the user visits and find out his preference automatically. We then estimate users' interests by learning a rating function $fr(\mathsf{u}_i, \mathsf{x}_j) \in [0, 1] \cup \{\texttt{NIL}\}$ from the training data. The symbol NIL means the value is undefined, and so $fr(\mathsf{u}_i, \mathsf{x}_j) = \texttt{NIL}$ if $\mathsf{x}_j \notin \mathsf{X}_{\mathsf{u}_i}$. Now let $\mathsf{u}_a \in \mathsf{U}$ denote the *active* user for whom the recommendations will be generated. The set of items $\mathsf{X} - \mathsf{X}_{\mathsf{u}_a}$ are referred

Algorithm 9: CF-based Recommendation

Input: users U, items X, rating function $fr(\cdot,\cdot)$
Output: recommendations $\mathcal{R}(\mathsf{u}_a)$

```
1  begin
2      Construct the rating vector u⃗_i = (r_{i,1}, ..., r_{i,j}, ..., r_{i,N}) for each user u_i;
3      Initialize B(u_a) = ∅;
4      foreach u_i ∈ U − {u_a} do
5          if cos(u_a, u_i) > 0 then
6              B(u_a) ← B(u_a) ∪ {u_i};
7          end
8      end
9      B'(u_a) ← sort_desc(B(u_a));
10     Initialize R(u_a) = ∅;
11     foreach u_i ∈ B'(u_a) do
12         if x_j ∈ X_{u_i} and x_j ∉ X_{u_a} then
13             R(u_a) ← R(u_a) ∪ {x_j};
14         end
15     end
16     return R(u_a);
17 end
```

as the candidate items. Our goal is to select a subset of items $\mathcal{R}(\mathsf{u}_a) \subseteq \mathsf{X} - \mathsf{X}_{\mathsf{u}_a}$ that maximally satisfy the interest of user u_a.

The basic recommendation algorithm using collaborative filtering is described in Algorithm 9. Assume the interest of each user u_i can be represented by a vector of his ratings upon items $\vec{\mathsf{u}}_i = (\mathsf{r}_{i,1}, \ldots, \mathsf{r}_{i,j}, \ldots, \mathsf{r}_{i,N})$, in which $\mathsf{r}_{i,j} = fr(\mathsf{u}_i, \mathsf{x}_j)$. Then the formulation of u_a's neighbourhoods $\mathcal{B}(\mathsf{u}_a) = \{\mathsf{u}_i | \mathsf{u}_i \in \mathsf{U}, \mathsf{u}_i \neq \mathsf{u}_a, \mathsf{X}_{\mathsf{u}_i} \cap \mathsf{X}_{\mathsf{u}_a} \neq \emptyset\}$ is equivalent to find the set $\{\mathsf{u}_i | \cos(\vec{\mathsf{u}}_a, \vec{\mathsf{u}}_i) > 0, \mathsf{u}_i \neq \mathsf{u}_a\}$ (the `NIL` values are ignored in the cosine calculation). Therefore, the final recommendations are achieved as $\mathcal{R}(\mathsf{u}_a) = \{\mathsf{x} | \mathsf{x} \in \cup \mathsf{X}_{\mathsf{u}_i} - \mathsf{X}_{\mathsf{u}_a}, \mathsf{u}_i \in \mathcal{B}(\mathsf{u}_a)\}$. These recommended items are generally sorted according to the cosine value of u_a and their corresponding neighbourhoods.

In Algorithm 9 the cosine function is used to evaluate the similarity between users based on their rating vectors. Due to the existence of data sparseness problem, some users sharing similar interests are not considered as neighbours only because they

do not visit the same items. Therefore, the basic collaborative filtering approach does not perform well in practice. To improve recommendation quality, many researchers now focus on exploiting domain knowledge about items as the supplement of user ratings when generating recommendations [4] [106] [165]. In their works the pair-wise item similarities based on their semantic features are combined with the similarities between user ratings to find the active user's neighbourhood.

Anand et al. proposed a general framework by using the semantically enriched user profiles for movie recommendation [4]. With the relational similarity measure between items defined as Equations 2.8 and 2.9, the pairwise similarities between users are computed by the Generalized Cosine Max (GCM) measure:

$$fs_{user}(\mathsf{u}_i, \mathsf{u}_{i'}) = \sum_{(\mathsf{x}_j, \mathsf{x}_{j'}) \in \mathcal{P}} fr(\mathsf{u}_i, \mathsf{x}_j) \times fr(\mathsf{u}_{i'}, \mathsf{x}_{j'}) \times fs_{item}(\mathsf{x}_j, \mathsf{x}_{j'}) \qquad (9.1)$$

where the candidate set $\mathcal{P} \subseteq \mathsf{X}_{\mathsf{u}_i} \times \mathsf{X}_{\mathsf{u}_{i'}}$ is computed by the procedure FindPair (shown in Algorithm 10). Since the number of candidate items are very large, it is not possible to pre-compute all item similarities and dump them into any storage media. And because the computational expense of evaluating item similarities in real-time based on domain knowledge is substantially high (as we discussed in Chapter 2), only the top-$\hat{n}$ item pairs are used in the recommendation procedure [4].

In addition, the importance of different attributes/relations might be different when describing the items. From the perspective of web personalization, such differences can be interpreted as the personal preference of the users upon the items. Anand et al. suggested that users with different attribute weights in different visits may reflect the context of the user visits, and so they should be dynamically computed and stored in the user profiles [4]. Specifically, set $\{imp_{\mathsf{u,v}}(\mathsf{c})\}$ were used to represent the importance of attribute/relation c in describing the underlying factors that cause the user u to exhibit the observed behaviors within the visit v. To calculate the value of impact imp, the Kullback-Liebler Divergence [35] is applied to measure the divergence of the *observed*

Algorithm 10: FindPair

Input: items $\mathsf{X}_{\mathsf{u}_i}$, items $\mathsf{X}_{\mathsf{u}_{i'}}$, measure fs_{item}, number of pairs $\hat{n}$
Output: set of item pairs $\mathcal{P}$

