# A GRAPH-BASED APPROACH FOR SEMANTIC DATA MINING

by

HAISHAN LIU

**COMMITTEE DRAFT**

**THIS IS A DRAFT COPY FOR REVIEW AND EDITING PURPOSES ONLY!**

**This draft format in no way complies with the University of Oregon Style Graduate School Style Guide for theses or dissertations. This draft may employ 10 point fonts for paper conservation, line numbering for ease of editing, or watermarking and editing notations; all of which violate the Graduate School Style Guide for final documents. See the uothesis.cls documentation for instructions on producing a properly formatted document.**



A DISSERTATION

Presented to the Dept. of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

March 2012

DISSERTATION APPROVAL PAGE

Student: Haishan Liu

Title: A Graph-based Approach For Semantic Data Mining

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Dept. of Computer and Information Science by:

| | |
|---|---|
| Dr. Dejing Dou | Chair |
| Dr. Dejing Dou | Advisor |
| Dr. Arthur Farley | Inside |
| Dr. Allen Malony | Inside |
| Dr. Daniel Lowd | Inside |
| Dr. Don Tucker | Ouside |

and

| | |
|---|---|
| Richard Linton | Vice President for Research & Innovation/ Dean of the Graduate School |

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded March 2012

DISSERTATION ABSTRACT

Haishan Liu

Doctor of Philosophy

Department of Computer and Information Science

March 2012

Title: A Graph-based Approach For Semantic Data Mining

Approved: _____
Dr. Dejing Dou

ipsum lorem

CURRICULUM VITAE

NAME OF AUTHOR: Haishan Liu

PLACE OF BIRTH: Shanghai

DATE OF BIRTH: October 12th 1983

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:
University of Oregon, Eugene, OR
Shanghai Jiao Tong University, Shanghai, China

DEGREES AWARDED:
Master of Science in Computer Science, 2008, University of Oregon
Bachelor of Information Security, 2006, Shanghai Jiao Tong University.

AREAS OF SPECIAL INTEREST:
Data mining, machine learning, the Semantic Web

PROFESSIONAL EXPERIENCE:

GRANTS, AWARDS AND HONORS:

Graduate Teaching & Research Fellowship, Computer and Information Science, 2006 to present

NSF Travel Grant, International Semantic Web Conference, 2010

NSF Travel Grant, International Conference on Data Mining, 2011

PUBLICATIONS:

Haishan Liu, Paea LePendu, Ruoming Jin, and Dejing Dou. A Hypergraph-based Method for Discovering Semantically Associated Itemsets. In Proceedings of the 11th IEEE International Conference on Data Mining (ICDM), 2011.

Haishan Liu and Dejing Dou. Breaking the Deadlock: Simultaneously Discovering Attribute Matching and Cluster Matching with Multi-Objective Simulated Annealing. In Proceedings of the International Conference on Ontologies, Databases and Application of Semantics (ODBASE), pages 698–715, 2011.

Dejing Dou, Han Qin, and Haishan Liu. Semantic Translation for Rule-based Knowledge in Data Mining. In Proceedings of the 22nd International Conference on Database and Expert Systems Applications (DEXA), part II, pages 74–89, 2011.

Haishan Liu. Towards Semantic Data Mining. In: Doctoral Consortium of the 9th International Semantic Web Conference (ISWC), November 2010.

Christopher Townsend, Jingshan Huang, Dejing Dou, Shivraj Dalvi, Patrick J. Hayes, Lei He, Wen-chang Lin, Haishan Liu, Robert Rudnick and Hardik Shah. OMIT: Domain Ontology and Knowledge Acquisition in MicroRNA Target Prediction. On-The-Move Conferences 2010 (OTM): 1160–1167

Haishan Liu, Gwen Frishkoff, Robert Frank, and Dejing Dou. Ontology-based Mining of Brainwaves: A Sequence Similarity Technique for Mapping Alternative Descriptions of Patterns in Event Related Potentials (ERP) Data. In Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pages 43–54, 2010.

Gwen Frishkoff, Paea LePendu, Robert Frank, Haishan Liu, and Dejing Dou. Development of Neural Electromagnetic Ontologies (NEMO): Ontology-based Tools for Representation and Integration of Event-related Brain Potentials. In Proceedings of the International Conference on Biomedical Ontology (ICBO), pages 31–34, 2009.

Haishan Liu and Dejing Dou. An Exploration of Understanding Heterogeneity through Data Mining. In Proceedings of KDD'08 Workshop on Mining Multiple Information Sources, pages 18–25, 2008.

Jongwan Kim, Dejing Dou, Haishan Liu, and Donghwi Kwak. Constructing a user preference ontology for anti-spam mail systems. In Proceedings of the 20th Canadian Conference on Artificial Intelligence (Canadian AI), pages 272–283, 2007.

ACKNOWLEDGEMENTS

**COMMITTEE DRAFT**

**THIS IS A DRAFT COPY FOR REVIEW AND EDITING PURPOSES ONLY!**

**This draft format in no way complies with the University of Oregon Style Graduate School Style Guide for theses or dissertations. This draft may employ 10 point fonts for paper conservation, line numbering for ease of editing, or watermarking and editing notations; all of which violate the Graduate School Style Guide for final documents. See the uothesis.cls documentation for instructions on producing a properly formatted document.**

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

Table Page

Table Page

# CHAPTER I

# INTRODUCTION

Data mining, also referred to as knowledge discovery in databases (KDD), is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [**?** ]. But the measure of what is meant by "useful" to the user is dependent on the user as well as the domain within which the data mining system is being used. Therefore, the role of domain knowledge in the discovery process is essential. Fayyad *et al.* [**?** ] contended that the use of domain knowledge is important in all stages of the data mining process including, for example, data transformation, feature reduction, algorithm selection, post-processing, model interpretation and so forth.

The first step towards using domain knowledge is to acquire the knowledge from experts and thus model and codify the knowledge in computer. Russell and Norvig [**?** ] emphasized that a data mining system must have some method for obtaining the background knowledge and can no longer make naive speculations, and should use its background knowledge to learn more and more effectively. this process of modeling knowledge in computer systems to facilitate problem solving is studied in the field of knowledge representation/engineering. Research in this field has resulted in many sophisticated technologies such as expert system. However, at present, knowledge representation and data mining remain largely separate disciplines. Although it is widely stated that exploring the fusion of the two fields is worthwhile in many applications where substantial human expertise exists alongside data resources, as many researchers have pointed out, work along this line is still in its infancy [**? ? ? ? ?** ].

Previous research has made limited attempts to build data mining systems that are capable of incorporating domain knowledge in a principled manner. In those systems

where users are allowed to express domain knowledge, such knowledge is often specified in a very application-specific form, scope and granularity. The ad hoc manner of representing knowledge hinders the interoperability and reuse of codified knowledge [**?** ]. Moreover, we have also observed that in many data mining systems, domain knowledge is embedded into the implementations in the form of implicit assumptions and heuristics, which makes abstraction and refactoring particularly hard. Sometimes even data mining practitioners themselves are not aware of employing domain knowledge in such a manner. This motivates us to develop a framework for explicit incorporation of domain knowledge in a data mining system so that insights can be drawn from both knowledge and data in a systematic and holistic way. We call such technology the semantic data mining.

Latest research in knowledge representation, particularly in the area of W3C's Semantic Web [**?** ] that seeks to embed semantic content in web pages, has led to mature standards such as the Web Ontology Language (OWL [**?** ]) for authoring ontologies. An ontology is an explicit specification of a conceptualization [**?** ]. Today, Semantic Web ontologies have become a key technology for intelligent knowledge processing, providing a framework for sharing conceptual models about a domain [**?** ]. Ontologies explicate domain knowledge hence providing a way to separate knowledge from implementations [**?** ]. Much effort has been devoted to developing tools for coding and managing OWL ontologies [**?** **?** ]. Many of the technologies proposed by the W3C already existed before they were positioned under the W3C standards and recommendations. These are used in various contexts, particularly those dealing with information that encompasses a limited and defined domain, and where sharing data is a common necessity, such as scientific research. Prominent examples of such efforts include the Gene Ontology (GO [**?** ]), Unified Medical Language System (UMLS [**?** ]), and Neural ElectroMagnetic Ontologies (NEMO [**?** **?** ]).

The OWL ontology language is built on the W3C's Resource Description Framework (RDF) that provides a simple model to describe information resources as a graph. The core of the framework is the RDF statements consisting of triples including a subject (the resource being described), a predicate (the property) and an object (the property value). This simple model of assertions leads to a network of information resources, interrelated by properties which establish relations, or links, between resources and property values. The term RDF Graph is defined as a set of RDF triples (which can be illustrated by a node-arc-node link). Therefore, any collection of RDF data is an RDF Graph [**?** ].

At the same time, there has been a surge of interest in tackling the problem of mining semantically rich datasets, where objects are linked in a number of ways. In fact, many datasets of interest today are best described as a linked collection, or a graph, of interrelated objects [**?** ]. These graphs may represent homogeneous networks, in which there is a single-object type and link type (such as web pages connected by hyperlinks), or richer, heterogeneous networks, in which there may be multiple object and link types (such as DBpedia, a data source containing structuralized information from Wikipedia). Many traditional information resource and format can be viewed as graphs or linked collections as well. Such links and their characteristics are explored, often implicitly, by those well established data analysis and mining techniques, whose formalism of the problem are however typically not based on graphs. For example, consider a simple transaction table and the problem of frequent itemset generation. An itemset is deemed frequent if its *support*, i.e., the percentage of transactions which contain that itemset, is above a threshold. If we characterize the "co-occurrence" relationship (items appearing together in a tuple) as a link between items, then the transaction table can be viewed as a graph consisting a set of items connected by such links. Furthermore, in this sense, the

problem of frequent pattern mining can be reformulated as to identify sets of nodes in the graph that are heavily connected by the co-occurrence link.

It can be seen that, the interface between domain knowledge and data mining can be possibly achieved by using graph representations in which distinct sorts of knowledge that has been traditionally differently represented can be structured in a unified manner. For example, previously, one important aspect of the distinction between domain knowledge and data is the different representations for *ontological* and *factual* knowledge. Ontological knowledge is related to general categories, also called concepts or classes (such as those defined in OWL ontologies). Factual knowledge makes assertions about a specific situation (*e.g.*, "this specific entity belongs to a certain category. and has a certain relationship with another entity, such as those defined in knowledge bases or relational databases). However, this distinction can be obscured by the simple semantics of the RDF model given the fact that RDF allows a combined specification of both schema and data structured under this schema. Since RDF's abstract triple syntax has a graph nature, and graph is one of the most fundamental objects studied in mathematics that has a strong relation with logical languages and data structures, developing graph-based approach becomes promising for providing a common ground to interface with both domain knowledge and data mining. Therefore, in this dissertation, we explore a graph-based approach for modeling both knowledge and data, and for analyzing the combined information source from which insight can be derived systematically. We expect this novel paradigm to contribute to the development of new principles towards semantic data mining.

Since graph-based semantic data mining is a new field, many interesting research directions related to it are yet to be explored. This dissertation studies two such research directions, which have the potential to make significant contributions towards improving

the data mining systems carried out by systematically incorporating domain knowledge. Brief description on these two directions are presented below.

1. **A graph interface for both knowledge representation and data mining.**

   The interface between knowledge representation and data mining is achieved by employing the RDF model and by the fact that RDF allows a combined specification of both schema and data structured under this schema.

   RDF's ability to represent disparate, abstract concepts has led to its increasing use in knowledge representation. To express schematic information, RDF by itself provides only the most basic predefined concepts, the so-called RDF core vocabulary and RDF Schema. RDF Schema permits elementary structuring of a vocabulary, which enables a type-focused understanding of a vocabulary for machine agents [**?** ]. However, RDF Schema's expressiveness is limited to basic assertions about class and property relationships. "Richer" schema and ontology defining languages (e.g., OWL) that are built upon RDF continue to evolve.

   On the other hand, in practice, vast amount of data is often persisted in relational databases (RDB). Mapping from RDB to RDF has gained increasing attention and led to implementation of generic mapping tools as well as domain–specific applications. The W3C launched the RDB2RDF incubator group to explore issues involved in mapping RDB to RDF. A work-in-progress survey paper has been published documenting approaches in this field [**?** ].

   An outstanding advantage of expressing data in RDF is the explicit modeling of relationships between entities that are either implicit or non–existent in RDB. And in such way, one can achieve the incorporation of "domain semantics", an important aspect to fully leverage the expressivity of RDF models that enables data

mining systems to explore beyond pure data. Furthermore, the role of RDF as an integration platform for data from multiple sources is another main motivations for mapping RDB to RDF.

RDF's abstract triple syntax has a graph nature. Graphs are mathematical objects that enjoy wide-spread usage for many tasks, which include the visualization and analysis of data for humans, mathematical reasoning, and the implementation as a data structure for developing data mining algorithms. Besides the common graph-theoretic model of RDF as labeled, directed multi-graphs, Hayes has established that RDF can be also represented as hypergraphs (bipartite graphs) [**?** ]. This result constitutes an important aspect of the theoretical basis of this dissertation and is discussed in the Background Chapter. We have also developed various ways to represent data for different data mining tasks, including association rule mining, classification, clustering and so forth, details of which shall be described in Chapter III and IV.

2. **A graph theoretic analysis approach for mining the combined information source of both data and knowledge.**

   Using graph-based formalisms can bring new perspectives to traditional data mining tasks. For example, as mentioned before, the problem of frequent pattern mining can be reformulated as to identify sets of nodes in the graph that are heavily connected by the co-occurrence link. It is evident that the traditional measure of support is essentially restrained to only take into account direct links between items while ignoring the indirect ones (those through intermediaries). A prominent example of such indirect association was given by Swanson's land mark paper published in 1987 [**?** ] before the age of Web-scale computation: Let A stand for dietary fish oil; let B stand for a set of certain changes in the blood and circulatory system; and

let C stand for the amelioration of Raynaud's syndrome, a peripheral circulatory disorder. Swanson was the first to hypothesize that dietary fish oil might benefit people afflicted with Raynaud's syndrome (A is associated with C) given that it had been already widely known that dietary fish oil causes certain blood changes (A is associated with B) and that these same blood changes may ameliorate Raynaud's syndrome (B is associated with C). The latter two associations were supported by substantial bodies of evidence but were mutually isolated and apparently had never been brought together in print. Without discovering that indirect association (A and C) a potentially useful hypothesis that is worth testing would have been missed. A graph-based analysis approach is particularly suitable to effectively discover this kind of pattern.

In the context of a graph model for RDF, it is useful to introduce some kind of distance metric between information resources to approach the intuitive notion of relatedness between resources. The relation between resources is founded on the paths which connect them. Simply taking the path length as a distance metric would be a naive approach. More sophisticated metric may consider spectral information of the graph or employ weighting schemes to account for meaning of the connecting properties (edges). With a reasonable graph model as well as a meaningful distance metric defined, a lot of traditional mining problems, including frequent pattern mining, clustering and classification, can be tackled in a new angle with the ability to tap on domain knowledge expressed in the same graph model.

At the same time, We note that these problems may be more challenging in the graph domain which is due to the cost of the greater expressive power associated with graphs. In addition, different mining tasks and applications require different kinds of graphs and face varied corresponding challenges. For example, some data

graphs are relatively small but involve a large number of semantic types resulting in highly heterogeneous connection. On the other hand, in many large scale domains such as the web and social networks, the node labels (*e.g.*. URLs) are distinct, but the number of nodes is overwhelming. Such graphs are also challenging because the degree distributions of these graphs are highly skewed [**?** ]. Drawing inspiration from a broad body of literature on graph mining and graph spectral analysis, we aim at developing effective and efficient graph theoretic analysis methods for mining the unified graph representation modeling information in both domain knowledge and data.

The remainder of this dissertation is organized as follows. In Chapter II, we discuss the background areas related to the original research work presented. The main contributions of this dissertation are presented in Chapters III, IV, V and VI. Chapters III discusses the theory of hypergraph-based representation of both knowledge and data. Chapters IV describes hypergraph-based analysis method for traditional data mining tasks including association rule mining, classification and clustering. Chapter V is dedicated to the theory of mining on a combined information source of both knowledge and data graph. Chapter VI demonstrates implementation and case studies. Finally, in Chapter VII, we discuss future directions for the research work and provide some concluding remarks.

# CHAPTER II

# BACKGROUND

This chapter covers the background areas and related work necessary to understand the contributions of this dissertation. It discusses the current state of the effort to incorporate domain knowledge in data mining. In addition, it describes the various formalisms proposed in the literature to use graphs in tackling data mining and knowledge representation problems. We conclude with a high-level summary of these background areas.

## 2.1. Domain Knowledge in Data Mining

Domain knowledge relates to information about a specific domain or data that is collected from previous systems or documentation, or elicited from domain experts. In the rest of the section, we highlight a body of studies that aims at exploring ways to employ domain knowledge in data mining. The results from these studies strongly attest to the positive influence of domain knowledge. Domain knowledge can affect the discovery process within a data mining system in at least two ways. First, it can make patterns more visible by generalizing the attribute values, and second, it can constrain the search space as well as the rule space.

In order to effectively summarize and compare different previously proposed systems, we propose a reference framework to classify different kinds of domain knowledge at a very high abstraction level as detailed in the following.

– Knowledge about the domain: This category contains information related to a specific domain of discourse, usually obtained from either domain experts or previous

data mining processes. Examples of such knowledge include concept hierarchy, integrity constraints, *etc.*

- Knowledge about the data: This category contains information about dataset, including how it is generated, transformed and evolved. This knowledge is obtained from data producers (people who carry out experiments or collect data) or database managers. *e.g.*, in a database of spatial information, one of the images may have been recorded with a very skew angle on the object. When processing the database the discovery process must take this information into account.

- Knowledge about the data mining process: This category contains information pertaining to specific data mining sessions, including goals, parameters and variables related to the experiment. *e.g.*, attributes of interest within data, roles of attributes as being whether antecedents or consequences in association rule mining, and the measure of interestingness for discovered patterns.

The summarized work can be divided into two groups. The first group does not explicitly leverage any knowledge representation approaches to model domain knowledge. The second group explores mainly ontological knowledge (concept hierarchy) and uses formal ontology languages to encode such knowledge. The kind of domain knowledge involved in the first group is broader which covers all categories discussed in the above reference classification scheme. However, it is achieved at the cost of less formality which often result in ad-hoc expression of domain knowledge that has a very application-specific form, scop and granularity.

In one of the earliest studies on the subject, Pazzani and Kibler [**?** ] developed a general purpose relational learning algorithm called FOCL, which combines explanation-based and inductive learning. In a later study, they conducted an experiment comparing

FOCL with a domain theory to FOCL without a domain theory. A partial knowledge base of an expert system was used as the domain theory. They found that incorporating domain theory significantly reduced misclassification costs when larger training sets were used.

In another study, Ambrosino and Buchanan [**?** ] examined the effects of adding domain knowledge to a rule induction program for predicting the risk of mortality in patients with community–acquired pneumonia. They developed a graphical data exploration tool for domain experts to encode domain knowledge and interact with the data mining process. The domain experts participated in two stages of mining. They were first asked to modify the existing set of attributes according to their domain knowledge, and then they were prompted with mining results and were able to modify the mined models (rules) directly. The experiment contained an experimental where the domain knowledge was incorporated as mentioned above, and a control group without domain knowledge. The experimental group performed significantly better (lower percent mean error) than control group.

Sinha and Zhao [**?** ] conducted an extensive comparative study on the performance of seven data mining classification methods—naive Bayes, logistic regression, decision tree, decision table, neural network k-nearest neighbor, and support vector machine—with and without incorporating domain knowledge. The application they focused on was in the domain of indirect bank lending. An expert system capturing a lending expert's knowledge of rating a borrower's credit is used in combination with data mining to study if the incorporation of domain knowledge improves classification performance. In their study, the domain knowledge used was partial, meaning that it could only lead to intermediate results but was not sufficient to make the final prediction. They cascaded the expert system with a data mining classifier. The experiment adopted an experimental vs.

control paradigm, similar to Ambrosino et al.'s early experiment in 1999. The prediction proposed by the expert system was added to other inputs. Classifiers built using input data enhanced by the expert system's output formed the experimental group. For the control group, classifiers were built using the original set of input attributes (bypassing the expert system). Their results showed that incorporation of domain knowledge significantly improves classification results with respect to both misclassification cost and AUC (Area Under Curve). Hence they concluded by calling for more attention in combining domain knowledge and data mining. They articulated that, in knowledge engineering, the focus is on the knowledge of a human expert in a specific problem area. On the other hand, the focus of data mining is on the data available in an organization. Expert systems and data mining methods could play complementary roles in situations where both knowledge and data are available.

Hirsh and Noordewier [**?** ] argued that if learning is to be successful, the training data must be encoded in a form that lets the learner recognize underlying regularities. The application domain they focused on was the problem DNA sequence classification. They proposed to use background knowledge of molecular biology to re-express data in terms of higher-level features that molecular biologists use when discussing nucleic-acid sequences. The high lever features were Boolean valued, representing the presence or absence of the feature in a given DNA sequence. Using C4.5 decision trees and backprop neural networks, they conducted experiments with and without the higher-level features. For both learning methods, the use of higher-level features resulted in significantly lower error rates.

Pohle [**?** ] contended that data mining techniques are good at generating useful statistics and finding patterns in large volumes of data, but "they are not very smart in interpreting these results, which is crucial for turning them into interesting,

understandable and actionable knowledge". The author viewed the lack of sophisticated tool to support incorporating human domain knowledge into the mining process as being the main factor responsible for the limitation. They also pointed out that ontologies were valuable technologies to incorporate domain knowledge and thus they propose to exploit ontologies when integrating knowledge mined from knowledge discovery process to an existing knowledge base.

Kopanas *et al.* [**?** ] conducted large scale data mining experiment exploring the role of domain knowledge in different phases of a large-scale data mining project, using a case study of customer insolvency in the telecommunications industry. They argued against the claim that data mining approaches eventually will automate the process and lead to discovery of knowledge from data with little or no support of domain experts and domain knowledge. For each stage in data mining they identified types of domain knowledge involved to either improve the performance or, in some case, make data mining process possible at all. They found that though domain knowledge plays a critical role mainly in the initial and final phases of the project, it influences the other phases to some degree as well. For example, in problem definition stage, domain knowledge involves business and domain requirements and other implicit, tacit knowledge. In data preparation stage, the useful domain knowledge involves semantics of corporate database. In data preprocessing and transformation stage, domain knowledge includes tacit and implicit knowledge for inferences. In feature and algorithm selection stage, main type of knowledge involves how to interpret selected features. In mining stage, domain knowledge focuses on inspection of discovered knowledge. In evaluation stage, domain knowledge defines performance criteria related to business objectives. In fielding knowledge base stage (incorporating mined knowledge with existing knowledge base), domain knowledge provides supplementary information for implementing the fusion.

In another study, Weiss *et al.* [**?** ] combined an expert system with a data mining method for generating better sales leads. They developed an expert system that interviews executives of small and medium-sized companies and, based on their responses, recommends promising sales leads. The question-answer pairs and the recommended solutions were stored as examples to be mined by the method of rule induction. The study demonstrated how a knowledge base can be used to guide a machine learning program. The techniques developed in the study would be useful for consultation systems whose questions have different costs of acquisition.

Daniels *et al.* [**?** ] demonstrated that data mining systems can be successfully combined with explicit domain knowledge. They pointed out that in theory there are two extreme situations that may occur with respect to the availability of domain knowledge. The first is that no prior knowledge whatsoever is available. The second is all relationship is known with certainty, up to a limited number of parameters. They then claimed that their study was positioned somewhere between these extremes. The authors focused on a special type of a priori knowledge, monotonicity, i.e., the sign of relationship between the dependent and independent variables, for economic decision problems. Prior knowledge was implemented as monotonicity constraints in decision tree and neural network classifiers. Addition of the knowledge resulted in smaller decision trees, and smaller variations of error R2 on the training and test sets for neural networks. The authors also claimed that the framework developed might serve as a tool to implement normative requirements. However, since monotonicity constraints were incorporated in the decision tree and neural networks by designing specific algorithms, it is not obvious how to generalize the algorithm design process to include other normative domain knowledge.

Yoon *et al.* [**?** ] studied semantic query optimization, a field that endeavors to optimize data mining queries by taking advantage of domain knowledge. The authors

demonstrated that significant cost reduction can be achieved by reformulating a query into a less expensive yet equivalent query that produces the same answer as the original one. They identified that in most cases, exhaustive analysis of data is infeasible. It is often necessary to perform a relatively constrained search on a specific subset of data for desired knowledge. The domain knowledge they utilized was classified into three categories, interfiled, category, and correlation, all of which can be represented in rule forms. When a data mining query is received, they first identify domain knowledge that is relevant to the query, and transform it accordingly. On the other hand, to select relevant domain knowledge without an exhaustive search of all domain knowledge, they developed a method that built tables for domain knowledge indexed by attributes.

Vikram and Nagpal [**?** ] developed an iterative association rule mining algorithm to integrate user's domain knowledge with association rule mining. The knowledge they request from the users is attributes of interest. According to users' specification, database is scrutinized to produce a working subset that only contains the attributes of interest while the rest are excluded. With this dataset, the Apriori procedure searches for frequent large itemsets. The advantage is apparent since irrelevant records are filtered out, the result is more meaningful and the running time is also reduced.

We summarize the above surveyed research systems in Table 2.2.. Each system is characterized by 1) its domain of application, 2) type of domain knowledge employed, 3) usage of domain knowledge, and 4) data mining techniques that are adapted to incorporate the domain knowledge.

| System | Problem domain | Type of domain knowledge | Usage of domain knowledge | Data mining method |
|---|---|---|---|---|
| Daniels *et al.* [? ] | Business Intelligence | Monotonicity constraints | modify mining algorithms to embody the knowledge directly | Decision tree and neural network |
| Ambrosino *et al.* [? ] | Medical decision | Attribute-relation, interpretation of result | Experts interact directly with mining in both pre– and post– processing stages | Decision tree |
| Pazzani *et al.* [? ] | Predicate learning | Taxonomy, attribute-relation rules, attribute correlations | Preprocessing data | FOCL |
| Sinha *et al.* [? ] | Business Intelligence | Expert rules | Rule's prediction cascaded as an input to classifier | Seven typical classification algorithms |
| Yoon *et al.* [? ] | Query optimization | Taxonomy, attribute relation rules and correlation | Transform data mining queries | Not specified |
| Hirsh *et al.* [? ] | DNA sequence classification | Attribute relation rules | Forming new set of attributes | C4.5 and neural network |
| Vikram *et al.* [? ] | Association rule mining | Attribute of interest | Preprocessing data | Association rules |
| Weiss *et al.* [? ] | Consultation | Question-answer pairs derived from interviewing experts | Question-answer pairs serve as part of the input to a mining system | No restriction |
| Kopanas *et al.* [? ] | Business intelligence | Comprehensive information pertaining to a domain | For each stage of mining, discussing the use of certain type of domain knowledge in general | No restriction |

TABLE 2.1. Summary of systems that employ domain knowledge.

Staab and Hotho [**?** ] describe an ontology-based text clustering approach. They develop a preprocessing method, called COSA, one of the earliest to utilize the idea of mapping terms in the text to concept in the ontology. The authors point out that the size of the high-dimensional term vector representation of the text document is the principal problem faced by previous algorithms. By mapping terms to concepts, it essentially aggregates terms and reduces the dimensionality.

The mapping of terms to concepts can be also seen as a process of semantic annotation. It is realized in COSA by using some shallow and efficient natural language processing tools. After the mapping process, COSA further reduces the dimensionality by aggregating concepts using the concept heterarchy defined in the "core ontology" used in their framework. The idea is navigating the heterarchy top-down splitting the concepts with most support (number of mapping terms) into their sub-concepts and abandoning the concepts with least support. The rationale is that too (in-) frequent concept occurrences are not appropriate for clustering. Note that the definition of a "core ontology" in this paper was developed prior to the emergence of OWL. The concept heterarchy should be thought of as equivalent to the subsumption hierarchy (taxonomy) in OWL. Despite the out-dated definition of terminology, COSA pioneers in incorporating ontology in text clustering and displays some generality over the confines of any specific domain.

Wen *et al.* [**?** ] devise a framework that solves the genomic information retrieval problem by using ontology-based text clustering. The core idea is an extension to COSA. Documents containing genomic information are first annotated based on UMLS so that the terms are mapped to concepts. Then the authors point out that even the dimension of clustering space is dramatically reduced, there still exists the problem that a document is often full of class-independent "general" words and short of class-specific "core" words, which leads to the difficulty of document clustering because class-independent words are

considered as noise in clustering. To solve this problem, the authors propose a technique for concept frequency re-weighing which takes into consideration the concept subsumption hierarchy defined in the domain ontology. Finally, from the re-weighed concept vector representation, a cluster language model can be generated for information retrieval.

Fang *et al.* [**?** ] propose an ontology-based web documents classification and ranking method. The contribution of this work is that they introduce a way to automatically augment or tailor the existing ontology to fit the specific purpose, while in previous work one has to either manually create an ontology from scratch or adopt some well established domain ontology. Their technique is to enrich a certain ontology using terms observed in the text document. This is done with the help of WordNet. Specifically, for example, if the sense of a term appears to be a synonym of the sense of a concept according to WordNet, the term will be added to the ontology as a sibling of the concept. The enriched ontology is then treated as a representation of the category to which some text document is classified. The proposed classification is done by simply comparing the similarity between ontologies and the term vector representing the text document. This implies that first, multiple ontologies should be provided for choice, and second, for each category of the corpus there should be one corresponding ontology. These assumptions appear cumbersome though the authors point to the Open Directory Project as a source for initial ontologies in their experiment. Moreover, this process does not fit into traditional classification as there is no training phase. It is more similar to clustering with known number of clusters.

Cheng *et al.* [**?** ] studied document clustering problem as a means to efficient knowledge management. They utilized ontologies to overcome the ambiguity problem frequently seen in natural language since "an ontology includes a selection of specific sets of vocabulary for domain knowledge model construction, and the context of each

vocabulary is represented and constrained by the ontology." They developed a system called Ontology-based Semantic Classification (OSC) Framework that consists of two main components: Content-based Free Text Interpreter (CFTI) and Context-based Categorization Agent (CCA). CFTI leverages on the Link Grammar capability for syntactical analysis of a sentence. At the same time, the lexical meaning analysis of a sentence is supported through the integration with ontological models such as the WordNet. The context models produced from CFTI correlate the content of a particular document with the context of the user. The role of the CCA is to further enhance the usability of these context models by classifying them according to the user interest. The OSC framework seems appealing but the authors did not provide any implementation details nor experiment results. It was more of a research proposal and it would be interesting to see the performance of the system when the authors make the proposal a reality.

Taghva *et al.* [**?** ] reported on the construction of an ontology that applies rules for identification of features to be used for an email classification system, called "Ecdysis". The ontology is designed for the purpose of encoding expert rules deciding the email category. Therefore it contains only those concepts that are aspects of such rules. CLIPS is used to implement rules and the inference with rules is based on a "match-and-fire" mechanism: One or more features of an email instance would be matched with instances of classes from the ontology. If there was a successful match, then the rule would fire, causing the email to have some certain feature. This feature becomes one of many that can now be used for training and classification with our Bayesian classifier Ecdysis. The authors claim that preliminary tests show that these additional features enhance the accuracy of the classification system dramatically.

Tenenboim *et al.* [**?** ] proposed an automatic method for classification of news using hierarchical news ontology. The system they developed was called ePaper. It is designed to aggregate news items from various news providers and delivers to each subscribed user a personalized electronic newspaper, utilizing content-based and collaborative filtering methods. The ePaper can also provide users "standard" (*i.e.*, not personalized) editions of selected newspapers, as well as browsing capabilities in the repository of news items. The classification task performed in the ePaper system aims at classifying each incoming news document to one or several concepts in the news Ontology. In this sense, only the target classes in the classification process are annotated by ontological terms. Since the users' profiles are also defined using the same set of ontological terms, a content-based filter is able to compare the similarity between a user's profile and classified categories of news. Based on results of the classifier and content-based filter, the personalization engine of the system is able to provide a personalized paper.

Lula *et al.* [**?** ] proposed an ontology-based cluster analysis framework. They discussed various aspects of similarity measure between objects and sets objects in ontology-based environment. They devised an ontology-based aggregation function to calculate similarity between two objects which takes into account taxonomy similarity, relationship similarity and attribute similarity. For example, path distance, Jaccard coefficient and measures based on information theory can be used to calculate taxonomy similarity. Relationship similarity can be determined by calculating similarity of objects that participate in the relationship. Attribute similarity is determined by comparing values of the attributes. The authors claim that the framework with ontology-based similarity measure opens the possibility for various clustering application. But apparently much work still remains. It is unclear how the aggregation function is defined though each of its components can be solved separately. A proper aggregation is highly possible to be

application-specific, which may suggest the need of a learning framework to derive such function.

Li *et al.* [**?** ] developed a new decentralized P2P architecture-ontology-based community overlays. The system exploits the semantic property of the content in the network to cluster nodes sharing similar interest together to improve the query and searching performance. To do that, they proposed a query routing approach that organizes nodes into community overlays according to different categories defined in nodes' content ontology. A community overlay is composed of nodes with similar interest. Queries are only forwarded to semantically related overlays, thus alleviating the traffic load. According to taxonomic information in the ontology, peers (nodes) can be clustered into ontological terms. The authors further discussed routing policy among communities and issues related to community overlay maintenance, which is out of scope of this paper. This study introduced a new data mining application besides text document clustering. But their principle remained the same as other surveyed work: ontology is used as an abstraction to data. By doing so, some performance metrics of the data mining task can be improved.

Adryan *et al.* [**?** ] developed a system called GO-Cluster which uses the tree structure of the Gene Ontology database as a framework for numerical clustering, and thus allowing a simple visualization of gene expression data at various levels of the ontology tree. Shen *et al.* [**?** ] proposed a new method of association rules retrieval that is based on ontology and Semantic Web. They argue that ontology-based association rules retrieval method can well deal with the problems of rule semantics sharing, rule semantics consistency and intelligibility.

| System | Ontology construction | Annotation method | Type of sources | Data mining method |
|---|---|---|---|---|
| Staab *et al.* (COSA) [**?** ] | Manual creation | Shallow NLP method | Text | Clustering based on "bag-of-concept" representation plus concept aggregation |
| Wen *et al.* [**?** ] | Off-the-shelf (UMLS) | Manual | Text | Clustering based on "bag-of-concept" representation plus concept frequency reweighing |
| Fang *et al.* [**?** ] | Manual creation of "core" ontology and update on the fly | Manual | Text | Clustering based on "bag-of-concept" representation plus feed back to enrich ontology |
| Cheng *et al.* (OSC) [**?** ] | Off-the-shelf (WordNet) | Rule-based NLP | Text | Not specified |
| Taghva *et al.* (Ecdysis) [**?** ] | Manually creation, incorporated with a rule inference system | Manual | Email / text | Classification with additional features derived from rules |
| Tenenboim *et al.* [**?** ] | Manual creation | Manual | News archive /text | Not specified |
| Lula *et al.* [**?** ] | Not specified | Manual | Text | Hierarchical agglomerative clustering |
| Li *et al.* [**?** ] | Off-the-shelf (Open Directory Project) | Manual | P2P user / resource profile data | Not specified |
| Adryan *et al.* [**?** ] | Off-the-shelf (Gene Ontology) | Manual | Gene expressions | Hierarchical clustering with instance regrouping based on GO annotation |

TABLE 2.2. Summary of ontology-based data mining systems.