begin
  Initialize the list $\mathcal{P}_0 \leftarrow \emptyset$;
  Initialize the map $\mathcal{M} \leftarrow \emptyset$;
  foreach $\mathsf{x}_j \in \mathsf{X}_{\mathsf{u}_i}$ **do**
    foreach $\mathsf{x}_{j'} \in \mathsf{X}_{\mathsf{u}_{i'}}$ **do**
      $s \leftarrow fs_{item}(\mathsf{x}_j, \mathsf{x}_{j'})$;
      $\mathcal{P}_0 \leftarrow \mathcal{P}_0 \cup \{s\}$;
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{s \rightarrow (\mathsf{x}_j, \mathsf{x}_{j'})\}$;
    end
  end
  $\mathcal{P}'_0 \leftarrow \mathtt{sort}_{desc}(\mathcal{P}_0)$;
  Initialize $\mathcal{P} \leftarrow \emptyset$, $\mathcal{P}_{\mathsf{u}_i} \leftarrow \emptyset$, $\mathcal{P}_{\mathsf{u}_{i'}} \leftarrow \emptyset$;
  foreach $s \in \mathcal{P}'_0$ **do**
    Retrieve the corresponding $(\mathsf{x}_j, \mathsf{x}_{j'})$ from M;
    if $\mathsf{x}_j \notin \mathcal{P}_{\mathsf{u}_i}$ **and** $\mathsf{x}_{j'} \notin \mathcal{P}_{\mathsf{u}_{i'}}$ **then**
      $\mathcal{P}_{\mathsf{u}_i} \leftarrow \mathcal{P}_{\mathsf{u}_i} \cup \{\mathsf{x}_j\}$;
      $\mathcal{P}_{\mathsf{u}_{i'}} \leftarrow \mathcal{P}_{\mathsf{u}_{i'}} \cup \{\mathsf{x}_{j'}\}$;
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathsf{x}_j, \mathsf{x}_{j'})\}$;
    end
  end
  return *the top-$\hat{n}$ pairs of items from $\mathcal{P}$*;
end

probability distribution p (of data instances in the visit) to an *expected* distribution q by assuming if the concept c has no effect upon the user's preference:

$$imp = \sum_{\mathsf{x} \in \mathsf{X}} p(\mathsf{x}) \log \frac{p(\mathsf{x})}{q(\mathsf{x})} \tag{9.2}$$

Three different definitions of the function $q(\mathsf{x})$ were introduced in [4], which lead to three kinds of impacts:

- If $q(\mathsf{x})$ is assumed to be a uniformed distribution over X, then

$$imp_{\,\mathsf{u}} = 1 - \frac{H[\,p(\mathsf{x})\,]}{\log J} \tag{9.3}$$

  where $H(\cdot)$ is the entropy measure and J is the cardinality of concept c.

- If $q(\mathsf{x})$ is chosen as the distribution $\hat{p}(\mathsf{x})$ that a virtual user likes all the items in X, then

$$imp_{\,\mathsf{i}} = \frac{\sum_{\mathsf{x}} p(\mathsf{x}) \log \frac{p(\mathsf{x})}{\hat{p}(\mathsf{x})}}{-\log[\min \hat{p}(\mathsf{x})]} \tag{9.4}$$

- If the positive and the negative feedback upon items are available, the weight is given by:

$$imp_{\,\mathsf{c}} = \sum_{\mathsf{x},\mathsf{y}} p(\mathsf{xy}) \log \frac{p(\mathsf{xy})}{p(\mathsf{x})\,p(\mathsf{y})} \tag{9.5}$$

  where the variable y indicates whether the user likes or dislikes the item.

Anand et al. compared the above three approaches and found that impact $imp_{\,\mathsf{i}}$ performs the best [4], so in this chapter we keep using $imp_{\,\mathsf{i}}$ for the recommendation generation. The total weight $w(\mathsf{c})$ for concept c is calculated by the impact values for the target visit v_a and for the candidate neighbour visit v_c:

$$w(\mathsf{c}) = \frac{2 \times imp_{\,\mathsf{i}}^{(a)}(\mathsf{c}) \times imp_{\,\mathsf{i}}^{(c)}(\mathsf{c})}{imp_{\,\mathsf{i}}^{(a)}(\mathsf{c}) + imp_{\,\mathsf{i}}^{(c)}(\mathsf{c})} \tag{9.6}$$

The obtained weights are used to calculate similarity between items based on Equa-

tion 2.8, which are in turn used to calculate similarity between users based on Equation 9.1. The cosine-based similarity function for users in Algorithm 9 is now replaced by the knowledge-based similarity function Equation 9.1. Because more domain knowledge are exploited in the computational procedure, the recommendation quality is substantially improved, as shown in [4].

From Equation 9.1 we can see that pairwise similarities between items play an important role in finding the neighborhood of users. Generally speaking, exploiting more domain information during the computation of item similarities can improve the quality of the derived recommendations, but at the same time make the computation more expensive and thus reduce the scalability of the whole system. To avoid such negative impact, our solution is to preserve the item similarities in an offline phase so that the recommender system can retrieve these values directly instead of computing them in the real time. However, the vast number of candidate items makes the exhaustive storage of item similarities practically infeasible, and so we try to integrate the hierarchical taxonomy to simulate the pair-wise item similarities in Section 9.2.1.

9.2 Integration of Taxonomies

In Chapter 6 we have studied how to build a taxonomy to better reflect the data distribution. Taking one step ahead, here we investigate some ways of integrating the derived taxonomy into the recommender systems. First we will use the taxonomy to preserve the pairwise item similarities in Section 9.2.1 so as to alleviate the computational complexity of comparing user visits. Then another taxonomy will be built in Section 9.2.2 for modeling user visits to further improve the system scalability.

9.2.1 Preserve Item Similarities

We wish to preserve the similarities between data instances within the taxonomic structure so as to facilitate the recommendation procedure. Specifically, edges in the taxonomy are assigned different weights and then the "real" similarity values between data

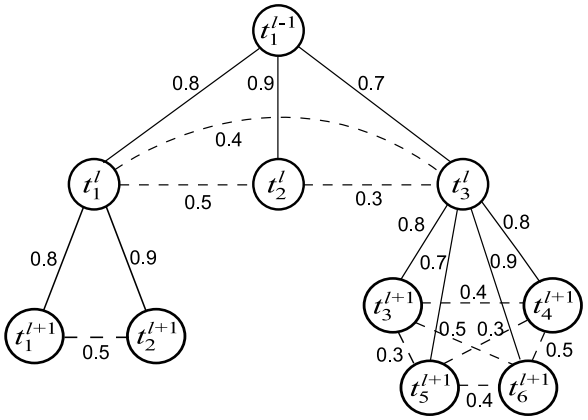


Figure 9.1: Hierarchical taxonomy for preserving similarity values

instances can be estimated by the following *path-based* similarities:

Definition 9.1. Given a weighted taxonomy T, the shortest path connecting two leaf nodes t_1 and t_Q in T is identified by $t_1 \rightarrow \ldots \rightarrow t_q \rightarrow t_{q+1} \rightarrow \ldots \rightarrow t_Q$. The path similarity is calculated by:

$$fs_{\text{path}}(t_1, t_Q) = \prod_{q=1}^{Q} w(t_q, t_{q+1}) \tag{9.7}$$

For example, assume two data instances x_1 and x_2 are contained in node t_1^{l+1} and t_4^{l+1} respectively in Fig. 9.1, the path connecting these two nodes is $t_1^{l+1} \rightarrow t_1^l \rightarrow t_3^l \rightarrow t_4^{l+1}$. Accordingly we can use all the edge weights along this path to estimate the original similarity value between x_1 and x_2: $fs_{\text{path}}(\mathsf{x}_1, \mathsf{x}_2) = 0.8 \times 0.4 \times 0.8 = 0.256$.

Now the question becomes: *How should we assign weights to the taxonomic edges so that the above path-based similarities effectively preserve the original item similarity values computed by Equations 2.8 and 2.9* ? We utilize the average value of item similarities as the weight of edges between nodes, and so:

- The weight of an edge connecting a parent node t_p and its child node t_c is:

$$w(t_p, t_c) = \frac{1}{|t_p| \cdot |t_c|} \sum_{i,j} fs_{item}(\mathsf{x}_i^{(t_p)}, \mathsf{x}_j^{(t_c)}) \tag{9.8}$$

- The weight of an edge connecting two sibling nodes t_c and $t_{c'}$ is:

$$w(t_c, t_{c'}) = \frac{1}{|t_c| \cdot |t_{c'}|} \sum_{i,j} fs_{item}(\mathsf{x}_i^{(t_c)}, \mathsf{x}_j^{(t_{c'})}) \tag{9.9}$$

where $|t_p|$ and $|t_c|$ denote the amount of items in nodes t_p and t_c respectively. By utilizing Equation 9.7 based on the derived taxonomy in the recommender systems, the cost of computing item similarities in real-time will be reduced greatly.

It is worth noting that Yin et al. proposed another data structure, named *SimTree*, in [157] for preserving pair-wise similarities between items, but our method is more appealing because: First, each leaf node in the SimTree corresponds to an item and each non-leaf node has at most c child nodes, where c is a user-specified parameter, and so "unnatural" data structures will be generated in practice if the parameter c is inappropriately specified. Actually finding the optimal value of c is not easy. In contrast, our approach can automatically generate the taxonomic structure that better reflects the real distribution of the data objects. Second, the initial SimTrees are built by using the frequent pattern mining techniques, and then path-based similarities in one SimTree is used to adjust the structures and edge-weights of other SimTrees. Property features of the data objects are not exploited, which will definitely degrade the accuracy of the final result. Our approach is based on the relational similarity measure (Equations 2.8 and 2.9) that exploits the ontological information of items. Therefore, more ontological information and knowledge are preserved in the derived taxonomy.

9.2.2 Group User Visits

Another possible way of improving the systems scalability is to pre-group candidate users in an offline phase based on the visited items. Such user groups can be regarded as "cliques" with different interests. When recommendations are to be generated online for the active user u_a, it is unnecessary to exhaustively examine all the users. Instead only those cliques of users that are most similar to u_a will be considered by the recommendation algorithm.

Formally we can build a taxonomy to organize all the candidate users by applying the ATG technique introduced in Chapter 6. A group of users that are most similar to each other (in the sense of their visiting items) are contained in the same leaf node and different groups of users are arranged in the hierarchy according to their pairwise similarity. Given that each node is represented by a set of ROs, we identify the appropriate leaf node that is closest to the active user by searching the taxonomy from top to bottom. When the number of users in the identified node is less than the required number of neighbours, we recursively exploit the ancestor nodes until accumulating enough neighbour users. The whole procedure is described in the Algorithm 11.

9.3 Experimental Results

We evaluate the performance of our algorithm in comparison with the following similarity calculation approaches: (1) baseline (ORG) which purely uses the original similarity values in the procedure of recommendation generation; (2) cluster model based on real similarity values (CLUS), which uses the original similarity values between two items if they belong to the same cluster or otherwise uses 0 instead; (3) SimTree proposed in [157], which simulates the similarity values based on the synthesized SimTree structure. Finally, the word "ATG-Item" is used to represent our approach of preserving the item similarities by the taxonomy. "ATG-User" and "ATG-Session" represent our approach of grouping users and sessions respectively for the recommendation algorithm.

Algorithm 11: Neighbourhood Formulation

Input: user taxonomy T_{u}, active user u_a, number of neighbourhoods b
Output: u_a's top-b neighbours

begin
  Initialize the result set $U_R \leftarrow \emptyset$;
  Node $t \leftarrow root(T_{\mathsf{u}})$;
  if *t is a leaf node in T_{u}* **then**
    Add all candidate users referenced by t into U_R;
  else
    $t \leftarrow \underset{t_c \in \texttt{GetChildren}(t)}{\arg\max} fs_{\mathsf{node}}(t_c, \mathsf{u}_a)$;
  end
  while $|U_R| < b$ **do**
    foreach $t_s \in \texttt{GetSiblings}(t)$ **do**
      Append all candidate users referenced by t_s into U_R;
    end
    $t \leftarrow \texttt{GetParent}(t)$;
  end
  Order the candidate users in set U_R according to their similarity with respect to u_a;
  return the top-b users in U_R as the neighbourhood of u_a;
end

Two criteria were used to evaluate the accuracy of the recommendations:

1. *Mean Absolute Error* (MAE): When the actual time spent by a user to visit a candidate item is known, the MAE between the predicted and the actual time spent reflects the prediction accuracy of the recommendations.

2. *F-Score*: This is the harmonic mean of precision and recall that have been used widely in the field of Information Retrieval [35]. The higher value of F-Score means a high probability that the items returned by the recommender system will satisfy the user's interest and the potentially interesting items will be discovered by the system at the same time.

Our experiments were conducted on two movie dataset:

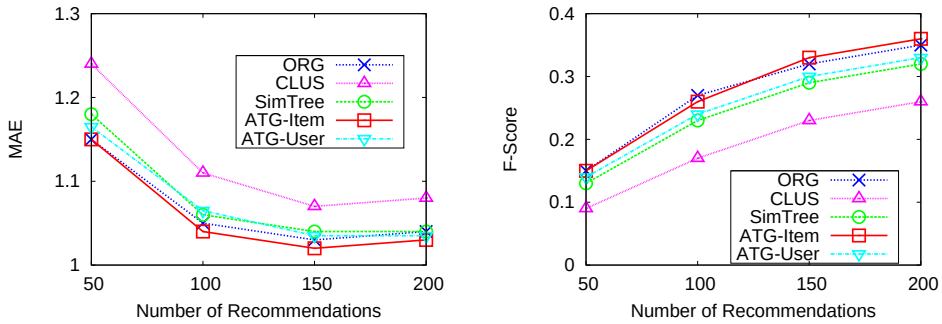
9.3.1 Movie Retailer

The first dataset is provided by an online movie retailer, which has been used in Section 4.5.3. It contains 62,955 movies, 40,826 actors, 9,189 directors as well as a genre taxonomy of 186 genres. In addition, there are 923,987 ratings in 15,741 sessions from 10,151 users. Since this dataset is very sparse compared with the previous one, we randomly select 85% user ratings as the training dataset and the other 15% ones as the test dataset. The number of sessions made by different users ranges from 1 to 814. Based on the user visits, we select 10,000 most popular movies for our analysis. The classic clustering method k-Medoid was adopted in CLUS, given the parameter $K = 50$. In SimTree, we set the parameter c, which controls the number of node's branches, as 15. In our ATG-based approach, the variance v was set as 0.40 for modeling item similarities and 0.10 for grouping user visits, which guarantee the items and users are divided sufficiently for building the taxonomies.