## 2.2. Graph-based Approach for Knowledge Representation

Graph-based approaches for representing knowledge have long been used in philosophy, psychology, and linguistics. The computer counterpart to this means is the so-called *semantic network* that represents knowledge in patterns of interconnected nodes and arcs which were first developed for artificial intelligence and machine translation.

The semantic network, and graph-based approaches for KR in general, are motivated by the desirable qualities of graph for both modeling and computation. From a modeling viewpoint, basic graphs are easily understandable by users, and it is always possible to split up a large graph into smaller ones while keeping its semantics. From the computational viewpoint, graph is one of the most studied objects in mathematics. Considering graphs instead of logical formulas provides another view of knowledge constructs (*e.g.*, some notions like path, cycle, or connected components are natural on graphs) and provides insights to algorithmic ideas [**?** ]. In light of these motivations, what is common to all semantic networks is a declarative graphic representation that can be used either to represent knowledge or to support automated systems for reasoning about knowledge.

According to Sowa [**?** ], following are six of the most common kinds of semantic networks.

1. Definitional networks focus on the is-a or subtype relation among concepts. The resulting network, also called a generalization or subsumption hierarchy, supports the rule of inheritance to propagate properties from a supertype to all of its subtypes. The information in these networks is often assumed to be necessarily true.

2. Assertional networks are designed to assert propositions. Unlike definitional networks, the information in an assertional network is assumed to be contingently

true, unless it is explicitly marked with a modal operator. Some assertional networks have been proposed as models of the conceptual structures underlying natural language semantics.

3. Implicational networks use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs, causality, or inferences.

4. Executable networks include some mechanism, such as marker passing or attached procedures, which can perform inferences, pass messages, or search for patterns and associations.

5. Learning networks build or extend their representations by acquiring knowledge from examples. The new knowledge may change the old network by adding and deleting nodes and arcs or by modifying numerical values, called weights, associated with the nodes and arcs.

6. Hybrid networks combine two or more of the previous techniques, either in a single network or in separate, but closely interacting networks.

Among all variants of semantic networks, we emphasize the most on the usage of definitional networks to incorporate domain knowledge in data mining. The kind of knowledge that are best captured by this kind of network is subsumption hierarchy, on which a distance (similarity) measure can be reasonably defined. Such measure is essential in many data mining tasks. It is possible to extend, in a straightforward manner, data mining algorithms that depend on analyzing distances between entities in factual knowledge to work with distances between those in ontological knowledge.

In addition, one of the most prominent KR formalism families among current systems of definitional networks, description logics (DLs), formerly called terminological

logics or concept languages, have been a successful attempt to combine well-defined logical semantics with efficient reasoning [**?** ]. They are derived from an approach proposed by Woods [**?** ] and implemented by Brachman [**?** ] in a system called Knowledge Language One (KL-ONE). Recent description logics are DAML+OIL [**?** ] and its successor OWL, which are intended for representing knowledge in the semantic web [**?** ]–a giant semantic network that spans the entire Internet.

### 2.2.1. Graph Representation of RDF

According to the W3C specification for RDF semantics [**?** ], an RDF graph, or simply a graph, is defined as a set of RDF triples. A subgraph of an RDF graph is a subset of the triples in the graph. A triple is identified with the singleton set containing it, so that each triple in a graph is considered to be a subgraph. A proper subgraph is a proper subset of the triples in the graph. A ground RDF graph is one with no blank nodes.

RDF triples can be visualized as a *directed labeled graph*,

$$\boxed{\text{subject}} \xrightarrow{\textit{predicate}} \boxed{\text{object}},$$

where subjects and objects are represented as nodes, and predicates as edges. The directed labeled graph model for RDF is straightforward and convenient in most cases. But inconsistency arises when using triples to make assertions on predicates. The directed labeled graph model of RDF makes the artificial distinction between resources and properties. The results of the understanding of RDF bounded by the this model becomes especially evident in the limitations of current RDF query languages as studied in [**?** ].

The following example illustrates this point of view.

In this example, two different levels of information are expressed. At the data level, the following statements describe the relationship between nodes representing people:

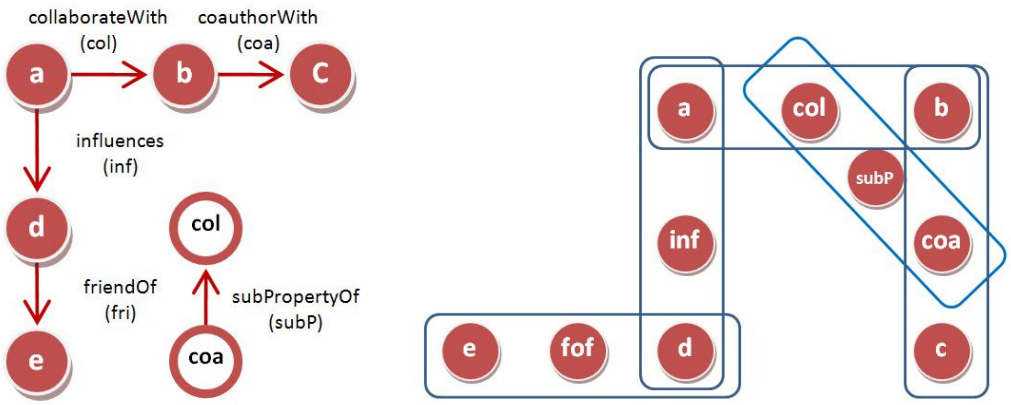



FIGURE 2.1. An example of nodes connected by different links and some relationship between the links, represented by A) a simple graph, and B) a hypergraph.

$\langle a\ col\ b\rangle$, $\langle b\ coa\ c\rangle$, $\langle a\ inf\ d\rangle$ and $\langle d\ fri\ e\rangle$. At the schema level, another statement asserting that *coa* is a sub-property of *col*: $\langle coa\ subP\ col\rangle$. In this case, *col* and *coa* become both nodes and edges in the directed labeled graph, causing the inconsistency (as depicted in Fig. 3.1.(A). One way to overcome this is to model RDF as a hypergraph.

A hypergraph [**?** ] is a generalization of a traditional graph where edges, called hyperedges, can connect more than two vertices. If each edge in a hypergraph covers the same number of nodes, it's called $r$-uniform hypergraph, $r$ being the number of nodes on each edge. Any RDF Graph can be represented by a simple ordered 3-uniform hypergraph, in which an RDF triple corresponds to a hypergraph edge, the nodes being the subject, predicate and object in this order. In this way, both meta-data and data level statements can be integrated in a consistent model. In Fig. 3.1.(b), the information in the above example is represented using a hypergraph.

Formally, a hypergraph $G = (V, E)$, is a pair in which $V$ is the vertex set and $E$ is the hyperedge set where each $e \in E$ is a subset of $V$. A weighted hypergraph is a hypergraph that has a positive number $w(e)$ associated with each hyperedge $e$; called the

weight of hyperedge $e$: Denote a weighted hypergraph by $G = (V, E, w)$. The degree of a vertex $v \in V$, $d(v)$, is defined as $d(v) = \sum_{v \in V, e \in E} w(e)$. The degree of a hyperedge $e$, denoted as $\delta(e)$, is the number of vertices in $e$, i.e. $\delta(e) = |e|$. A hyperedge $e$ is said to be incident with a vertex $v$ when $v \in e$. The hypergraph incidence matrix $\mathbf{H} \in \mathbb{R}^{|V| \times |E|}$ is defined as

$$h(v, e) = \begin{cases} 1, & v \in e \\ 0, & otherwise \end{cases}$$

Throughout the rest of the paper, the diagonal matrix forms for $\delta(e)$, $w(e)$, $d(v)$ are denoted as $\mathbf{D}_e$, $\mathbf{W} \in \mathbb{R}^{|E|}$, and $\mathbf{D}_v \in \mathbb{Z}^{|V|}$, respectively.

## 2.3. Graphs in Data Mining

### 2.3.1. Graph Representation of Relational Structure

An object set endowed with pairwise relationships can be naturally illustrated as a graph in which vertices represent objects, and any two vertices that have some kind of relationship are joined together by an edge. In the case of frequent itemset mining, a set of objects with the co-occurrence relationship can be represented as directed or undirected graphs. For illustrating this point of view, let us consider a relational table depicted in Figure 2.2.(a). One can construct an undirected graph where the set of vertices is the set of relational attributes (column items) and an edge joins two vertices if the they co-occur in a tuple (as illustrated in Figure 2.2.(b)). This graph is called *Gaifman graph* [**?** ] of a relational structure. The undirected graph can be further enriched by assigning to each edge a weight equal to the support of the 2-itemset consisting of vertices incident to the edge. Cliques (complete subgraphs) in the Gaifman graph, or *Gaifman cliques* for short, are of particular interest because every tuple (ground atom) in data corresponds to a Gaifman clique. However, ambiguity arises as not all Gaifman cliques have matching

tuple in the data. There exists cases where cliques are incidental in the sense that several relational ground atoms play together to induce a clique configuration in the Gaifman graph, but no ground atom covers the entire clique (e.g., the clique of $\{A, B, C, D\}$ in Figure 2.2.(b) does not correspond to any tuple in the relational table).

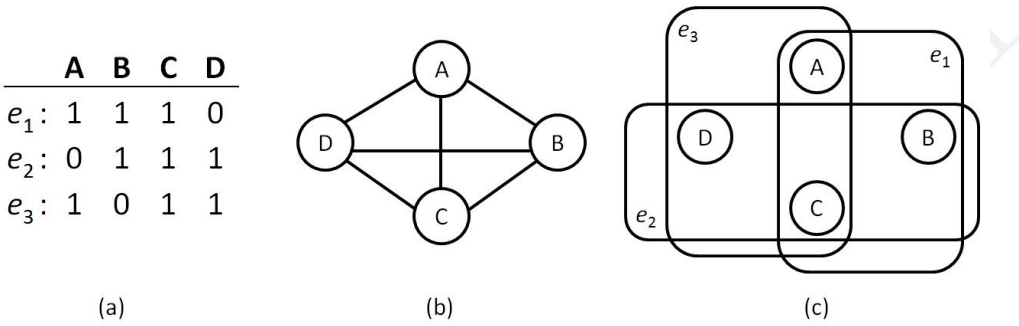



FIGURE 2.2. (a) an example transaction table; (b) the Gaifman graph representation of the table; (c) The hypergraph representation of the table

A natural way to remedy the ambiguity is to represent the relational data as a *hypergraph* [**?** ]. A hypergraph is a generalization of traditional graph. An edge in the hypergraph, called hyperedge, can connect more than two vertices. In other words, every hyperedge is an arbitrary nonempty subset of vertices. It is obvious that a simple graph is a special kind of hypergraph with each hyperedge containing only two vertices. In this paper, we propose to employ hypergraphs to model relational structure for finding semantically associated itemsets. Specifically, we propose to construct a hyperedge for each tuple. The relational attributes constitute the universe of vertices in the hypergraph. In this representation, each hyperedge has an exact one-to-one correspondent tuple (see Figure 2.2.(c), for example).

### 2.3.2. Similarity Measure

Given graph-based representation of information sources, meaningful similarity measure $s$ between nodes in the graph is critical in numerous data mining tasks. Take the simple network in Fig. 3.1.(B) for example, suppose given a task of friend recommendation based on the information in this graph, the interesting question is whether $c$ or $e$ is a better choice of recommendation to $a$. To answer this question, it is natural to compare the similarity measures $s(a, c)$ and $s(a, e)$. In a rough sense, on can identify in the hypergraph representation that there are two paths between $a$ and $c$ (the formal definition for paths in hypergraphs will be given in Section 2.3.2.1.), while only one between $a$ and $e$. It's intuitive to conclude that $a$ and $c$ are more similar, or closer, than $a$ and $e$. This gives us a hint that meaningful similarity measures on the graph should satisfy the following intuitions:

1. The more paths connecting two nodes, the closer they are.

2. The shorter the paths, the closer they are.

In other words, the more "short" connections between two given nodes, the more similar those nodes are. To this end, we propose to employ the following quantities as the candidate similarity measure since both of them have the desired property. They are, namely, the *commute time distance* based similarity measure from the random walk model on hypergraph, and the inner product similarity based on the *pseudoinverse of the hypergraph Laplacian*. They are all based on the random walk model on hypergraph. In the following, we briefly introduce the theory of random walk.

#### 2.3.2.1. Random Walk

**Random Walk on Simple Graph** Given a graph and a starting point we select a neighbor of it at random and move to this neighbor then we select a neighbor of this point at random and move to it etc. The random sequence of points selected this way is a random walk on the graph. In other words, a random walker can jump from vertex to vertex and each vertex therefore represents a state of the Markov chain. The average first-passage time $m(k|i)$ [**?** ] is the average number of steps needed by a random walker for reaching state $k$ for the first time, when starting from state $i$. The symmetrized quantity $n(i,j) = m(j|i) + m(i|j)$ called the average commute time [**?** ], provides a distance measure between any pair of states. The fact that this quantity is indeed a distance on a graph was proved independently by Klein and Randic [**?** ] and Gobel and Jagers [**?** ].

The Laplacian matrix $\mathbf{L}$ of a graph is widely used for finding many properties of the graphs in spectral graph theory. Given node degree matrix $\mathbf{D}$ and graph adjacency matrix $\mathbf{A}$, the Laplacian matrix of the graph is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The normalized Laplacian is given by $\mathbf{L}_N = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}$ is the identity matrix. The average commute time $n(i,j)$ can be computed in closed form from the Moore-Penrose pseudoinverse of $\mathbf{L}$ [**?** ], denoted by $\mathbf{L}^+$.

Various quantities derived from random walk on graph has been used in a number of applications. Fouss et al. [**?** ] compared twelve scoring algorithms based on graph representation of the database to perform collaborative movie recommendation. Pan et al. [**?** ] developed a similarity measure based on random walk steady state probability to discover correlation between multimedia objects containing data of various modalities. Yen et al. [**?** ] introduced a new k-means clustering algorithm utilizing the random walk average commute time distance. Zhou et al. [**?** ] presented a unified framework based on

neighborhood random walk to integrate structural and attribute similarities for graph clustering.

**Random Walk on Hypergraph** We can associate each hypergraph with a natural random walk which has the transition rule as described in [**?** ]. Given the current position $u \in V$; first choose a hyperedge $e$ over all hyperedges incident with $u$ with the probability proportional to $w(e)$; and then choose a vertex $v \in e$ uniformly at random. Obviously, it generalizes the natural random walk defined on simple graphs. Let $\mathbf{P}$ denote the transition probability matrix of this hypergraph random walk. Then each entry of $\mathbf{P}$ is

$$p(u,v) = \sum_{e \in E} w(e) \frac{h(u,e)}{d(u)} \frac{h(v,e)}{\delta(e)} .$$

In matrix notation, $\mathbf{P} = \mathbf{D}_v^{-1}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T$. Zhou et al. [**?** ] define the following normalized hypergraph Laplacian $\mathcal{L}$ based on the random walk model:

$$\mathcal{L} = \mathbf{I} - \mathbf{\Theta}, \text{ where } \mathbf{\Theta} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}. \qquad \text{(Equation 2.1.)}$$

# CHAPTER III

# GRAPH REPRESENTATION

The interface between domain knowledge and data is achieved by employing the RDF model and by the fact that RDF allows a combined specification of both schema and data structured under this schema.

RDF's abstract triple syntax has a graph nature. Graphs are mathematical objects that enjoy wide-spread usage for many tasks, which include the visualization and analysis of data for humans, mathematical reasoning, and the implementation as a data structure for developing data mining algorithms. Besides the common graph-theoretic model of RDF as labeled, directed multi-graphs, Hayes has established that RDF can be also represented as hypergraphs (bipartite graphs) [**?** ]. This result constitutes an important aspect of the theoretical basis of this proposal and is discussed in sections below. We propose to use the graph-based representation for RDF as a combined information source of both domain knowledge and data for different mining tasks, including association rule mining, classification, clustering and so forth.

## 3.1. Graph Representation for Domain Knowledge

Graph-based approaches for representing knowledge have long been used in philosophy, psychology, and linguistics. The computer counterpart to this means is the so-called *semantic network* that represents knowledge in patterns of interconnected nodes and arcs which were first developed for artificial intelligence and machine translation.

The semantic network, and graph-based approaches for KR in general, are motivated by the desirable qualities of graph for both modeling and computation. From a modeling viewpoint, basic graphs are easily understandable by users, and it is always possible

to split up a large graph into smaller ones while keeping its semantics. From the computational viewpoint, graph is one of the most studied objects in mathematics. Considering graphs instead of logical formulas provides another view of knowledge constructs (*e.g.*, some notions like path, cycle, or connected components are natural on graphs) and provides insights to algorithmic ideas [**?** ]. In light of these motivations, what is common to all semantic networks is a declarative graphic representation that can be used either to represent knowledge or to support automated systems for reasoning about knowledge.

Among all variants of semantic networks, we emphasize the most on the usage of definitional networks to incorporate domain knowledge in data mining. The kind of knowledge that are best captured by this kind of network is subsumption hierarchy, on which a distance (similarity) measure can be reasonably defined. Such measure is essential in many data mining tasks. It is possible to extend, in a straightforward manner, data mining algorithms that depend on analyzing distances between entities in factual knowledge to work with distances between those in ontological knowledge.