Since the dataset contains multiple visits made by the same user, the recommendations could be generated on the basis of users or visits (sessions). The corresponding experimental results are reported in Table 9.1. For each user or visit, 100 candidate

Table 9.1: Time Spent of Recommendations in Movie Dataset ($\times 10^3$ sec)

	User-based	Session-based
ORG	> 250	> 250
CLUS	13.78	18.61
SimTree	15.19	18.95
ATG-Item	14.22	17.53
ATG-User	8.63	12.48



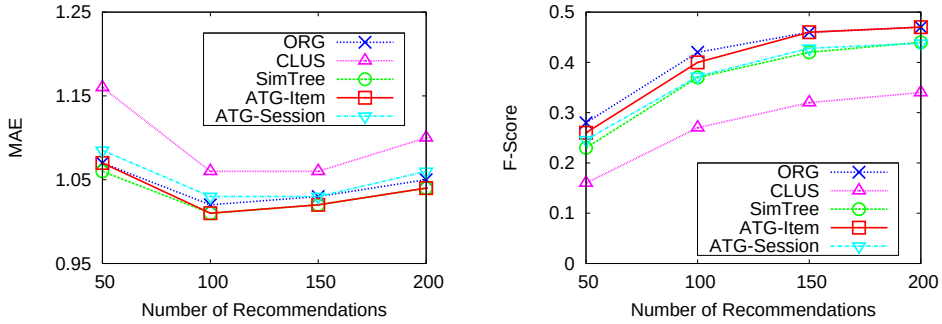
(a) MAE (b) F-Score

Figure 9.2: Recommendations for Movie Dataset based on Users

items were returned in total as the recommendations. Data in Table 9.1 prove that all model-based methods (CLUS, SimTree and ATG-Based) can effectively reduce a magnitude of time compared with that of computing the original similarity measure in real-time (ORG).

These approaches were also evaluated with respect to different recommendation numbers. The curves of MAE and F-Score are shown in Figs. 9.2a and 9.2b respectively. For the MAE criterion, both SimTree and ATG-Item performance close to the baseline ORG, but CLUS has 5% loss compared with the other methods. For the F-Score criterion, ATG-Item approach performs almost the same as ORG, but SimTree has 14.0% and 11.8% loss in the user-based and session-based cases respectively. Again CLUS is the worst among all the approaches.

When examining the trends of recommendation quality in Figs. 9.2a and 9.2b, we can see MAEs of different approaches drop as the increase of the recommendation numbers, but our ATG-Item approach is always better than the others. When evaluated



(a) MAE (b) F-Score

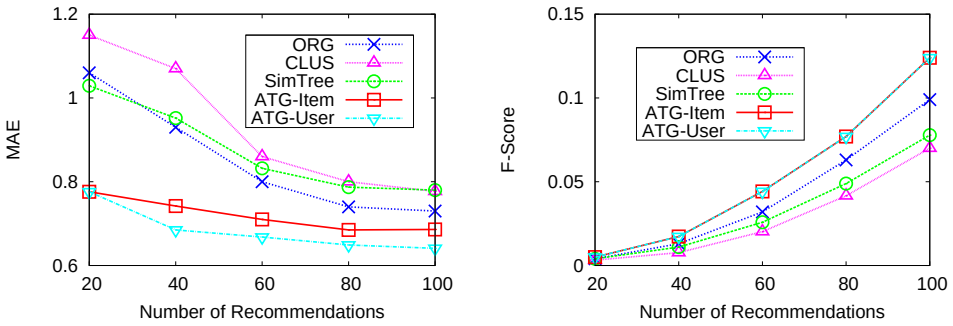
Figure 9.3: Recommendations for Movie Dataset based on Sessions

by F-Score, ATG-Item is very close to ORG under different values of recommendation numbers, while the approaches of CLUS and SimTree have about 30% and 12% loss respectively. As we expect, ATG-User and ATG-Session approaches perform worse than ATG-Item because they examine less users and sessions respectively when formulating the neighbourhood, but they successfully reduced the time spent by the recommender procedure and thus improved the system scalability.

9.3.2 MovieLens

Our second movie dataset is MovieLens, provided by the GroupLens Research at the University of Minnesota [1]. This dataset contains 3,883 movies, 31,650 actors and 1,903 directors. All the movies are assigned into 18 genre categories. In addition, there are 1,000,199 browsing records made by 6,040 users. We randomly select 4810 ($\approx 6040 \times 80\%$) users as the training data and the other 1230 ($\approx 6040 \times 20\%$) users as the test data, and hence the training and test datasets are composed of 788,136 and 212,063 browsing records respectively. In the test dataset, we again randomly label 42537 ($\approx 212063 \times 20\%$) ones as hidden so that they are invisible from the recommendation procedure. Finally, we compare the predicted results with the real (hidden) browsing records.

[1] `http://www.grouplens.org`



(a) MAE (b) F-Score

Figure 9.4: Recommendations for MovieLens Dataset

We keep most experimental parameters the same as those in the previous section: the number of clusters in CLUS is $K = 50$; in SimTree the branch parameter c is 15; in our ATG-based approach the variance v was set as 0.40, which guarantee movies to be divided sufficiently for building the taxonomies. One difference is that the number of recommendations is ranged from 20 to 100, not 50 to 200 as before. It is because the average number of hidden records per user in this case is $\frac{42537}{1230} \approx 35$.

Since it is assumed that each user has only one browsing session in the MovieLens dataset, our experiments are based on users only and the results are presented in Figure 9.4. When evaluated by the MAE criterion, CLUS has approximately 10% loss in comparison with ORG. SimTree performs very close to ORG when $NumRec \leq 60$, but becomes worse afterwards. On the other hand, our ATG-Item and ATG-User approaches achieved more than 7% and 12% improvement in comparison with ORG, especially when $NumRec \leq 60$. When evaluated by the F-Score criterion, CLUS still performs the worse and our ATG-Item and ATG-User approaches again outperform the others in the full range of $NumRec$.

9.4 Summary

In this chapter we utilize our automatic taxonomy generation techniques to facilitate the item similarity computation and the user neighbourhood formulation in recommender systems. Because the online exploitation of domain knowledge about items is very time-consuming, we can build the taxonomy to preserve the pairwise item similarities in an offline phase and later retrieve them directly for the online computation. The user visits are also grouped within another derived taxonomy to effectively reduce the number of candidate users to be considered in the procedure of neighbourhood formulation, which can further improve the scalability of recommender systems.

Chapter 10

Conclusion

Cluster analysis and taxonomy generation are important topics in data mining. In this thesis we introduced some novel ideas to extend them into the analysis of relational datasets and then applied them in the recommender systems. Our contributions are as the follows:

- In the first part we provided a comprehensive review of relational clustering, which reflects the state-of-the-art achievements in this field. The review systematically studied different ideas adopted in relational clustering algorithms and compared them with traditional propositional algorithms to show their great advantages.
- We define the concept of Representative Objects and use it to effectively approximate the distribution of the dataset. Furthermore, we develop several typical strategies to efficiently identify the representative objects in the dataset.
- Based on the representative objects, we design a multi-phase clustering framework for analyzing relational datasets. Two implementations of the framework are also provided that are suitable for different learning tasks.
- In the second part we extend our clustering framework to organize a relational dataset in the form of hierarchical taxonomy so as to help people more easily understand the intrinsic structure of the underlying data.

- Labelling the taxonomic nodes is seldom investigated systematically in previous research. Here we creatively adopt the Kullback-Liebler divergence to quantitatively evaluate the homogeneity of each node and the heterogeneity between sibling nodes, based on which labels are assigned to best summarize the content of each node.

- In the final part we use the automatically derived taxonomy to preserve the pairwise similarities between items, which is then applied to improve the accuracy and scalability of recommender systems when they are integrated with the exploitation of domain knowledge.

- We also use taxonomy generation techniques to analyze user visits and then incorporate the obtained data model to further improve the recommendation efficiency.

Relational cluster analysis and taxonomy generation are promising research directions with great theoretical and practical values. This thesis only addresses the problem from the proximity respective, and so our approaches have advantages such as universal applicability, straight-forwarding and ease of being implemented. Nevertheless, in some cases other clustering algorithms based on more sophisticated ideas might be more suitable. In addition, we introduced several user-specified parameters to control the procedures of clustering and taxonomy generation, and so it is interesting if we can develop some automated mechanisms in the future to find the optimized values for these parameters, which will make our approaches more robust and smart.

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Towards the next generation of recommendation systems: A survey of the state-of-the-art and possible extensions. *IEEE Transaction on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, pages 487–499, Santiago, Chile, Sep 1994. Morgan Kaufmann. ISBN 1-55860-153-8.

[3] Sarabjot S. Anand and Bamshad Mobasher. Intelligent techniques for web personalization. In Bamshad Mobasher and Sarabjot S. Anand, editors, *Intelligent Techniques in Web Personalisation*, volume 3169 of *Lecture Notes in Artificial Intelligence*, pages 1–36. Springer-Verlag, 2005.

[4] Sarabjot S. Anand, Patricia Kearney, and Mary Shapcott. Generating semantically enriched user profiles for web personalization. *ACM Transactions on Internet Technologies*, 7(4): 22–42, August 2007. ISSN 1533–5399.

[5] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on management of data (SIGMOD'99)*, pages 49–60, New York, NY, USA, 1999. ACM. ISBN 1-58113-084-8. doi: http://doi.acm.org/10.1145/304182.304187.

[6] Adam Anthony and Marie desJardins. Open problems in relational data clustering. In *Proceedings of ICML Workshop on Open Problems in Statistical Relational Learning*, Pittsburgh, PA, 2006.

[7] Arthor Asuncion and David J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[8] Eric L. Backer and Anil K. Jain. Clustering performance measure based on fuzzy set decomposition. PAMI-3(1):66–75, Jan 1981.

[9] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. ISBN 0-201-39829-X.

[10] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/245108.245124.

[11] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005. ISSN 1533-7928.

[12] Shenghua Bao, Yunbo Cao, Bing Liu, Yong Yu, and Hang Li. Mining latent associations of objects using a typed mixture model–a case study on expert/expertise mining. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 803–807, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2701-9. doi: http://dx.doi.org/10.1109/ICDM.2006.109.

[13] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[14] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, USA, 2002.

[15] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713.

[16] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning (ICML'04)*, pages 11–18, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-5. doi: http://doi.acm.org/10.1145/1015330.1015360.

[17] Jeff A. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, University of California, Berkeley, 1997.

[18] Daniel Boley. Hierarchical taxonomies using divisive partitioning. Technical Report TR-98-012, Department of Computer Science, University of Minnesota, Minneapolis, 1998.

[19] Jean-François Boulicaut and Baptiste Jeudy. Constraint-based data mining. In Oded Maimon and Lior Rokach, editors, *The Data Mining and Knowledge Discovery Handbook*, pages 399–416. Springer, 2005. ISBN 0-387-24435-2.

[20] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors. *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer, Berlin, 2007.

[21] Robin Burke. Hybrid systems for personalized recommendations. In Bamshad Mobasher and Sarabjot S. Anand, editors, *Intelligent Techniques in Web Personalisation*, volume 3169 of *Lecture Notes in Artificial Intelligence*, pages 133–152. Springer-Verlag, 2005.

[22] Pu-Jen Cheng and Lee-Feng Chien. Auto-generation of topic hierarchies for web images from users' perspectives. In *Proceedings of the twelfth international conference on Information and knowledge management (CIKM'03)*, pages 544–547, New York, NY, USA, 2003. ACM. ISBN 1-58113-723-0. doi: http://doi.acm.org/10.1145/956863.956969.

[23] Pu-Jeng Cheng, Ching-Hsiang Tsai, Chen-Ming Hung, and Lee-Feng Chien. Query taxonomy generation for web search. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM'06)*, pages 862–863, New York, NY, USA, 2006. ACM. ISBN 1-59593-433-2. doi: http://doi.acm.org/10.1145/1183614.1183768.

[24] Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. Using odp metadata to personalize search. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'05)*, pages 178–185, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: http://doi.acm.org/10.1145/1076034.1076067.

[25] Shui-Lung Chuang and Lee-Feng Chien. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, pages 75–82, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1754-4.

[26] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing conceptual, partitional and agglomerative clustering for learning taxonomies from text. In *Proceedings of European Conference on Artificial Intelligence (ECAI'04)*, pages 435–439. IOS Press, 2004.

[27] Patrick Clerkin, Padraig Cunningham, and Conor Hayes. Ontology discovery for the semantic web using hierarchical clustering. In *Proceedings of Semantic Web Mining Workshop co-located with ECML/PKDD*, Freiburg, Germany, Sep 2001.

[28] Edgar Frank Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[29] Paolo Corsini, Beatrice Lazzerini, and Francesco Marcelloni. A new fuzzy relational clustering algorithm based on the fuzzy c-means algorithm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9(6):439–447, 2005. ISSN 1432-7643. doi: http://dx.doi.org/10.1007/s00500-004-0359-6.

[30] Honghua Dai and Bamshad Mobasher. A road map to more effective web personalization: Integrating domain knowledge with web usage mining. In *Proceedings of International Conference on Internet Computing (IC'03)*, pages 58–64, 2003.

[31] Belur V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Computer Society Press, 1991.