In addition, one of the most prominent KR formalism families among current systems of definitional networks, description logics (DLs), formerly called terminological logics or concept languages, have been a successful attempt to combine well-defined logical semantics with efficient reasoning [**?** ]. They are derived from an approach proposed by Woods [**?** ] and implemented by Brachman [**?** ] in a system called Knowledge Language One (KL-ONE). Recent description logics are DAML+OIL [**?** ] and its successor OWL, which are intended for representing knowledge in the semantic web [**?** ]–a giant semantic network that spans the entire Internet.

OWL ontologies can be used along with information written in RDF, and OWL ontologies themselves are primarily exchanged as RDF documents. RDF's abstract triple

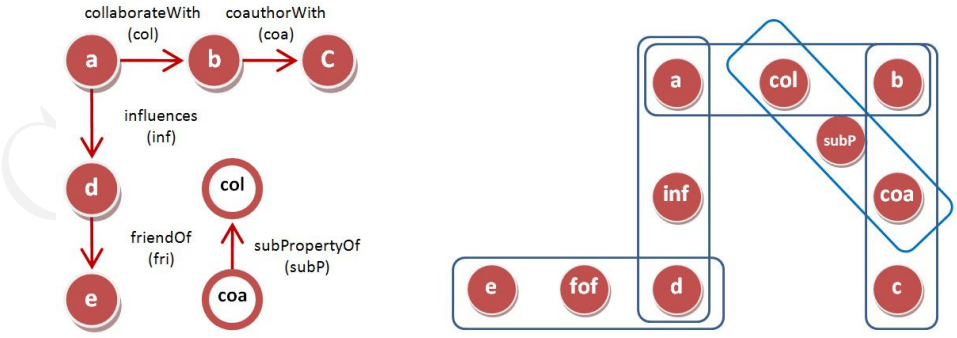
syntax has a graph nature. The RDF graph is defined as follows according to the W3C specification for RDF semantics [**?** ]:

**Definition 3.1..1** (**RDF Graph**)**.** An RDF graph is defined as a set of RDF triples. A subgraph of an RDF graph is a subset of the triples in the graph. A triple is identified with the singleton set containing it, so that each triple in a graph is considered to be a subgraph. A proper subgraph is a proper subset of the triples in the graph. A ground RDF graph is one with no blank nodes.

RDF triples can be visualized as a *directed labeled graph* as follows:

$$\boxed{\text{subject}} \xrightarrow{\textit{predicate}} \boxed{\text{object}},$$

where subjects and objects are represented as nodes, and predicates as edges. The directed labeled graph model for RDF is straightforward and convenient in most cases. But inconsistency arises when using triples to make assertions on predicates. The directed labeled graph model of RDF makes the artificial distinction between resources and properties, which may cause inconsistency in the graph representation. The following example illustrates this point of view.




(A) (B)

FIGURE 3.1. An example of nodes connected by different links and some relationship between the links, represented by A) a simple graph, and B) a hypergraph.

**Example 3.1..1** (**Inconsistent representation of RDF directed labeled model**). In this example, a set of RDF statements is asserted to describe the relationship among a group of people. The information expressed include two different levels, i.e., the meta (ontological) data and factual data. The factual data level consists of following statements: $\langle a\ collaborate\ b\rangle$, $\langle b\ coauthor\ c\rangle$, $\langle a\ influence\ d\rangle$ and $\langle d\ friendOf\ e\rangle$. The meta-data level contains one single statement asserting that *coauthor* is a sub-property of *collaboration*: $\langle coauthor\ subProperty\ collaboration\rangle$. In this case, the representation of *collaboration* and *coauthor* is inconsistent. They are represented as nodes at the factual data level and edges at the meta-data level (see Figure 3.1.(A)).

To overcome this, Hayes et al. [**?** ] proposed to model RDF as a *hypergraph*. A hypergraph [**?** ] is a generalization of a traditional graph where edges, called hyperedges, can connect more than two vertices. If each edge in a hypergraph covers the same number of nodes, it's called $r$-uniform hypergraph, $r$ being the number of nodes on each edge. Any RDF Graph can be represented by a simple ordered 3-uniform hypergraph, in which an RDF triple corresponds to a hypergraph edge, the nodes being the subject, predicate and object in this order. In this way, both meta-data and data level statements can be integrated in a consistent model.

In Fig. 3.1.(B), the information in the Example 3.1..1 is represented by a hypergraph and the inconsistency in the directed labeled graph representation is eliminated.

**Definition 3.1..2** (**Hypergraph**). Formally, a hypergraph $G = (V, E)$, is a pair in which $V$ is the vertex set and $E$ is the hyperedge set where each $e \in E$ is a subset of $V$. A weighted hypergraph is a hypergraph that has a positive number $w(e)$ associated with each hyperedge $e$; called the weight of hyperedge $e$: Denote a weighted hypergraph by $G = (V, E, w)$. The degree of a vertex $v \in V$, $d(v)$, is defined as $d(v) = \sum_{v\in V, e\in E} w(e)$. The degree of a hyperedge $e$, denoted as $\delta(e)$, is the number of vertices in $e$, i.e. $\delta(e) = |e|$. A

hyperedge $e$ is said to be incident with a vertex $v$ when $v \in e$. The hypergraph incidence matrix $\mathbf{H} \in \mathbb{R}^{|V|\times|E|}$ is defined as

$$h(v,e) = \begin{cases} 1, & v \in e \\ 0, & otherwise \end{cases}$$

Throughout the rest of the paper, the diagonal matrix forms for $\delta(e)$, $w(e)$, $d(v)$ are denoted as $\mathbf{D}_e$, $\mathbf{W} \in \mathbb{R}^{|E|}$, and $\mathbf{D}_v \in \mathbb{Z}^{|V|}$, respectively.

A hypergraph $G = (V, E)$ can be transformed to a *bipartite graph* $BG$ as follows:

**Definition 3.1..3** (**Transformation from RDF hypergraph to RDF bipartite graph**)**.** Let the sets $V$ and $E$ are the partitions of $BG$. The node pair $(v_1, e_1)$ is connected with an edge if and only if vertex $v_1$ is contained in edge $e_1$ in $H$. Conversely, any bipartite graph with fixed parts and no unconnected nodes in the second part represents some hypergraph in the manner described above. This bipartite graph can be represented by incidence matrices. Such a matrix can be understood as the node adjacency matrix of a bipartite graph.

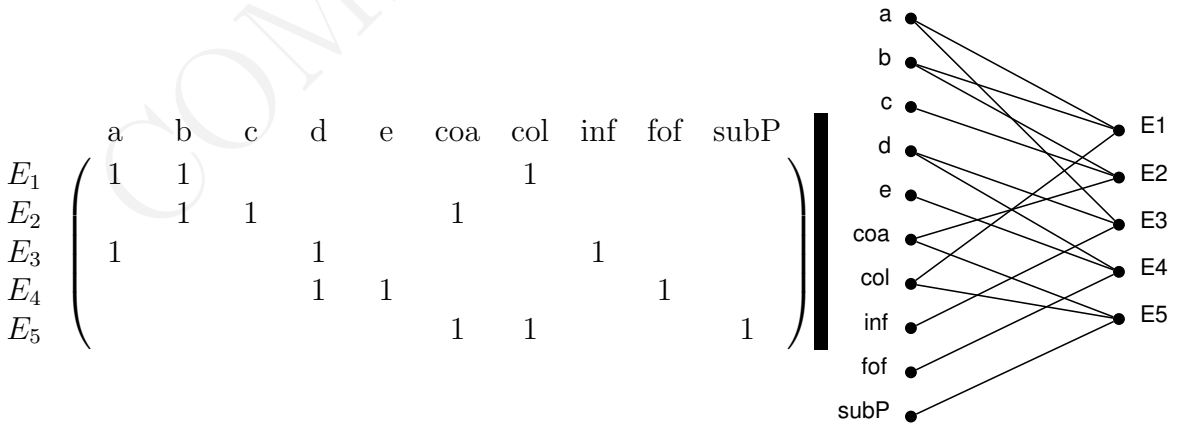



FIGURE 3.2. Incidence matrix representing the hypergraph of figure 3.1.(B) and the corresponding incidence graph.

RDF bipartite graphs have many desirable properties for developing intuitive mining algorithms. Thus, the we propose to use bipartite graphs to represent domain knowledge and data expressed in RDF.

**Example 3.1..2** (**Hypergraph incidence matrix and corresponding bipartite graph**)**.** Figure 3.2. (A) shows the incidence matrix according to the hypergraph in Figure 3.1. for Example 3.1..1. Figure 3.2. (B) shows the corresponding bipartite graph. Hypergraph incidence matrices represent membership of a node in an edge with a "1" in the corresponding entry.

Example 3.1..2 illustrates the general method that can be applied to all hypergraphs to transform to their bipartite graph form. In the case of a hypergraph representing an RDF Graph, since nodes in a RDF statement are ordered (subject followed by predicate then object), this ordering must be preserved in the incidence matrix. A *labeled bipartite graph* can be derived to further capture the ordering and roles of nodes.

**Definition 3.1..4** (**RDF labeled bipartite graph**)**.** In the hypergraph incidence matrix, instead of using "1/0" according to the occurrence of a node in an hyperedge, we choose to label them by S, P or O to represent the role (subject, predicate, or object) of the node in that underlying RDF statement–edge. Hence, when deriving the bipartite graph of a hypergraph incidence matrix, an edge will be added for every S, P, O entry of the matrix, and this edge will be labeled with the corresponding character (S, P, or O). Thus, the only difference between the graph derived from the incidence matrix of any hypergraph and an RDF Graph hypergraph is the fact that each edge has one out of three labels [**?** ].

In the rest of the proposal, when we use RDF bipartite graph, we mean RDF labeled bipartite graph for short.

**Example 3.1..3** (**RDF labeled bipartite graph**). Figure 3.3. illustrates an example of RDF hypergraph represented as labeled bipartite graph. The left side shows a portion of an ontology in biomedical domain on zebrafish anatomy [**?** ] visualized as a directed labeled graph. Two different relationships are depicted in the figure, namely, "subClassOf" and "part_of". The corresponding labeled bipartite graph representation is shown on the right side. Circle nodes are the *statement nodes* representing the RDF statements. Each statement node is connected to three *value nodes* representing the three components of a statement (subject, predicate, and object). Edge labels S, P, and O indicate the role of the value nodes in the statement.

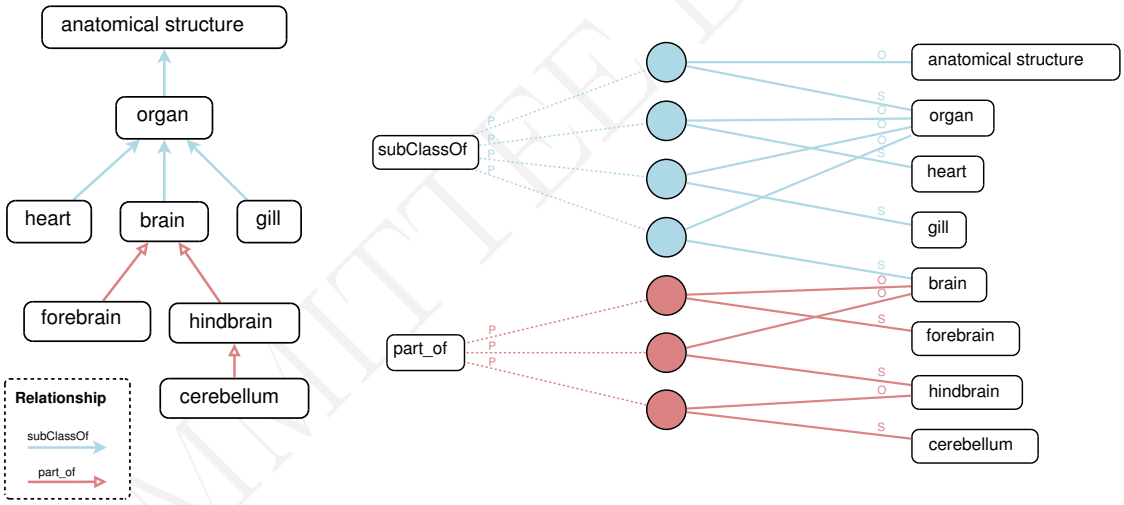



FIGURE 3.3. A portion of a zebrafish anatomy ontology represented as a directed labeled graph (A) and a RDF bipartite graph (B)

## 3.2. Graph Representation for Relational Structure

Various graph representation for relational structure has been proposed in literature to tackle different data mining tasks. In the case of frequent itemset mining, a set of objects with the co-occurrence relationship can be represented as directed or undirected graphs (see the example in Figure 2.2.).

Another way to represent relational structure is to first transform it to RDF, and by utilizing the graph nature of RDF, the relational structure can then be represented as graphs. Mapping from RDB to RDF has gained increasing attention and led to implementation of generic mapping tools as well as domain–specific applications. The W3C launched the RDB2RDF incubator group to explore issues involved in mapping RDB to RDF. A work-in-progress survey paper has been published documenting approaches in this field [**?** ].

A straightforward method for mapping RDB to RDF is discussed by Berners Lee [**?** ] as defined in the following.

**Definition 3.2..1** (**Context–independent mapping from RDB to RDF**)**.** Without linking to any explicit definition of domain semantics (such as those defined in domain ontologies), a RDB can be transformed to RDF following the steps below:

1. A RDB row is a RDF subject node.
2. The column name of a RDB table is a predicate node.
3. A RDB table cell is an object node.

Many systems leverage these mappings to automatically generate mappings between RDB and RDF. Even though these automatically generated mappings often do not capture complex domain semantics that are required by many applications, these mappings can serve as a useful starting point to create more customized, domain–specific mappings.

**Example 3.2..1** (**RDF bipartite graph for a nominal-valued RDB**)**.** Table 3.1. (A) shows a relational table with nominal features (columns). The table has $m$ rows, each one annotated on the border with labels $r_1 \dots r_m$, and $n$ columns named $f_1 \dots f_n$. Applying

the steps in Definition 3.2..1 for mapping RDB to RDF mapping, the corresponding RDF statements are listed in Table 3.1. (B). From these statements, a RDF bipartite graph is derived, shown in Figure 3.4., as the graph representation for the underlying relational table in Table 3.1. (A).

| | $f_1$ | $\cdots$ | $f_n$ |
|---|---|---|---|
| $r_1$ : | $v_{11}$ | $\cdots$ | $v_{1n}$ |
| $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $r_m$ : | $v_{m1}$ | $\cdots$ | $v_{mn}$ |

(A)

| $s$ | $p$ | $o$ |
|---|---|---|
| <$r_1$> | <$f_1$> | <$v_{11}$> |
| <$r_1$> | <$f_n$> | <$v_{1n}$> |
| <$r_m$> | <$f_1$> | <$v_{m1}$> |
| <$r_m$> | <$f_n$> | <$v_{mn}$> |

(B)

TABLE 3.1. An example of a relational table with nominal features (A) and its corresponding RDF triple form (B).

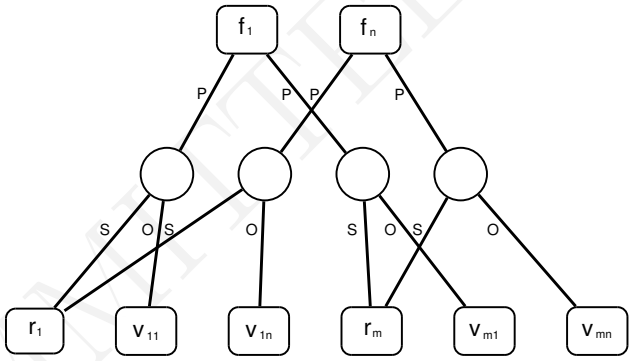



FIGURE 3.4.

For relational tables with binary (boolean) features, the RDF representation can be more compact. In some applications, only cells with positive ("1") values are of interest. In this case, an auxiliary predicate can be introduced to link a row and positive cell values in that row.

**Example 3.2..2 (RDF bipartite graph for a binary-valued RDB with positive values).** Table 3.2. (A) shows an $m$-by-$n$ relational table with binary features. We use

an auxiliary predicate <mentions> to denote a positive occurrence of a feature in one row, e.g., the statement <$r_1$> <mentions> <$f_n$> corresponds to the value "1" in the $n$-th feature in the first row. Consequently, the whole Table 3.2. (A) maps to only two RDF statements in Table 3.2. (B).

| | $f_1$ | $\cdots$ | $f_n$ |
|---|---|---|---|
| $r_1$ : | 0 | $\cdots$ | 1 |
| $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $r_m$ : | 1 | $\cdots$ | 0 |

(A)

| $s$ | $p$ | $o$ |
|---|---|---|
| <$r_1$> | <mentions> | <$f_n$> |
| <$r_m$> | <mentions> | <$f_1$> |

(B)

TABLE 3.2. An example relational table with binary features.

Using the auxiliary predicate (<mentions>) greatly simplifies the resulting RDF graph by reducing the number of distinct predicates from $n$, according to the process in Definition 3.2..1, to only 1. This has profound implications for developing efficient analysis and mining methods based on the RDF bipartite graph.

However, the auxiliary predicate is feasible only when linking a row node with its positive value nodes in a binary-valued scenario. If negative cell values are also of interest and needs to be included in the RDF graph, the trick shown in the following example can be performed so that we can still use a single auxiliary predicate to link both positive and negative values:

**Example 3.2..3** (**RDF bipartite graph for a binary-valued RDB with both positive and negative values**)**.** Table 3.3. (A) is derived from Table 3.2. (A) by adding a reverse column for each of its original columns, i.e., for each $f_i$, $i \in [1, n]$, a reverse $f'_i$ is created where $\forall k \in [1, m]$ , $v_{ki} = \neg v_{ki'}$. In this way, we can use the auxiliary predicate <mentions> to link to negative values by using the reverse column, because, for example, <$r_1$> <mentions> <$f'_1$> is equivalent to asserting ¬(<$r_1$> <mentions> <$f_1$>) .

Table 3.3. (B) shows the RDF statements based on Table 3.3. (A) which essentially captures information of both positive and negative values from Table 3.2. (A).

| | $f_1$ | $f'_1$ | $\cdots$ | $f_n$ | $f'_n$ |
|---|---|---|---|---|---|
| $r_1:$ | 0 | 1 | $\cdots$ | 1 | 0 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $r_m:$ | 1 | 0 | $\cdots$ | 0 | 1 |

(A)

| $s$ | $p$ | $o$ |
|---|---|---|
| <$r_1$> | <mentions> | <$f'_1$> |
| <$r_1$> | <mentions> | <$f_n$> |
| <$r_m$> | <mentions> | <$f_1$> |
| <$r_m$> | <mentions> | <$f'_n$> |

(B)

TABLE 3.3. An example expanded relational table with binary features.

The process of adding reverse columns to binary-valued RDB tables described in Example 3.2..3 can be extended to nominal-valued tables as well. By doing this we can achieve the desirable property of having only one predicate in the resulting RDF graph. The process is called RDB nominal value expansion as defined below.

**Definition 3.2..2** (**RDB nominal value expansion**)**.** In a nominal valued RDB table, for each feature $f_i$ taking values on the set $V_i = \{v_{i1}, v_{i2}, \ldots\}$, we define $|f_i|$ to denote the number of its distinct values, i.e. $|f_i| = |V_i|$. The RDB nominal value expansion process transforms each nominal feature $f_i$ to $|f_i|$ number of binary features $(f_{i1}, f_{i2}, \ldots, f_{i|f_i|})$. The value of $k$-th row in $f_{ij}, (j \in [1, |f_i|)$, is "1", if and only if $f_i$ takes the value $v_{ij}$ in the $k$-th row.

**Example 3.2..4** (**RDB nominal value expansion**)**.** Table 3.4. (A) shows an nominal-valued RDB table with concrete column names and values. We use the formula, Outlook={sunny, overcast, rainy}, to denote the set of distinct values the feature "Outlook" can take on. Similarly, we have Temperature={hot, mild, cool}, and Humidity={high, low}. Table 3.4. (B) shows the resulting table after nominal value expansion based on Definition 3.2..2, and based on which, we derive the RDF statements all using one single auxiliary predicate <mentions>, as partly shown in Table 3.4. (C).

| | O | T | H |
|---|---|---|---|
| $r_1$ : | sunny | hot | high |
| $r_2$: | rainy | cool | low |
| ⋮ | ⋮ | | |
| $r_m$ : | overcast | mild | low |

(A)

| | O_s | O_o | O_r | T_h | T_m | T_c | H_h | H_l |
|---|---|---|---|---|---|---|---|---|
| $r_1$ : | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $r_2$ : | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| ⋮ | ⋮ | | | | | | | |
| $r_m$ : | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

(B)

| $s$ | $p$ | $o$ |
|---|---|---|
| <$r_1$> | <mentions> | <O_s> |
| <$r_1$> | <mentions> | <T_h> |
| <$r_1$> | <mentions> | <H_h> |
| <$r_2$> | <mentions> | <O_r> |
| | ... | |

(C)

TABLE 3.4. An example nominal-valued relational table after nominal value expansion. (A) shows the original table where O stands for "outlook", T for "temperature" and "H" for humidity. (B) shows the expanded table. (C) shows the corresponding RDF triples derived from (B).

### 3.3. Combining Data Graphs and Ontology Graphs

In order to seamlessly interface data and domain knowledge in a mining framework, the information in both sources needs to be first associated. This is achieved by the process called *semantic annotation*. Semantic Annotation aims at assigning to the basic element of data links to formal semantic descriptions. Such elements should constitute the semantics of their source, for example, named entities in a document, certain part of an image depicting someone's head portrait.

Semantic annotation is crucial in realizing semantic data mining by bridging formal semantics in domain knowledge with data. What is more important, semantic annotation also enables a spectrum of new applications, such as indexing, retrieval, inference, categorization, query answering etc. These applications can also play vital roles in semantic data mining.

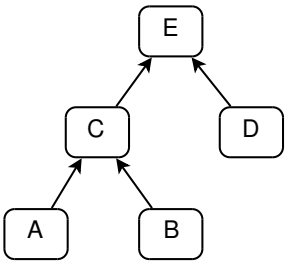
In the following, we assume data is annotated, meaning the links between entities in the data and formal semantic descriptions (such as those in ontologies) are established. A unified graph incorporating information from both data and domain knowledge can be created. Data mining algorithms dealing with such graph enjoys the benefits of a seamless integration of domain semantics. The following example shows the combination of an ontology graph and a data graph.

**Example 3.3..1** (**Combining an ontology graph and a data graph**)**.** Figure 3.5. (A) shows an simple ontology in some domain with only subsumption relationship defined for five entities (A–B). Figure 3.5. (B) shows a binary-valued RDB table in the same domain with the set of entities serving as features. We use the same entity labels in the ontology and the RDB table because we assume the mapping between ontology nodes and table features are pre-assigned or established by automatic annotation. Figure 3.6. (B) shows the RDF statements derived from both the ontology and the RDB table. Figure 3.6. (A) demonstrates the combined RDF bipartite graph.

Positions of nodes in Figure 3.6. (A) can be rearranged to form Figure 3.7.. This graph demonstrates a tripartite structure where row nodes ($r_1$–$r_5$) fall on one partition, column nodes (A–E) another, and statement nodes in between. A plethora of graph-theoretic algorithms and graph mining techniques can be leveraged to analyze the configuration of connecting paths in this graph to answer questions such as grouping in rows and columns (for solving the the task of clustering and association mining respectively). Predicate nodes can serve as a sign to introduce different weights to paths in order to distinguish different semantic information conveyed by those predicates.

Edges and statement nodes in Figure 3.7. are depicted in two colors to signify their sources of origin. Red edges and nodes denote information from the data (RDB table), and blue ones from the ontology. It can be seen from the graph that the contribution of

ontology is to introduce extra paths with different semantic types (The data is structured under the "mentions" relationship and the ontology is structured under the subsumption relationship). A mining algorithm that is able to deal with the data graph can be naturally extended without major modification to handle domain knowledge coded in the ontology.




(A)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $r_1$ : | 1 | 1 | 0 | 0 | 0 |
| $r_2$ : | 1 | 1 | 1 | 0 | 0 |
| $r_3$ : | 0 | 1 | 1 | 0 | 0 |
| $r_3$ : | 0 | 0 | 0 | 1 | 0 |

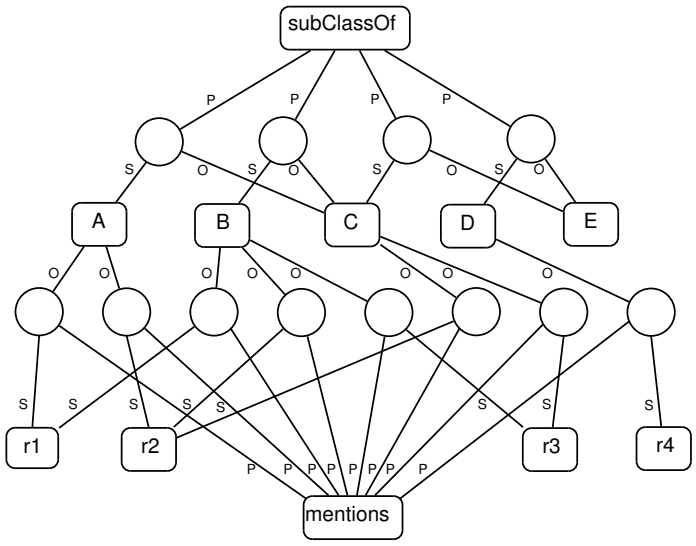
(B)

FIGURE 3.5. An example of a binary-valued relational table (B) about five concepts ("A"–"E"), and the ontological relationship among these concepts shown as a directed graph (A).

### 3.3.1. Representing Different Kinds of Ontological Semantics

In order to leverage the increasingly larger and richer collection of domain ontologies, especially in scientific fields such as the biomedical domain, we propose to develop methods for the representation of different types of ontological semantics which supports efficient semantic data mining. We propose to develop a weighting scheme to distinguish paths in the RDF bipartite graph that contains these different relationships such as class subsumption, part_of, and other general or domain–specific properties.

**Example 3.3..2** (**Assigning weights to different relationship**)**.** Figure 3.8. shows an example of RDF bipartite graph representing information on peoples (A–E) where multiple relationships can be identified to link them. E.g., A, B, C and D are linked by the coauthorship relationship, while D and E are linked by the more general collaboration




| $s$ | $p$ | $o$ |
|---|---|---|
| <A> | <subClassOf> | <C> |
| <B> | <subClassOf> | <C> |
| <C> | <subClassOf> | <E> |
| <D> | <subClassOf> | <E> |
| | | |
| <r1> | <mentions> | <A> |
| <r1> | <mentions> | <B> |
| <r2> | <mentions> | <A> |
| <r2> | <mentions> | <B> |
| <r2> | <mentions> | <C> |
| <r3> | <mentions> | <B> |
| <r3> | <mentions> | <C> |
| <r4> | <mentions> | <D> |

(A) (B)

FIGURE 3.6. The RDF bipartite graph representation (A) given triples shown in (B) based on the information described in Figure 3.5..

relationship (in fact, coauthorship is defined as a sub-property of collaboration in the ontology). A, B and C are professors, D and E are PhD students, and both professors and PhD students are researchers. In this complex lattice of relationships, we hope to distinguish these relationships by assigning task–specific weights to the related paths (e.g., as conveyed by different colorings in the graph). In order to achieve this, we propose to develop a framework for guiding the (semi-) automatic assignment of weights.

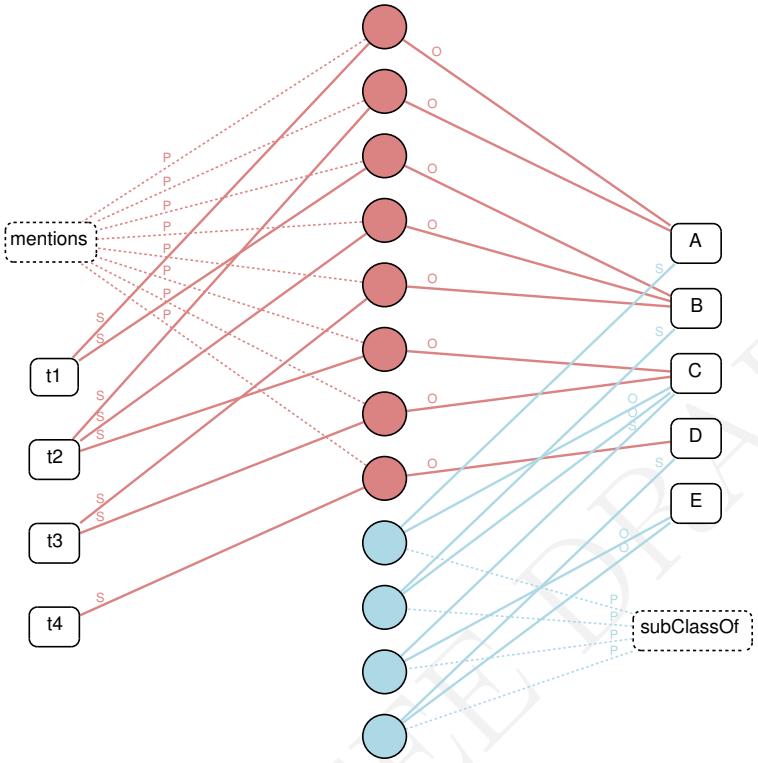



FIGURE 3.7. This figure show that, grouping the nodes according to whether they are row elements or column elements in Figure 3.5. (B), the bipartite graph shown in Figure 3.6. (A) can be further transformed to a tripartite graph.

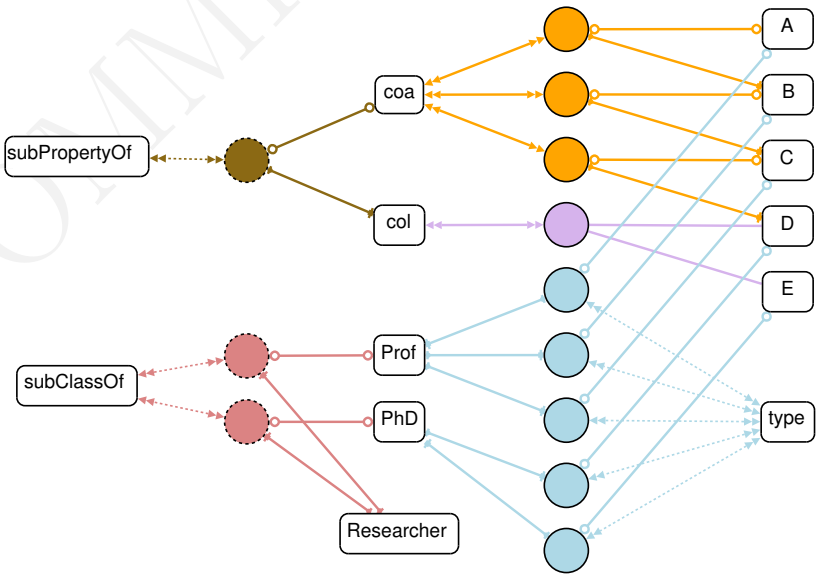



FIGURE 3.8.

# CHAPTER IV

# SEMANTIC ANNOTATION

Annotation in general (traditionally) is a summary made of information in some source such as document, online record, image, video and so forth. With the emergence and development of the Semantic Web, there have been increasing demands for *semantic annotation*, a kind of meta-data derived based upon specific knowledge about the world, rather than indifferent to any ontological commitments. In other words, semantic annotation aims at assigning to the basic element of information links to formal semantic descriptions [**?** ]. Such elements should constitute the semantics of their source, for example, named entities in a document, certain part of an image depicting someone's head portrait.

Semantic annotation is crucial in realizing semantic data mining by bridging formal semantics in Semantic Web meta-data with data. The majority of data underpinning a wide spectrum of data mining applications are stored in various formats, including structured sources such as relational databases (RDB) with their proven track record of scalability and reliability, or semi-structured sources such as spreadsheets with their advantage of low maintenance and cheaper overheads, or even unstructured sources such as text corpus. The problem of how to impart knowledge encoded in Semantic Web ontologies to all sorts of data becomes a major challenge in realizing the semantic data mining. We believe *semantic annotation* is the solution to this challenge. It aims at assigning semantic descriptions to elements of data. To ease the burden of common users that are not familiar with the Semantic Web, we develop a learning-based semantic search algorithm to suggest appropriate semantic descriptions for annotation.

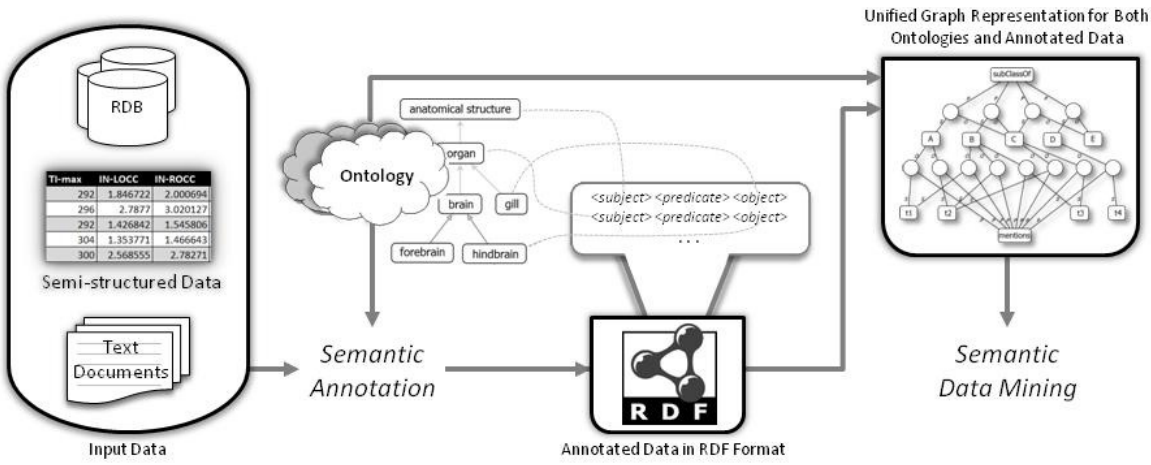



FIGURE 4.1. Data stored in different sources (e.g., relational databases, spreadsheets, texts etc) are semantically annotated. The output is in RDF format where elements of data are linked to formal semantic descriptions in ontologies. Annotated data and ontologies together are then represented in a unified graph representation for subsequent analysis in semantic data mining.

## 4.1. Automatic Annotation by Semantic Search

Semantic annotation is crucial in realizing semantic data mining by bringing meanings to data. The function of semantic annotation in the workflow of a semantic data mining system is illustrated in Figure 4.1. Annotating unstructured data (e.g., text) has been studied more extensively than annotating (semi-)structured data due to the proliferation of information extraction techniques that facilitate automatic entity recognition from text. Since large amount of data for knowledge discovery applications are stored in (semi-)structured sources, we focus on annotating (semi-)structured data.

The annotation process can be generally divided into two steps. The first is to establish mappings between existing Semantic Web terms and terms need to be annotated in the data. The second step is to come up with a local ontological structure constituting the semantic web terms to model the data.