[32] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transaction on Information Systems*, 22(1):143–177, 2004. ISSN 1046-8188. doi: http://doi.acm.org/10.1145/963770.963776.

[33] Thomas G. Dietterich and Ryszard S. Michalski. A comparative review of selected methods for learning from examples. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 41–81. Springer, Berlin, Heidelberg, 1984.

[34] Richard C. Dubes. Cluster analysis and related issues. In *Handbook of pattern recognition & computer vision*, pages 3–32. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1993. ISBN 981-02-1136-8.

[35] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience Publication, 2001.

[36] Sašo Džeroski. Multi-relational data mining: An introduction. *SIGKDD Explorations Newsletter*, 5(1):1–16, 2003. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/959242.959245.

[37] Sašo Džeroski and Nada Lavrač. *Relational Data Mining*. Springer, September 2001. ISBN 3-540-42289-7.

[38] Magdalini Eirinaki, Michalis Vazirgiannis, and Iraklis Varlamis. SEWeP: Using site semantics and a taxonomy to enhance the web personalization process. In *Proceedings of the 9th SIGKDD Conference*, pages 99–108, New York, USA, 2003.

[39] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiao-Wei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[40] Brian Everitt and Torsten Hothorn. *A Handbook of Statistical Analyses Using R*. Chapman & Hall/CRC, Boca Raton, FL, 2006. ISBN 1-584-88539-4.

[41] Brian Everitt, Landau Sabine, and Leese Morven. *Cluster Analysis (4th Edition)*. Hodder Arnold, 2001. ISBN 0-340-76119-9.

[42] Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/34.990138.

[43] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022852608280.

[44] Dániel Fogaras and Balázs Rácz. Scaling link-based similarity search. In *Proceedings of the 14th Int'l World Wide Web Conference*, pages 641–650, Chiba, Japan, 2005. ACM.

[45] Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis, and density estimation. Technical Report 380, University of Washington, Dept. of Stats., Oct 2000.

[46] Prasanna Ganesan, Hector Garcia-Molina, and Jennifer Widom. Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems (TOIS)*, 21(1):64–93, 2003. ISSN 1046-8188.

[47] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explorations Newsletter*, 7(2):3–12, 2005. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/1117454.1117456.

[48] Rayid Ghani and Andrew Fano. Building recommender systems using a knowledge base of product semantics. In *Proceedings of the Workshop on Recommendation and Personalization in e-Commerce, at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems*, 2002.

[49] Mark Girolami. Mercer kernel-based clustering in feature space. 13(4):780–784, 2002.

[50] Mark A. Gluck and James E. Corter. Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference on Artificial Intelligence*, pages 831–836, Detroit, MI, USA, 1985. Morgan Kaufmann.

[51] Philip W. Goetz, Robert Mchenry, and Dale Hoiberg, editors. *Encyclopedia Britannica 2009*. Britannica Inc, Chicago, IL, USA, 2009.

[52] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, June 1998.

[53] Lars W. Hagen and Andrew B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992. ISSN 0278-0070.

[54] Eui-Hong (Sam) Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Hypergraph based clustering in high-dimensional data sets: A. summary of results. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pages 15–22. 1997.

[55] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques (2nd Edition)*. Morgan Kaufmann, 2006.

[56] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.

[57] Richard J. Hathaway, John W. Davenport, and James C. Bezdek. Relational dual of the c-means clustering algorithms. *Pattern Recognition*, 22(2):205–212, 1989.

[58] Thomas Hofmann and Jan Puzicha. Statistical models for co-occurrence data. Technical Report AIM-1625, 1998.

[59] Frank Höppner, Frank Klawonn, Rudolf Kruse, and Thomas Runkler. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. Wiley, 1999.

[60] Tamás Horváth, Stefan Wrobel, and Uta Bohnebeck. Relational instance-based learning with lists and terms. *Mach. Learn.*, 43(1-2):53–80, 2001. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1007668716498.

[61] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. ISBN 0-13-022278-X.

[62] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[63] Glen Jeh and Jennifer Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'02)*, pages 538–543, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: http://doi.acm.org/10.1145/775047.775126.

[64] David Jensen. Statistical challenges to inductive inference in linked data. In *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*, 1999.

[65] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computational Linguistics (ROCLING X)*, 1997.

[66] David R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-out algorithm. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms (SODA'93)*, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0-89871-313-7.

[67] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in VLSI design. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 526–529, 1997.

[68] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[69] Vipul Kashyap, Cartic Ramakrishnan, Christopher Thomas, and A. Sheth. TaxaMiner: an experimentation framework for automated taxonomy bootstrapping. *Int. J. Web Grid Serv.*, 1(2):240–266, 2005. ISSN 1741-1106. doi: http://dx.doi.org/10.1504/IJWGS.2005.008322.

[70] Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, 1990.

[71] Mathias Kirsten and Stefan Wrobel. Extending k-means clustering to first-order representations. In *Proceedings of the 10th International Conference on Inductive Logic Programming (ILP'00)*, pages 112–129, London, UK, 2000. Springer-Verlag. ISBN 3-540-67795-X.

[72] Mathias Kirsten and Stefan Wrobel. Relational distance-based clustering. In *Proceedings of Fachgruppentreffen Maschinelles Lernen (FGML-98)*, pages 119–124, 10587 Berlin, 1998. Techn. Univ. Berlin, Technischer Bericht 98/11.

[73] Mathias Kirsten, Stefan Wrobel, and Tamá Horváth. Distance based approaches to relational learning and clustering. In *[37]*, pages 213–232. 2001.

[74] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pages 307–314, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7.

[75] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[76] Arno J. Knobbe, Hendrik Blockeel, Arno P. J. M. Siebes, and D. M. G. van der Wallen. Multi-relational data mining. Technical Report INS-R9908, Information Systems (INS), 1999.

[77] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R.Gordon, and John Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communication of the ACM*, 40(3):77–87, 1997.

[78] Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In *Relational Data Mining*, pages 262–286. 2001.

[79] Kumar M. Krishna and M. Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29 Volume(3):433 – 439, June 1999.

[80] Raghu Krishnapuram and Krishna Kummamuru. Automatic taxonomy generation: Issues and possibilities. In *Proceedings of Fuzzy Sets and Systems (IFSA)*, LNCS 2715, pages 52–63. Springer-Verlag Heidelberg, 2003.

[81] Raghu Krishnapuram, Anupam Joshi, Olfa Nasraoui, and Liyu Yi. Low-complexity fuzzy relational clustering algorithms for web mining. *IEEE Transactions on Fuzzy Systems*, 9: 595–607, Aug. 2001. doi: 10.1109/91.940971.

[82] Solomon Kullback. The kullback-leibler distance. *The American Statistician*, 41:340–341, 1987.

[83] Krishna Kummamuru, Rohit Lotlikar, Shourya Roy, Karan Singal, and Raghu Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th international conference on World Wide Web (WWW'04)*, pages 658–665, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988762.

[84] Nada Lavrač, Filip Železný, and Sašo Džeroski. Local patterns: Theory and practice of constraint-based relational subgroup discovery. In Katharina Morik, Jean-François Bouli-

caut, and Arno Siebes, editors, *Local Pattern Detection, International Seminar*, LNCS 3539, Germany, 2005. Springer. ISBN 978-3-540-26543-6. doi: 10.1007/11504245_5.

[85] Dawn Lawrie, W. Bruce Croft, and Arnold Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'01)*, pages 349–357, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6. doi: http://doi.acm.org/10.1145/383952.384022.

[86] Dawn J. Lawrie and W. Bruce Croft. Generating hierarchical summaries for web searches. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (SIGIR'03)*, pages 457–458, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3. doi: http://doi.acm.org/10.1145/860435.860549.

[87] Vladimir I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[88] Tao Li and Sarabjot S. Anand. HIREL: An incremental clustering algorithm for relational datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM'08)*, pages 887–892, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3502-9. doi: http://dx.doi.org/10.1109/ICDM.2008.116.

[89] Tao Li and Sarabjot Singh Anand. DIVA: A variance-based clustering approach for multi-type relational data. In *Proceedings of the ACM Sixteenth Conference on Information and Knowledge Management (CIKM'07)*, pages 147–156, Lisboa, Portugal, 2007.

[90] Tao Li and Sarabjot Singh Anand. Automated taxonomy generation for summarizing multi-type relational datasets. In *Proceedings of 2008 International Conference in Data Mining (DMIN'08) joint with the 2008 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'08)*, pages 571–577, Las Vegas, Nevada, USA, 2008.

[91] Tao Li and Sarabjot Singh Anand. Labeling nodes of automatically generated taxonomy for multi-type relational datasets. In *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'08)*, pages 317–326, Turin, Italy, 2008.

[92] Tao Li and Sarabjot Singh Anand. Multi-type relational clustering approaches: Current state-of-the-art and new directions. In *Proceedings of the International Conference on Intelligent Systems and Networks (IISN'08)*, India, 2008.

[93] Tao Li and Sarabjot Singh Anand. Incorporating automated taxonomy generation for relational datasets to improve recommender systems. In *Proceedings of the 10th International Conference on Electronic Commerce and Web Technologies (EC-Web 09)*, pages 120–131, Linz, Austria, 2009.

[94] Yuhua Li, Zuhair A. Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transaction on Knowledge and Data Engineering*, 15(4):871–882, 2003. ISSN 1041-4347. doi: http://dx.doi.org/10.1109/TKDE.2003.1209005.

[95] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36:451–461, 2001.

[96] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[97] Guowei Liu, Weibin Zhu, and Yong Yu. A unified probabilistic framework for clustering correlated heterogeneous web objects. In *Proceeding of the seventh Asia-Pacific Web Conference (APWeb2005)*, pages 76–87, Shanghai, China, March 2005.

[98] Xin Liu, Yi-Hong Gong, Wei Xu, and Sheng-Huo Zhu. Document clustering with cluster refinement and model selection capabilities. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'02)*, pages 191–198, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0. doi: http://doi.acm.org/10.1145/564376.564411.

[99] Bo Long, Xiaoyun Wu, Zhongfei (Mark) Zhang, and Philip S. Yu. Unsupervised learning on k-partite graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06)*, pages 317–326, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: http://doi.acm.org/10.1145/1150402.1150439.

[100] Bo Long, Zhongfei (Mark) Zhang, Xiaoyun Wu, and Philip S. Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning (ICML'06)*, pages 585–592, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-383-2. doi: http://doi.acm.org/10.1145/1143844.1143918.

[101] Stuart E. Middleton, Harith Alani, Nigel Shadbolt, and David C. De Roure. Exploiting synergy between ontologies and recommender systems. In *Proceedings of Semantic Web Workshop on WWW2002*, 2002.

[102] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004. ISSN 1046-8188. doi: http://doi.acm.org/10.1145/963770.963773.

[103] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[104] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982. doi: http://dx.doi.org/10.1016/0004-3702(82)90040-6.

[105] Bamshad Mobasher. Data mining for personalization. In *[20]*, chapter 3, pages 90–135. 2007.

[106] Bamshad Mobasher, Xin Jin, and Yanzan Zhou. Semantically enhanced collaborative filtering on the web. In B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopolou, and G. Stumme, editors, *Web Mining: From Web to Semantic Web*, volume 3209 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2004.

[107] Francisco Moreno-Seco, María Luisa Micó, and José Oncina. Extending LAESA fast nearest neighbour algorithm to find the k-nearest neighbours. *Lecture Notes in Computer Science*, 2396:718–724, 2002.

[108] Adrian Muller, Jochen Dorre, Peter Gerstl, and Roland Seiffert. The TaxGen framework: Automating the generation of a taxonomy for a large document collection. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, volume 2, page 2034, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0001-3.

[109] Maurice D. Mulvenna, Sarabjot S. Anand, and Alex G. Büchner. Personalization on the net using web mining: Introduction. *Communication of ACM*, 43(8):122–125, 2000. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/345124.345165.

[110] Jennifer Neville, Micah Adler, and David Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop, 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.

[111] Andrew Y. Ng, Michael I. Jordan, Yair Weiss, Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proceeding of 14th Advances in Neural Information Processing Systems*, pages 849–856. MIT Press, 2001.

[112] Raymond T. Ng and Jiawei Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002. ISSN 1041-4347. doi: http://dx.doi.org/10.1109/TKDE.2002.1033770.

[113] Li Niu, Xiao-Wei Yan, Cheng-Qi Zhang, and Shi-Chao Zhang. Product hierarchy-based customer profiles for electronic commerce recommendation. In *Proceedings of the 1st International Conference on Machine Learning and Cybernetics*, volume 2, pages 1075–1080, 2002.

[114] Mark O'Connor and Jon Herlocker. Clustering items for collaborative filtering. In *Proceedings of SIGIR-2001 Workshop on Recommender Systems*, 2001.

[115] Jonathan J. Oliver, Rohan A. Baxter, and Chris S. Wallace. Unsupervised learning using MML. In *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, pages 364–372, 1996.

[116] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *[20]*, chapter 10, pages 325–341. 2007.

[117] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, pages 727–734, San Fracisco, USA, 2000. Morgan Kaufmann.

[118] Alex Pothen, Horst D. Simon, and Kan-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990. ISSN 0895-4798. doi: http://dx.doi.org/10.1137/0611030.

[119] John Ross Quinlan. Induction of decision trees. In *Maching Learning*, volume 1, pages 81–106. Springer, Hingham, MA, USA. doi: http://dx.doi.org/10.1023/A:1022643204877.

[120] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, pages 17–30, 1989.

[121] Luc De Raedt. *Logical and Relational Learning: From ILP to MRDM (Cognitive Technologies)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2008. ISBN 3540200401.