Most of previous work in annotating (semi-)structured data focus on the second step. Some skip the first step and bootstrap the ontological terms and structure from the local data itself. For example, a number of systems that map data in RDB to RDF format leverage a set of rules such as "table to class and column to predicate". Bernes-Lee expands the rule as follows [**?** ]: 1) An RDB record is a RDF node. 2) The column name of an RDB table is an RDF predicate. And 3) An RDB table cell is a value. Other examples of rules involved in mapping RDB schema to OWL ontology include "foreign keys to object property and non-key attributes to datatype property". Similar idea has been adopted in annotating spreadsheets as well. Existing spreadsheet-to-rdf tools typically map spreadsheets to star-shaped RDF graphs, i.e., each row is an instance, with each column representing a property. Some tools try to express richer spreadsheet semantics, e.g., Han et al. developed a spreadsheet-to-rdf tool called RDF123 [**?** ] that allows users to define mappings to arbitrary graphs.

We argue that mapping RDB or spreadsheet to linked data (e.g., RDF) without referencing to existing semantic descriptions does not lend itself well to aiding semantic data mining. The automatically constructed self-contained local ontology may be applicable to describe a specific dataset but is most likely too rough to capture the full domain semantics that is necessary to express meaningful domain knowledge. Moreover, with the advent of the Semantic Web and pervasive connectivity, an increasing number of ontologies has been made widely available for reuse. These ontologies are created by thorough knowledge engineering process and should serve as better models for annotation. However, on the other hand, the sheer number of Semantic Web ontologies and lack of effective search functionality can lead to a huge hidden barrier for common users. Choosing proper Semantic Web ontologies and terms (classes and properties) requires familiarity with appropriate ontologies and the terms they define. There is very few

system that is able to provide automatic suggestions. To solve this problem, we propose a learning-based semantic search algorithm to suggest proper Semantic Web terms and ontologies for annotation given semantically related words and general domain and context information.

### 4.1.1. Proposed Learning-based Semantic Search Algorithm

In order to suggest suitable Semantic Web terms and ontologies for users to annotate their data, we propose a learning-based semantic search algorithm. We first submit a list of terms appeared in the schema of (semi-)structured data to our semantic search algorithm and then use the returned results for annotation. In a fully automatic setting, the search algorithm is configured to return the top-1 hit; while in an interactive setting, the search algorithm returns ordered top-k search results for users to decide. Previous semantic search algorithms leverage a variety of measures, including lexical and structural similarities (see details below) to rank Semantic Web documents according to how likely they can be semantically matched to the search terms. However, using any single measure alone may not be sufficient to achieve the optimal result. We propose to combine various measures to a weighted feature-based search model, where the weights are learned from training data. We believe the incorporation of learning techniques will improve the semantic search result.

#### 4.1.1.1. Feature-based Semantic Search Model.

Consider a set of ontologies $\mathbf{O} = \{O_1...O_m\}$ returned as the search result for a specific search term. Let $\mathbf{\Phi} = \{\phi_1(O_i)...\phi_m(O_m)\}$ be a vector of real-valued feature functions $\phi : O \mapsto \mathbb{R}$ that compute rank indicating how ontologies should be ordered in the search result. The one with the highest rank is the top-hit for a specific search. Let

$\mathbf{W} = \{w_1 ... w_m\}$ be a vector of real-valued weights associated with each feature. A score is computed for each ontology $O_i$ by taking the dot product of the features and weights:

$$\tau(O_i, \mathbf{W}) = \mathbf{\Phi} \cdot \mathbf{W} \ .$$

We can leverage a variety of ranking methods proposed in the literature as the feature functions $\mathbf{\Phi}$. For example, Alan et al. [**?** ] proposed four types of measurements to evaluate the ranks for ontologies given a list of search terms: The Class Match Measure (CMM) measures the coverage of an ontology of the given search terms; The Centrality Measure (CEM) is aimed to asses how representative a class is of an ontology according to the assumption that the more central a class is in the hierarchy, the more likely it is well analyzed and represented; Density Measure (DEM) examines how well a concept is represented by taking into consideration its neighboring concepts; And Semantic Similarity Measure (SSM) calculates the distance between classes in an ontology that match the search terms. The Swoogle search engine [**?** ][**?** ] weighs different types of links between Semantic Web data and rank them using link-based algorithms to evaluate the importance of Semantic Web objects at three levels of granularity: documents terms and RDF graphs. Maedche et al. [**?** ] described a two level similarity measure. The first level is the so called lexical comparison level, where a lexical similarity measure, called string matching (SM), is proposed based on Levenshtein's edit distance to compare two lexical entries $L_i$ and $L_j$: $SM(L_i, L_j) = max\left(0, \frac{min(|L_i|,|L_j|) - ed(L_i,L_j)}{min(|L_i|,|L_j|)}\right) \in [0, 1]$. The second level is the conceptual comparison level where two taxonomies are compared by examining the semantic cotopies of the sets of concepts from the two ontologies.

#### 4.1.1.2. Training Set.

Our algorithm to determine the vector of weights $\mathbf{W}$ requires a *training set* of known top hit in the search result (chosen by human):

$$\mathcal{T} = \{< \mathbf{O}^1, l_1 > ... < \mathbf{O}^n, l_n >\} ,$$

where each set of ontology namespaces $\mathbf{O}^i = \{O_1...O_k\}$ is associated with label $l_i \in \{1...k\}$, indicating which of the ontologies should be selected for annotating the specific term $t_i$ (i.e., $O_{l_i} \in O$ is the true ontology selected by human as the best choice for annotating the term). There are several ways to estimate $\mathbf{W}$ from the training set $\mathcal{T}$ as described below.

#### 4.1.1.3. Subgradient Descent.

We can view the weight learning as maximum margin structured learning problem. Given a training set and loss function, the learned $\mathbf{W}$ should score each known top-hit result $O_{l_i}$ higher than all other $O$ by at least $\mathcal{L}(O_{l_i}, O)$, where $\mathcal{L}$ is the loss function. Mathematically, this constraint is

$$\forall i, O \in \{\mathbf{O}^i \backslash O_{l_i}\}, \mathbf{W} \cdot \mathbf{\Phi}(O_{l_i}) \geq \mathbf{W} \cdot \mathbf{\Phi}(O) + \mathcal{L}(O_{l_i}, O) ,$$

where $\{\mathbf{O}^i \backslash O_{l_i}\}$ is the set of possible ontologies returned by a specific search query excluding the gold standard ontology $O_{l_i}$ chosen by human. We can express this constraint as the following convex program:

$$\min_{\mathbf{W}, \zeta^i} \frac{\lambda}{2} \|\mathbf{W}\|^2 + \frac{1}{d} \sum_{i=1}^{d} \zeta^i$$

$$s.t. \forall i, O \in \mathbf{O}, \mathbf{W} \cdot \mathbf{\Phi}(O_{l_i}) + \zeta_i \geq \mathbf{W} \cdot \mathbf{\Phi}(O) + \mathcal{L}(O_{l_i}, O) \ ,$$

where $\lambda$ is a regularization term that prevents overfitting. We can rearrange the convex program to show that the optimal $\mathbf{W}$ minimizes

$$c(\mathbf{W}) = \frac{1}{d} \sum_{i=1}^{d} r^i(\mathbf{W}) + \frac{\lambda}{2} \|\mathbf{W}\|^2 \ , \qquad \text{(Equation 4.1.)}$$

where $r_i(\mathbf{w}) = \max_{O \in \{O_{l_i}\}} (\mathbf{W} \cdot \mathbf{\Phi}(O) + \mathcal{L}(O_{l_i}, O)) - \mathbf{W} \cdot \mathbf{\Phi}(O_{l_i})$ . This objective function is convex but nondifferentiable. We can therefore minimize it with subgradient descent, an extension of gradient descent to nondifferentiable objective functions. The subgradient of Equation Equation 4.1. is

$$\lambda \mathbf{W} + \frac{1}{d} \sum_{i=1}^{d} (\mathbf{\Phi}(O^*_{l_i}) - \mathbf{\Phi}(O_{l_i})) \ ,$$

where $O^*_{l_i} = argmax_{O \in \{O_{l_i}\}} \mathbf{W} \cdot \mathbf{\Phi}(O) + \mathcal{L}(O_{l_i}, O)$ is the predicted top hit ontology. Based on these ideas, we can iteratively compute the subgradient of equation Equation 4.1. [**?** ] and converge at the optimal weights.

#### 4.1.1.4. Logistic Regression.

The second method is based on logistic regression (sometimes called maximum entropy classification). We modify the traditional logistic regression loss function to rank, rather than classify, instances. Let the binary random variable $C_i$ be 1 if and only if ontology $O_i$ is the gold standard chosen by human. Given $\mathbf{W}$ and $\mathbf{\Phi}$, we can compute the probability of $C_i$ as follows:

$$p\left(C_i = 1|\mathbf{O}, \mathbf{W}\right) = \frac{e^{\tau(O_i, \mathbf{W})}}{\sum_{O_j \in \mathbf{O}} e^{\tau(O_j, \mathbf{W})}} \ ,$$

where the score for ontology $O_i$ is normalized by the scores for every other ontologies. We can estimate $\mathbf{W}$ from the training set $\mathcal{T}$ by minimizing the negative log-likelihood of the data given $\mathbf{W}$:

$$\mathcal{L}\left(\mathbf{W}, \mathcal{T}\right) = -\sum_{O^i \in \mathcal{T}} \log p\left(C_{l_i}|\mathbf{O}, \mathbf{W}\right) \ . \qquad \text{(Equation 4.2.)}$$

Note that this is the sum of probabilities for each of the gold standard ontology for the current setting of $\mathbf{W}$. We can also add a Gaussian prior over $\mathbf{W}$ with fixed mean and variance to mitigate over-fitting. We can find the setting of $\mathbf{W}$ that minimizes Equation Equation 4.2. using limited-memory BFGS, a gradient ascent method with a second-order approximation [**?** ].

## 4.2. Annotation by Multiple Ontologies

The presence of heterogeneity among schemas supporting vast amount of information demands advanced solution for semantic integration of disparate data sources to facilitate interoperability and reuse of the information. The challenge is especially pronounced in many scientific domains where a massive amount of data are produced independently and thus each having their own data vocabulary. While manual integration is time-consuming and requires expensive specialized human capital, the development of automatic approaches becomes imminent to aid inter-institute collaborations. One purpose of the present paper is to suggest a method for solving a specific kind of ontology matching problem under some severe constraints that can cause traditional methods to be ineffective. The constraints that we deal with are, namely, 1) little-to-no string-based or linguistic similarity between terminologies, and 2) all numeric typed data instances. This

phenomenon is commonly seen in integrating scientific datasets which involves discovery of correspondences among distinct numeric-typed summary features ("attributes") that are used to characterize datasets that have been collected and analyzed in different research labs. We call this the *attribute matching* problem.

Our study of matching alternative attribute sets is closely related to the schema matching problem. According to the type of instance value, various instance-based approaches have been developed in previous research. For example, for textual attributes, a linguistic characterization based on information retrieval techniques can be applied [**?** ]; for nominal attributes, evaluation of the degree of overlap of instance values is a preferred approach. Larson et al. [**?** ] and Sheth et al. [**?** ] discussed how relationships and entity sets could be integrated primarily based on their domain relationships. Similarity of partially overlapped instance set can be also calculated based on measures such as Hamming distance and Jaccard coefficient; for numeric attributes, most methods use aggregated statistics to characterize the attributes, e.g., 'SSN' and 'PhonNo' can be distinguished based on their respective patterns [**?** ]. Hybrid systems that combine several approaches to determine matching often achieve better performance. For example, SemInt [**?** ] is a comprehensive matching prototype exploiting up to 15 constraint-based and 5 content-based matching criteria. The LSD (Learning Source Descriptions) [**?** ] system uses several instance-level matchers (learners) that are trained during a preprocessing step. The iMAP [**?** ] system uses multiple basic matchers, called searches, e.g., text, numeric, category, unit conversion, each of which addresses a particular subset of the match space.

Due to the nature of many scientific datasets, we face several unique challenges. First, the data under study are semi-structured, thus invalidating those matching methods that presume a complete, known-in-advance schematic structure. In addition, totally

different labels (usually acronyms or pseudowords) are widely adopted for the same or similar metrics, rendering lexical similarity-based methods unsuitable. Moreover, an important limitation of previous instance-based matching methods is their inability to handle numerical instances appropriately in certain domain applications. They use statistical characterization extracted from the numerical instances, such as range, mean and standard deviation, to determine match. However such information is too rough to capture patterns in data that are crucial in determining the correspondence.

To solve this difficulty, in the present paper, viewing the two matching problems as combinatorial optimization problems with distinct yet interrelated objective functions, we propose a novel approach using a multi-objective heuristics to discover attribute matching and cluster matching simultaneously. The objectives in the optimization are to minimize distances of attribute matching and cluster matching respectively. We explore the widely used simulated annealing algorithm as the metaheuristics algorithm and briefly compare its performance with the evolutionary multi-objective algorithm in experiments.

*Density Profile*: To represent clusters using density profiles, the attribute's range in each cluster is first discretized into a number of bins, and the similarity between two clusters corresponds to the number of points of each cluster falling within these bins. The formal definition for this number of points is the *density* of an attribute-bin region for cluster $c_k$ in clustering $C$, denoted as $dens_C(k, i, j)$. It refers to the number of points in the region $(i, j)$—the $j$-th bin of the $i$-th attribute—that belongs to the cluster $c_k$ of clustering $C$. For example, for clustering $C$ in Fig. 4.2., $dens_C(1, 1, 1) = 8$, because there are 8 data points in region $(1, 1)$—the first bin of the first attribute $x$—that belongs to the first cluster $c_1$.

The density profile vector $V_C$ for a clustering $C$ is formally defined as an ordered tuple:

$$V_C = \Big[ dens_C(1,1,1),\ \ldots,\ dens_C(1,1,Q),$$
$$dens_C(1,2,1),\ \ldots,\ dens_C(1,M,Q),$$
$$dens_C(2,1,1),\ \ldots,\ dens_C(N,M,Q) \Big], \qquad \text{(Equation 4.3.)}$$

where $Q$ is the number of bins in each of the $M$ attributes, and $N$ is the number of clusters in $C$.

*The ADCO measure*: After the density profile vectors of two clusterings $C$ and $C'$ are obtained, the degree of similarity between $C$ and $C'$ can be determined by calculating the dot product of the density profile vectors: $sim(C, C') = V_C \cdot V_{C'}$ .

The $ADCO(C, C')$ measure is defined as $sim(C, C')$ normalized by the maximum achievable similarity when using either of the two clusterings:

$$ADCO(C, C') = \frac{sim(C, C')}{NF(C, C')},$$

where $NF(C, C') = max\big[sim(C, C),\ sim(C', C')\big]$.

### 4.2.1. The Multi-Objective Simulated Annealing Framework

**Problem Definition:** We tackle two integration tasks in this work, namely, the attribute matching and cluster matching problems. We cast the dual matching problems to a multi-objective optimization problem so that the matchings can be solved simultaneously. The two objective functions to be optimized are defined as the total distance of matched elements in attribute and cluster matching respectively. To this end,

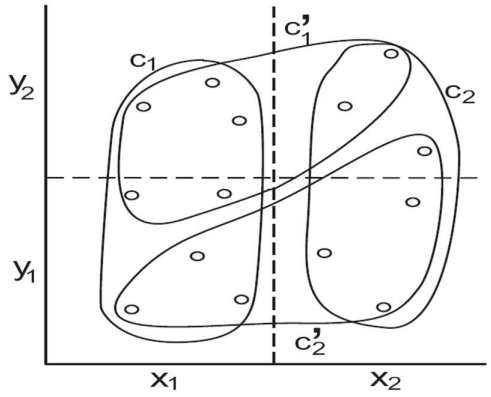



FIGURE 4.2. Two clusterings $C = \{c_1, c_2\}$ and $C' = \{c'_1, c'_2\}$. Two attributes $X$ (attribute 1) and $Y$ (attribute 2) are discretized into 2 bins each. See [**?** ] for details.

we explore methods to represent attributes and clusters so that distance measure can be reasonably defined. We assume that the optimal matching lies at the Pareto front in this multi-objective problem.

We use metaheuristics search algorithm to solve this multi-objective optimization problem. In the following we describe the widely used simulated annealing algorithm and how it can be adapted to multi-objective optimization and applied to solve the matching problems. Later in the Experiment Section, we briefly describe an evolutionary multi-objective algorithm and compare their performance.

To solve the dual matching problems, we adopt a strategy of multi-objective simulated annealing (MOSA) described in [**?** ], in which the acceptance criterion in the simulated annealing process is established based on the idea of Pareto-domination based fitness. Fitness of a solution is defined as one plus the number of dominating solutions in Pareto-optimal set. The larger the value of fitness, the worse is the solution. Initially, fitness difference between the current and the generated solution is less and the temperature is high so any move is accepted due to both of them. This gives a way to explore the full solution space. As the number of iterations increases, temperature

decreases and fitness difference between the current and generated solutions may increase. Both of them make the acceptance move more selective and it results in a well-diversified solution in true Pareto-optimal solutions. Details of our adaptation of the above multi-objective simulated annealing framework is outlined in Algorithm 1.

**Algorithm 1** Multi-Objective Simulated Annealing

**Input:** Empty Pareto-optimal set of solutions $\mathbf{\Sigma}$
**Input:** Empty current decision vector $\mathbf{X} = [x_a, x_c]$
**Input:** Initial temperature $T$
  $count = 0$
  **while** $T > threshold$ **do**
    $initialize(\mathbf{X})$
    Put $\mathbf{X}$ in $\mathbf{\Sigma}$
    $\mathbf{X'} = generate\_solution(\mathbf{X})$
    $S_{\mathbf{X'}} = evaluate\_solution(\mathbf{X'})$
    $\Delta S = S_{\mathbf{X'}} - S_X$
    **if** $r = rand(0,1) < exp(\frac{-\Delta S}{T})$ **then**
      $\mathbf{X} = \mathbf{X'}$
      $S_X = S_{X'}$
    **end if**
    count = count + 1
    //Periodically restart
    **if** $count == restart\_limit$ **then**
      $\mathbf{X} = select\_random\_from\_Pareto(\mathbf{\Sigma})$
      continue
    **end if**
    $reduce\_temperature(T)$
  **end while**

Mathematically, the processes involved in the proposed multi-objective simulated annealing framework can be defined as follows.

$$X = [x_a, x_c]$$

$$F = [f_a, f_c]$$

$$P_a([x_a^{(n-1)}, x_c^{(n-1)}]) = [x_a^{(n)}, x_c^{(n-1)}]$$

$$P_c([x_a^{(n-1)}, x_c^{(n-1)}]) = [x_a^{(n-1)}, x_c^{(n)}]$$

$$G_{c|a}([x_a^{(n)}, x_c^{(n-1)}]) = [x_a^{(n)}, x_c^{(n)}]$$

$$G_{a|c}([x_a^{(n-1)}, x_c^{(n)}]) = [x_a^{(n)}, x_c^{(n)}]$$

$$G \circ P([x_a^{(n-1)}, x_c^{(n-1)}]) = [x_a^{(n)}, x_c^{(n)}]$$

$X$ is the decision vector that contains two variables for attribute matching, $x_a$, and cluster matching, $x_c$, respectively (details in Section 4.2.2.). $F$ is the objective function vector that contains two criterion functions ($f_a$ and $f_c$) to evaluate attribute matching and cluster matching decisions (details in Section 4.2.4.). $P$ is the random perturbation function that takes a decision vector in the $(n-1)$th iteration and partially advances it to the $n$th iteration (we use $P_a$ or $P_c$ to distinguish between the random selections). The partial candidate decision generation function $G$ takes the output of $P$ and fully generate a decision vector for the $n$th iteration (by advancing the left-out variable in $P$ to its $n$th iteration). Thus, the compound function $G \circ P$ fulfils the task of generating an $n$th-iteration candidate decision vector given the $(n-1)$th one (details in Section 4.2.5.2.).

### 4.2.2. Decision Variable

The domains of the decision variables in the matching problems take values on a permutation space. In other word, by formalizing the problem of finding correspondent

elements of two sets $S$ and $S'$ of cardinality $n$ as an optimization problem, the solution is completely specified by determining an optimal permutation of $1, \ldots, n$. For instance, for two sets of three elements, their indexes range over $\{0, 1, 2\}$. Applying a permutation $\pi = \{2, 0, 1\} \in S_3$ on $S'$ can be viewed as creating a mapping (bijection) from elements on the new positions of $S'$ to elements on the corresponding positions in $S$. In this example, the permutation $\pi$ on $S'$ specifies the following correspondences: $S_0 \leftrightarrow S'_2$, $S_1 \leftrightarrow S'_0$, and $S_2 \leftrightarrow S'_1$.

Formally, let $P_n$ ($n \in \mathbb{N}$) be the symmetric group of all permutations of the set $\{1, 2, \ldots, n\}$. Given two sets $S$ and $S'$ with the same cardinality of $n$, performing identity permutation on one set and an arbitrary permutation $\pi \in S_n$ on the other specifies a matching (or mathematically speaking, mapping) between the two sets. In the multi-objective optimization formalism for solving attribute matching and cluster matching problems, the decision vector has two variables: $X = [x_a, x_c]$. If we have $M$ attributes and $N$ clusters to match respectively, then $x_a \in P_M$ and $x_c \in P_N$.

### 4.2.3. Data Representation

The central objects of interest in our study, namely, the numeric-typed attributes and clusters, need to be represented in ways that meaningful quantities can be defined to measure the "goodness" of a matching decision. To this end, we propose to use the *segmented statistical characterization* to represent attributes, and the *density profiles* to represent clusters. Details of these representations are described below.

#### 4.2.3.1. Representation of Attributes:

Numeric-typed attributes can be represented by the segmented statistical characterization, in which data instances are first partitioned into groups (e.g., through

unsupervised clustering) and then characterized by a vector of indicators, each denoting a statistical characterization of the corresponding group. For example, if values of an attribute $A$ are clustered into $n$ groups, then it can be represented by a vector of segmented statistical characterization as follows:

$$V_A = \Big[\mu_1, \mu_2, \ldots, \mu_n\Big],$$

where we choose the mean value $\mu_i$ for cluster $i$ as the statistical indicator in our implementation.

#### 4.2.3.2. Representation of Clusters:

Clusters can be represented by density profiles [**?** ] as described in Section **??**. The attribute's range in each cluster is discretized into a number of bins, and the similarity between two clusters corresponds to the number of points of each cluster falling within these bins. Given this, density profile vector $V_C$ for a clustering $C$ is formally defined as an ordered tuple by Equation Equation 4.3. where $dens_C(k, i, j)$ refers to the number of points in the region $(i, j)$—the $j$-th bin of the $i$-th attribute—that belongs to the cluster $c_k$ of clustering $C$.

### 4.2.4. Objective Functions

The objective functions in the attribute matching and cluster matching problems are criteria to evaluate the "goodness" of matchings. We use the sum of pair-wise distances between matched elements (see Figure **??** for example) as the objective function. Given this, to determine the form of objective functions amounts to defining proper pair-wise

distance measures for the attribute and cluster matching problems respectively, as detailed in the following.

#### 4.2.4.1. Distance function between two attributes

The pairwise distance between two attributes are defined as the Euclidean distance between their segmented statistical characterization vectors, and $f_a$ calculates the sum of pair-wise distances under the attribute matching specified by $x_a$:

$$f_a(x_a) = \sum_{k=1}^{M} \mathcal{L}\left((V_a)^k,\ (V'_a)^{x_a(k)}\right) = \sum_{k=1}^{M} \sqrt{\sum_{i=1}^{N} \left(\mu_i^k - (\mu')_i^{x_a(k)}\right)^2}\ , \qquad \text{(Equation 4.4.)}$$

where $x_a \in P_M$.

#### 4.2.4.2. Distance function between two clusters

The ADCO similarity described in Section **??** can be transformed to a distance defined as follows [**?** ]:

$$D_{ADCO}(C, C') = \begin{cases} 2 - ADCO(C,C') & ,\ if\ C \neq C' \\ 0 & ,\ otherwise \end{cases} \qquad \text{(Equation 4.5.)}$$

We use $D_{ADCO}$ as the pair-wise distance between two clusters under the density profile representation, and $f_c$ calculates the sum of pair-wise distances under the cluster matching

specified by $x_c$

$$f_c(x_c) = \sum_{k=1}^{N} D_{ADCO}\left((V_c)^k,\ (V_c')^{x_c(k)}\right)$$
$$= \sum_{k=1}^{N}\left(2 - \sum_{i=1}^{M}\sum_{j=1}^{Q}\left(dens(k,i,j)\times dens(x_c(k),i,j)\right)\Big/ \max\left[\sum_{i=1}^{M}\sum_{j=1}^{Q} dens(k,i,j)^2, \sum_{i=1}^{M}\sum_{j=1}^{Q} dens(x_c(k),i,j)^2\right]\right), \quad \text{(Equation 4.6.)}$$

where $x_c \in P_N$.

### 4.2.5. Generation of New Solution

In each iteration of the simulated annealing process, we randomly generate candidate decision in the neighborhood of the last-iteration decision by applying two consecutive processes, namely, the random perturbation and the partial candidate decision generation, as described below.

#### 4.2.5.1. Random Perturbation:

In each iteration, we select at random one variable (either $x_a$ or $x_c$) in the decision vector and perturb it by randomly swapping two positions in the selected variable. This advances that variable from $(n-1)$th iteration to $n$th iteration. Then the following partial candidate generation process is carried out to bring the other variable also to $n$th iteration.

#### 4.2.5.2. Partial candidate decision generation

Given $x_c^{(n)}$, derive $x_a^{(n)}$:

$$\begin{aligned} x_a^n &= \arg\min_{\pi} f_a(\pi, x_c^{(n)}) \\ &= \arg\min_{\pi} \sum_{k=1}^{M} \mathcal{L}\Big( (V_a)^k,\ (V_a')^{\pi(k)} \Big) \\ &= \arg\min_{\pi} \sum_{k=1}^{M} \sqrt{\sum_{i=1}^{N} \left( \mu_i^k - (\mu')^{\pi(k)}_{x_c^{(n)}(i)} \right)^2} \end{aligned} \qquad \text{(Equation 4.7.)}$$

Given $x_a^{(n)}$, derive $x_c^{(n)}$:

$$\begin{aligned} x_c^n &= \arg\min_{\pi} f_c(\pi, x_a^{(n)}) \\ &= \arg\min_{\pi} \sum_{k=1}^{N} D_{ADCO}\Big( (V_c)^k,\ (V_c')^{\pi(k)} \Big) \\ &= \arg\max_{\pi} \sum_{k=1}^{N} \left( \sum_{i=1}^{M} \sum_{j=1}^{Q} \left( dens(k,i,j) \times dens(\pi(k), x_a^{(n)}(i), j) \right) \Big/ \right. \\ &\left. \max\left[ \sum_{i=1}^{M} \sum_{j=1}^{Q} dens(k,i,j)^2, \sum_{i=1}^{M} \sum_{j=1}^{Q} dens(\pi(k), x_a^{(n)}(i), j)^2 \right] \right) \end{aligned} \qquad \text{(Equation 4.8.)}$$

To calculate $\pi$ that satisfies equations Equation 4.7. and Equation 4.8., rather than iterating through all possible permutations, we can consider the equation as a minimum-cost assignment problem. Table **??**(A), for example, illustrates a distance table between two attribute sets $A$ and $A'$. Matching of the two sets can be considered as an assignment problem where the goal is to find an assignment of elements in $\{A_i\}$ to those in $\{A_i'\}$ that yields the minimum total distance without assigning each $A_i$ more than once. This problem can be efficiently solved by the Hungarian Method in polynomial time

of $O(K_{min}^3)$ [**?** ]. It is worth noting that by formulating the problem as the assignment problem, we assume the matching between two sets to be a one-to-one function.

# CHAPTER V

# GRAPH-BASED DATA MINING

## 5.1. Overview

By developing the unified representation for both data and domain knowledge, and utilizing ontological annotations, we can produce one RDF hypergraph, which serves as the basis for perform semantic data mining in a systematic way. Given this, the main research challenge is how to utilize the data and ontology together for semantic data mining. In this dissertation, we focus on three fundamental data mining tasks: *frequent pattern mining*, *classification*, and *clustering*:

**Frequent Pattern Mining**: Frequent itemsets play an essential role in many data mining tasks that try to discover interesting relations between features in large databases. In our preliminary work [**?** ], we develop a hypergraph-based method to discover semantically associated itemset (features connected by indirect links) using only the data itself. With additional information from ontology (domain knowledge), the unified RDF hypergraphs will enable us to discover hidden association between entities, between entities and concepts, and between concepts. Intuitively, these associations are defined in terms of the paths linking the nodes and the node labels should be taken into consideration as they represent different semantics.

## 5.2. Hypergraph-based Data Mining

### 5.2.1. Graph-based similarity

**Average Commute Time Similarity** $s_{CT}$

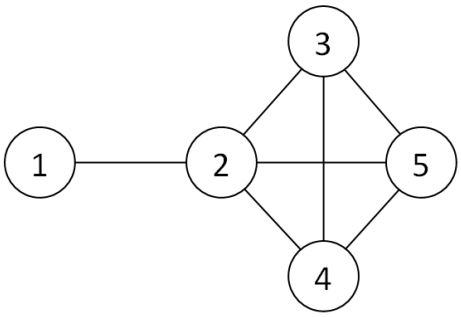



FIGURE 5.1. .

| Euclidian Distance | | | | | | Commute Distance | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Index | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 1 | 1.85 | 1.85 | 2.41 | 0 | 12.83 | 19.79 | 19.79 | 20.34 |
| 2 | 1 | 0 | 1 | 1 | 1.41 | 12.83 | 0 | 6.96 | 6.96 | 7.51 |
| 3 | 1.85 | 1 | 0 | 1.41 | 1 | 19.79 | 6.96 | 0 | 7.51 | 6.96 |
| 4 | 1.85 | 1 | 1.41 | 0 | 1 | 19.79 | 6.96 | 7.51 | 0 | 6.96 |
| 5 | 2.41 | 1.41 | 1 | 1 | 0 | 20.34 | 7.51 | 6.96 | 6.96 | 0 |

TABLE 5.1. .

As already mentioned, the commute-time distance $n(i, j)$ between two nodes $i$ and $j$ has the desirable property of decreasing when the number of paths connecting the two nodes increases and when the length of paths decreases. This is indeed an intuitively satisfying property of the effective resistance of the equivalent electrical network [**?** ]. The usual shortest-path distance (also called geodesic distance) does not have this property: the shortest-path distance does not capture the fact that strongly connected nodes are closer than weakly connected nodes.

### 5.2.2. Mining Semantic Associations

In this section, we present our preliminary study for discovering semantically associated itemsets based on hypergraph. The goal of this study is to show that, using graph based formalism, we can obtain interesting patterns that are unable to be captured

| | A | B | C | D |
|---|---|---|---|---|
| $e_1$: | 1 | 1 | 1 | 0 |
| $e_2$: | 0 | 1 | 1 | 1 |
| $e_3$: | 1 | 0 | 1 | 1 |

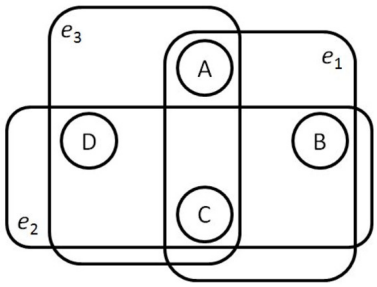



FIGURE 5.2. .

by traditional methods. Specifically, we propose to construct a hyperedge for each tuple. The relational attributes constitute the universe of vertices in the hypergraph. Given the hypergraph representation of relational structure, our approach starts by first generating semantically associated 2-itemsets. A 2-itemset $\langle i, j \rangle$ is considered semantically associated if the hypergraph-based similarity measure $s(i, j)$ exceeds some threshold. In the following subsections, we propose two similarity measures $s_{CT}$ and $s_{L+}$ based on, respectively, the average commute time distance on hypergraph and the inner-product-based representation of the pseudoinverse of Hypergraph Laplacian. Given discovered semantically associated 2-itemsets, we propose a hypergraph expansion method along with two search strategies, namely, the clique and connected component search, in the resulting graph for finding semantically associated $k$-itemsets ($k > 2$).

#### 5.2.2.1. Methods for Generating 2-itemsets

In the following we describe two similarity measures that define the strength of bond between a pair of semantically associated items.

To compute commute-time distance between vertices in a hypergraph, we need to first define the combinatory hypergraph Laplacian **L**. It follows from Zhou et al's

formalism of normalized hypergraph Laplacian in Equation Equation 2.1. that:

$$\mathbf{L} = \mathbf{D}^{1/2}\mathcal{L}\mathbf{D}^{-1/2} = \mathbf{D}_v - \mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T \qquad \text{(Equation 5.1.)}$$

The average commute time $n(i,j)$ on simple graph can be computed in closed form from the Moore-Penrose pseudoinverse of $\mathbf{L}$ [**?** ], denoted by $\mathbf{L}^+$ with elements $l_{ij}^+ = [\mathbf{L}^+]_{ij}$. It can be shown that $n(i,j)$ on hypergraph can be calculated in the same manner. The pseudoinverse $\mathbf{L}^+$ is given by the following equation:

$$\mathbf{L}^+ = (\mathbf{L} - \mathbf{e}\mathbf{e}^T/n)^{-1} + \mathbf{e}\mathbf{e}^T/n, \qquad \text{(Equation 5.2.)}$$

where $\mathbf{e}$ is a column vector made of 1s (i.e., $\mathbf{e} = [1, 1, \ldots, 1]^T$). The formula for the computation of $n(i,j)$ takes the form of the following equation:

$$n(i,j) = V_G(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+), \qquad \text{(Equation 5.3.)}$$

where $V_G = tr(\mathbf{D}_v)$ is the volume of the hypergraph. If we define $\mathbf{e}_i$ as the $i$th column of $\mathbf{I}$ (i.e., $\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^T$), Equation Equation 5.3. can be transformed to:

$$n(i,j) = V_G(\mathbf{e}_i - \mathbf{e}_j)^T\mathbf{L}^+(\mathbf{e}_i - \mathbf{e}_j), \qquad \text{(Equation 5.4.)}$$

Since $n(i,j)$ is a distance, it is straightforward to convert it to a similarity measure $s_{CT}(i,j)$ by normalize it to unit range and subtract from 1.

**Pseudoinverse-based Inner-Product Similarity** $s_{L+}$

Equation Equation 5.4. can be mapped into a new Euclidean space that preserves the commute time distance:

$$\begin{aligned} n(i,j) &= V_G(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \\ &= V_G(\mathbf{x}'_i - \mathbf{x}'_j)^T (\mathbf{x}'_i - \mathbf{x}'_j) \\ &= V_G \|\mathbf{x}'_i - \mathbf{x}'_j\|^2, \end{aligned} \qquad \text{(Equation 5.5.)}$$

where $\mathbf{x}'_i = \mathbf{\Lambda}^{1/2}\mathbf{U}^T\mathbf{e}_i$, $\mathbf{U}$ is an orthonormal matrix made of eigenvectors of $\mathbf{L}^+$ (ordered in decreasing order of corresponding eigenvalue $\lambda_k$) and $\mathbf{\Lambda} = \mathbf{Diag}(\lambda_k)$. In this way, the transformed node vectors $\mathbf{x}'_i$ are exactly separated in the new $n$-dimensional Euclidean space. From this definition, it follows that $\mathbf{L}^+$ is the matrix containing inner products of the transformed vectors $\mathbf{x}'_i$ as shown below:

$$\begin{aligned} \mathbf{x}'^T_i \mathbf{x}'_j &= (\mathbf{\Lambda}_i^{1/2}\mathbf{x}_i)^T \mathbf{\Lambda}_j^{1/2}\mathbf{x}_j = \mathbf{x}_i^T \mathbf{\Lambda} \mathbf{x}_j \\ &= \mathbf{e}_i^T \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\mathbf{e}_j = \mathbf{e}_i^T \mathbf{L}^+ \mathbf{e}_j = l_{ij}^+. \end{aligned} \qquad \text{(Equation 5.6.)}$$

Therefore, $\mathbf{L}^+$ can be considered as a similarity matrix for the nodes—that is

$$s_{L^+}(i,j) = l_{ij}^+ \,. \qquad \text{(Equation 5.7.)}$$

The inner-product-based similarity measures are well-studied for the vector-space model of information retrieval. It has been shown that when computing proximities between documents, inner-product-based measures outperform Euclidean distances [**?** ].