[122] Jan Ramon. *Clustering and instance based learning in first order logic*. PhD thesis, Katholieke Universiteit Leuven, 2003.

[123] Jan Ramon and Maurice Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37(10):765–780, 2001.

[124] Thomas Reinartz. A unified view on instance selection. *Data Mining and Knowledge Discovery*, 6:191–210, 2002.

[125] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstorm, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[126] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

[127] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[128] Mark Roubens. Pattern classification problems and fuzzy sets. *Fuzzy Sets and Systems*, 1:239–253, 1978.

[129] Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, New York, US, 1987.

[130] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International*

World Wide Web Conference, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: http://doi.acm.org/10.1145/371920.372071.

[131] J. Ben Schafer, Dan Frankowshi, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *[20]*, chapter 9, pages 291–324. 2007.

[132] Sonny Han Seng, Seng Chee, Jiawei Han, and Ke Wang. RecTree: An efficient collaborative filtering method. In *Lecture Notes in Computer Science*, pages 141–151. Springer Verlag, 2001.

[133] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 625–56, 1948.

[134] Roded Sharan and Ron Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB'00)*, pages 307–316, Menlo Park, CA, 2000. AAAI Press.

[135] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[136] Shun-Hong Sie and Jian-Hua Yeh. Automatic ontology generation using schema information. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)*, pages 526–531, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2747-7. doi: http://dx.doi.org/10.1109/WI.2006.48.

[137] Benno Stein and Oliver Niggemann. On the nature of structure and its identification. In *Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'99)*, pages 122–134, London, UK, 1999. Springer-Verlag. ISBN 3-540-66731-8.

[138] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *Proceedings of KDD Workshop on Text Mining*, 2000.

[139] Petre Stoica and Randolph L. Moses. *Introduction to spectral analysis*. Upper Saddle River, N.J. : Prentice Hall, 1997.

[140] Bhushan Shankar Suryavanshi, Nematollaah Shiri, and Sudhir P. Mudur. A fuzzy hybrid collaborative filtering technique for web personalization. In *Proceedings of Intelligent Techniques for Web Personalization (ITWP) Workshop co-located with IJCAI*, 2005.

[141] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introdution to data mining*. Addison-Wesley, 2005.

[142] Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In Bernhard Nebel, editor, *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 870–878, Seattle, US, 2001.

[143] Enrique Vidal. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (AESA). *Pattern Recogn. Lett.*, 15(1):1–7, 1994. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/0167-8655(94)90094-9.

[144] Filip Železný, Nada Lavrač, and Sašo Džeroski. Constraint-based relational subgroup discovery. In *Proceedings of 2th ACM SIGKDD Workshop on Multi-Relational Data Mining (MRDM)*, Washington DC, USA, 2003.

[145] Jidong Wang, Huajun Zeng, Zheng Chen, Hongjun Lu, Tao Li, and Wei-Ying Ma. ReCoM: Reinformcement clustering of multi-type interrelated data objects. In *Proceedings of the 26th ACM SIGIR conference on Research and development in informaion retrieval (SIGIR'03)*, pages 274–281, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-646-3. doi: http://doi.acm.org/10.1145/860435.860486.

[146] Yair Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the 2nd International Conference on Computer Vision (ICCV)*, pages 975–982, 1999.

[147] Douglas Brent. West. *Introduction to graph theory (2nd Edition)*. Upper Saddle River, N.J. : Prentice Hall, 2001.

[148] Scott White and Padhraic Smyth. Algorithms for estimating relative importance in networks. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'03)*, pages 266–275, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: http://doi.acm.org/10.1145/956750.956782.

[149] Michael P. Windham. Numerical classification of proximity data with assignment measure. *Journal of Classification*, 2:157–172, 1985.

[150] Michael P. Windham and Adele Cutler. Information ratio for validating mixture analyses. *Journal of American Statistical Association*, 87(420):1188–1192, 1992.

[151] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition)*. Morgan Kaufmann, June 2005. ISBN 0-12-088407-0.

[152] Janusz Wnek. Lsi-based taxonomy generation: The taxonomist system. In Paul B. Kantor, Gheorghe Muresan, Fred Roberts, Daniel Dajun Zeng, Fei-Yue Wang, Hsinchun Chen, and Ralph C. Merkle, editors, *ISI*, volume 3495 of *Lecture Notes in Computer Science*, pages 389–394. Springer, 2005. ISBN 3-540-25999-6.

[153] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, New Mexico State University, Las Cruces, New Mexico, 1994.

[154] Wensi Xi, Edward A. Fox, Weiguo Fan, Benyu Zhang, Zheng Chen, Jun Yan, and Dong Zhuang. SimFusion: measuring similarity using unified relationship matrix. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'05)*, pages 130–137, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: http://doi.acm.org/10.1145/1076034.1076059.

[155] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transaction on Neural Networks*, 16:645–678, May 2005.

[156] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Yong Yu, Wei-Ying Ma, WenSi Xi, and Edward Fox. MRSSA: an iterative algorithm for similarity spreading over interrelated objects. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM'04)*, pages 240–241, New York, NY, USA, 2004. ACM. ISBN 1-58113-874-1. doi: http://doi.acm.org/10.1145/1031171.1031222.

[157] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. LinkClus: efficient clustering via heterogeneous semantic links. In *Proceedings of the 32nd international conference on very large data bases (VLDB'06)*, pages 427–438. VLDB Endowment, 2006. ISBN 1595933859.

[158] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. CrossClus: user-guided multi-relational clustering. *Data Mining and Knowledge Discovery*, 15(3):321–348, 2007. ISSN 1384-5810. doi: http://dx.doi.org/10.1007/s10618-007-0072-z.

[159] Philip S. Yu. Data mining and personalization technologies. In *Proceedings of the Sixth International Conference on Database Systems for Advanced Applications (DASFAA'99)*, pages 6–13, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0084-6.

[160] Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. A unified framework for clustering heterogeneous web objects. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE'02)*, pages 161–172, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1766-8.

[161] Hong-Yuan Zha, Chris Ding, Ming Gu, Xiaofeng He, and Horse Simon. Spectral relaxation for k-means clustering. In *Proceedings of the 14th Advances in Neural Information Processing Systems (NIPS'01)*, pages 1057–1064, 2001.

[162] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, 1996.

[163] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 11th international conference on Information and knowledge management (CIKM'02)*, pages 515–524, New York, NY, USA, 2002. ACM. ISBN 1-58113-492-4. doi: http://doi.acm.org/10.1145/584792.584877.

[164] Cai-Nicolas Ziegler, Georg Lausen, and Lars Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM'04)*, pages 406–415, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-874-1. doi: http://doi.acm.org/10.1145/1031171.1031252.

[165] Cai-Nicolas Ziegler, Sean M. Mcnee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web (WWW'05)*, pages 22–32, New York, NY, USA, 2005. ACM Press. ISBN 1595930469. doi: 10.1145/1060745.1060754.