#### 5.2.2.2. Effective Computation

In high dimensional data sets, the computations of the Hypergraph Laplacian and the pseudoinverse becomes intractable. We discuss two approaches to mitigate this scalability problem.

To compute Hypergraph Laplacian $\mathbf{L}$ in Equation Equation 5.1. requires multiplication of hypergraph incidence matrices $\mathbf{H}$ and its transpose $\mathbf{H}^T$. Since $\mathbf{H}$ grows in proportion to the size of underlying transaction data (each node corresponds to a column and each hyperedge corresponds to a row), it eventually becomes unable to fit in memory when the size exceeding a certain amount. In this case the computation can still be carried out using a block partitioned matrix product by performing operations only on the submatrices of tractable sizes. Owing to the fact that, in most cases, $|V|$ is much smaller than $|E|$, $\mathbf{H}$ can then be partitioned into $s$ vertical stripes and the square matrix $\mathbf{D}_e$ into $s$ diagonal blocks. The multiplication in Equation Equation 5.1. can be calculated by $\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^T = \sum_{\gamma=1}^{s} \mathbf{H}_\gamma \mathbf{D}_{e\gamma}^{-1} \mathbf{H}_\gamma^T$. Note that $\mathbf{H}$ is sparse in many applications. This property can be exploited to gain high performance and due to its importance much effort has been devoted to the study resulting a number of libraries and routines from which we can leverage.

As the number of nodes grows, to compute pseudoinverse in closed form using Equation Equation 5.2. also becomes intractable. A procedure based on Cholesky factorization to compute $\mathbf{L}^+$ for large sparse matrices [**?** ] is proved useful. It allows to compute $\mathbf{L}^+$ in a column-by-column manner. In particular, the procedure involves the following steps for computing the $i$th column of $\mathbf{L}^+$:

1. Compute the projection $\mathbf{y}_i$ of base vector $\mathbf{e}_i$ on the column space of $\mathbf{L}$.

2. Find a solution $l_i^{*+}$ of the linear system $\mathbf{L}\mathbf{l} = \mathbf{y}_i$.

3. Project $l_i^{*+}$ on the row space of $\mathbf{L}$ to get $l_i^{+}$.

Since $\mathbf{L}$ is symmetric, its row space is the same as column space. The projection in step 1 and 2 can be represented by the matrix $(\mathbf{I} - \mathbf{e}\mathbf{e}^T/n)$. The equation in step 2 can be solved by first solving a reduced linear system: $\hat{\mathbf{L}}\hat{\mathbf{l}} = \hat{\mathbf{y}}_i$, where $\hat{\mathbf{L}}$, $\hat{\mathbf{l}}$, and $\hat{\mathbf{y}}$ are obtained respectively by removing the last row from $\mathbf{l}$, $\mathbf{y}$, and last row and column from $\mathbf{L}$. We observe that $\hat{\mathbf{L}}$ is full rank and positive definite and hence is able to be decomposed using the Cholesky factorization, $\hat{\mathbf{L}} = \mathbf{R}\mathbf{R}^T$. Since $\mathbf{R}$ is lower-triangular, one solution of $\hat{\mathbf{L}}\hat{\mathbf{l}} = \mathbf{R}\mathbf{R}^T\hat{\mathbf{l}} = \hat{\mathbf{y}}_i$ can be efficiently obtained by two back-substitutions. After solving the reduced linear system, the solution to the original equation in step 2 is therefore $(\mathbf{l}_i^{*+}) = [\hat{\mathbf{l}}_i^{*+}, 0]^T$. With the help of this technique, we are able to analyze datasets of a million rows and 10 thousand columns.

#### 5.2.2.3. Methods for Generating $k$-itemset ($k > 2$)

Now, we consider finding semantically associated $k$-itemset ($k > 2$) from given 2-itemsets. As is common in hypergraph theory, we can associate an induced graph $G(H)$ with every hypergraph $H$ by expanding every hyperedge $e$ in $H$ to a clique in $G(H)$. Edges in the induced graph $G(H)$ can be called *subedges* to avoid unnecessary confusion. We can further construct a pruned graph $G'(H)$ from $G(H)$ by applying the following inclusion rule on each subedge: the similarity between the incident nodes of a subedge has to be greater than a user-specified threshold $\theta$. In formal definition, given a hypergraph $H = (V, E)$, the pruned subgraph is $G'(H) = \{V, E'\}$ where

$$\begin{aligned} E' = \{(u, v) \in V^2 \,:\, & u \neq v \text{ and} \\ & u, v \in e \text{ for some } e \in E \text{ and} \\ & s(u, v) > \theta\}. \end{aligned}$$

Given $G'(H)$, finding semantically associated $k$-itemset ($k > 2$) can be formulated into two ways: finding cliques or connected components in $G'(H)$.

#### 5.2.2.4. Cliques of $G'(H)$

Finding cliques in $G'(H)$ corresponds to searching and testing in the powerset of $V$. Given the fact that every subset of a clique is also a clique, this downward-closure property can make efficient clique discovery algorithm possible in a way similar to the Apriori algorithm for finding frequent itemsets — with a "bottom up" manner, the candidate generation step extends valid $k-1$ length itemsets one item at a time, and groups of candidates are tested against $G'(H)$ to determine if they form cliques. The algorithm terminates when no further successful extensions are found.

#### 5.2.2.5. Connected Components of $G'(H)$

Complete subgraph (i.e., clique) is a very strong requirement that can limit the approach to restricted cases of semantically associated itemsets. One way to relax this requirement is to find connected components of $G'(H)$, which can be viewed as a closure under semantic association. The number of connected components equals the multiplicity of 0 as an eigenvalue of the Laplacian matrix of $G'(H)$. Although the set of connected components is not downward closed, there is efficient way to find all connected components of a graph in linear time using either breadth-first search or depth-first search. In either case, a search that begins at some particular vertex will find the entire connected component containing the vertex. When the search returns, loop through other vertices and start a new search whenever the loop reaches a vertex that has not already been included in a previously found connected component.

#### 5.2.2.6. Ranking of Itemsets

Once the semantically associated 2-itemsets and $k$-itemsets are generated, they can be ranked by a quantity indicating the strength of association among items in the set. We tentatively compute this quantity by averaging the total pairwise similarities over the number of subedges of the itemset's corresponding clique or connected component in $G'(H)$.

## 5.3. Incorporation of Ontologies

Graphs are mathematical objects that enjoy wide-spread usage for many tasks, which include the visualization and analysis of data for humans, mathematical reasoning, and the implementation as a data structure for developing data mining algorithms. Besides the common graph-theoretic model of RDF as labeled, directed multi-graphs, Hayes has established that RDF can be also represented as hypergraphs (bipartite graphs) [**?** ]. This result constitutes an important aspect of the theoretical basis of this paper and is discussed in sections below. We propose to use the graph-based representation for RDF as a combined information source of both domain knowledge and data for mining semantic associations.

We distinguish paths in the RDF bipartite graph by assigning weights to those paths that represent different semantic relationships such as class subsumption, part_of, and other general or domain–specific properties.

**Definition 5.3..1.** ***(Data model for the combined RDF bipartite graph)*** The unified RDF bipartite graph of both data and ontology is defined as $G = \langle V_v \cup V_s, E \rangle$, where $V_v$ denotes value nodes corresponding to RDF components (subject, predicate, or object), and $V_s$ denotes statement nodes corresponding to RDF statements. More specifically, statement nodes can be further divided according to whether they are from

data or ontology, i.e., $V_s = V_d \cup V_o$; the value nodes can be divided according to whether they represent rows or attributes in the data, i.e. $V_d = V_r \cup V_a$. The graph $G$ can be represented in a biadjacency matrix $\mathbf{M}$, where $\mathbf{M}(i, j)$ is non-zero if there is an edge between $\langle V_{v_i}, V_{s_j} \rangle$. For an unweighted graph, the value can be 0/1, while for a weighted graph, any non-negative value.

The biadjacency matrix $\mathbf{M}$ can be split into vertical stripes by statement nodes $V_s$. For example, according to Figure 3.6.(B), the bipartite graph corresponding to lower 8 RDF statements representing the underlying transaction table can be modeled as the matrix $\mathbf{M}_d$ in Equation Equation 5.9. (RDF statement nodes are labeled $s_1 \ldots s_8$ respectively); and the bipartite graph corresponding to upper 4 statements (labeled $s_9 \ldots s_{12}$) representing the subsumption hierarchy in the ontology can be modeled as the matrix $\mathbf{M}_o$ in Equation Equation 5.10..

To obtain the biadjacency matrix $\mathbf{M}$ of the combined RDF bipartite graph in Figure 3.6., we can simply concatenate $\mathbf{M}_d$ and $\mathbf{M}_o$ horizontally: $\mathbf{M} = [\mathbf{M}_d \; \mathbf{M}_o]$. In general, If there are $k$ different semantic relationships in the ontology, $\mathbf{M}_o$ can be further divided into more vertical stripes $\mathbf{M}_{o_i}, i = 1 \ldots k$, where $\mathbf{M}_{o_i}$ may represent, for example, the "part_of" lattice. Each $\mathbf{M}_{o_i}$ is distinguished from another by the respective weight. In this case, $\mathbf{M}$ is the horizontal concatenation of all the weighted vertical stripes as shown in Equation Equation 5.8.. After the concatenation, $\mathbf{M}$ can be represented as the form shown in Equation Equation 5.11..

$$\mathbf{M} = \begin{bmatrix} w_d \mathbf{M}_d & w_{o_1} \mathbf{M}_{o_1} & w_{o_2} \mathbf{M}_{o_2} & \ldots \end{bmatrix} \qquad \text{(Equation 5.8.)}$$

$$\mathbf{M}_d = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{cccccc} s_1 & s_2 & s_3 & \dots & & s_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & & & 0 \\ 0 & 0 & 1 & & & 0 \\ 0 & 0 & 0 & \dots & & 0 \\ 0 & 0 & 0 & & & 1 \\ 1 & 0 & 1 & & & 0 \\ 0 & 1 & 0 & & & 0 \\ 0 & 0 & 0 & \dots & & 0 \\ 0 & 0 & 0 & & & 1 \\ 0 & 0 & 0 & & & 0 \end{bmatrix} \end{array} \qquad \text{(Equation 5.9.)}$$

$$\mathbf{M}_o = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{bmatrix} s_9 & s_{10} & s_{11} & s_{12} \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \qquad \text{(Equation 5.10.)}$$

$$\mathbf{M} = \begin{array}{c} \\ r \\ a \end{array} \begin{array}{c} \begin{array}{cccc} ds & os_1 & os_2 & \dots \end{array} \\ \left[\begin{array}{c|c|c|c} \mathbf{M}_{dr} & \mathbf{0} & \mathbf{0} & \dots \\ \hline \mathbf{M}_{da} & \mathbf{O}_1 & \mathbf{O}_2 & \dots \end{array}\right] \end{array} \quad \text{(Equation 5.11.)}$$

By developing the unified representation for both data and domain knowledge, and utilizing ontology annotations, such as our results [**?** ], we can produce one RDF hypergraph, which serves as the basis for perform semantic data mining in a systematic way. Given this, the main research challenge is how to utilize the data and ontology together for semantic data mining. In this paper, we focus on one fundamental data mining tasks, namely, the *association mining*. With additional information from ontology (domain knowledge), the unified RDF hypergraphs will enable us to discover hidden association between entities, between entities and ontological concepts, and between ontological concepts. Intuitively, these associations are defined in terms of the paths linking the nodes and the node labels should be taken into consideration as they represent different semantics.

### 5.3.1. Similarity Ranking by Random Walk with Restart

Similar to the relevance score [**?** ], we believe that two items have a strong semantic association if they are related to many similar objects. We denote the similarity score between entities $e_1$ and $e_2$ by $s(e_1, e_2)$, where $s(e_1, e_2) \in [0, 1]$ and $s(e_1, e_2) = 1$ if $e_1 = e_2$. Now the problem of ranking semantic associations in the unified graph can be described as follows:

Given an attribute node $a$ in the unified graph $G = G_d \cup G_o$ and $a \in G_d \cap G_o$ we want to compute a similarity score $s(a, b)$ for all nodes $b(\neq a) \in G_d \cap G_o$. The result is a one-column vector containing all similarity scores of the entities with respect to $a$ [**?** ].

The motivation is to apply random walks with restart (RWR) from the given node $a$, and use the steady-state probability of each node at convergence as the similarity measure, i.e., the similarity score of node $b$ is defined as the probability of visiting $b$ via a random walk which starts from $a$ and goes back to $a$ with a probability $c$. In more detail, RWR in a bipartite graph works as follows: assume we have a random walker that starts from node $a$. For each step, the walker chooses randomly among the available edges from the current node it stays. After each iteration, with probability $c$, it resets its position back to node $a$. The final steady-state probability that the random walker reach node $b$ is the similarity score of L with respect to $a$: $s(a, b)$. We choose the random walk approach to compute the relevance score because it gives node $b$ high ranking if $b$ and $a$ are connected by many nodes; this is because the random walker has more paths to reach $b$ from $a$. The purpose of the periodic restart of the random walk is to raise the chance that close related nodes are visited more often than other nodes.

In the following, we first propose an algorithm for random walk-based similarity ranking on a unified RDF bipartite graph. The algorithm can be used in such situations as, for example, if users are interested in products that are usually bought together in the same transactions by different customers, or common side effects of the same drugs prescribed to different patients, etc.

Given the biadjacency matrix $\mathbf{M}$ in Equation Equation 5.8. for the combined RDF bipartite graph $G$, we can construct the adjacency matrix $\mathbf{A}$ of $G$ as following:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{0} \end{bmatrix}$$

The probability of a random walker taking a particular edge $\langle a, b \rangle$ from a node $a$ while traversing the graph is proportional to the edge weight over the total weight of all

outgoing edges from $a$, i.e., $P(a,b) = A(a,b)/\Sigma_{i=1}^{m+n} A(a,i)$. Therefore, the Markov transition matrix $P$ of $G$ is constructed as: $P = normc(A)$, where $normc(A)$ normalizes $A$ such that every column sum up to 1.

First, we transform the input attribute node $a$ into a $(k+n) \times 1$ query vector $\mathbf{q}_a$ with 1 in the $a$-th row and 0 otherwise. Second, we need to compute the $(k+n) \times 1$ stead-state probability vector $\mathbf{u}_a$ over all nodes in $G$. Last we extract the probabilities of the row nodes as the similarity score vectors. Note that $\mathbf{u}_a$ can be computed by an iterated method from the following lemma.

**Lemma 5.3..1.** Let $c$ be the probability of restarting random-walk from the node $a$. Then the steady-state probability vector $\mathbf{u}_a$ satisfies

$$\mathbf{u}_a = (1-c)P_A\mathbf{u}_a + c\mathbf{q}_a \ . \qquad \text{(Equation 5.12.)}$$

---

**Algorithm 2** Calculate Semantic Association

---

**Input:** query attribute $a$, bipartite matrix $M$, restarting probability $c$, tolerant threshold $\epsilon$

**Output:** $y = x^n$

$\mathbf{q}_a \Leftarrow \mathbf{0}$

$\mathbf{q}_a(a) = 1$ (set $a$-th element of $\mathbf{q}_a$ to 1)

**while** $|\Delta \mathbf{u}_a| > \epsilon$ **do**

$$\mathbf{u}_a = (1-c) \begin{bmatrix} normc(\mathbf{M})\mathbf{u}_a(k+1:k+n); \\ normc(\mathbf{M}^T)\mathbf{u}_a(1:k) \end{bmatrix} + c\mathbf{q}_a$$

**end while**

**return** $\mathbf{u}_a(1:k)$

---

The iterative update of $\mathbf{u}_a$ in the algorithm (inside the while loop) is modified from Lemma 5.3..1 while avoiding materializing $\mathbf{A}$ and $\mathbf{P}$ for scalability.

# CHAPTER VI

# CASE STUDY

## 6.1. Ontology Matching

## 6.2. Experiment

Because we are interested in understanding the property of the Pareto front obtained by our method, we conducted a series of experiments to highlight tradeoffs of the objectives functions. First, to illustrate the proposed method is indeed capable of determining matching between numeric-typed attributes and clusters, we synthesized a dataset simulating some extreme conditions under which previous methods are ineffective. Also, from the results obtained on the synthetic dataset, we empirically study tradeoffs between the two objective functions. Then, to evaluate the scalability of the method, we carry out a series of tests on a set of data with varied sizes. Finally, encouraged by these results, we applied our methods to actual neuroscience ERP (event-related potentials) data to highlight the applicability of our method to the neuroscience domain.

### 6.2.1. Synthetic Dataset

#### 6.2.1.1. Data Generation:

In the synthetic dataset, we generated values for each attribute in such a way that each attribute can be divided into several clusters, and each cluster corresponds to a Gaussian distribution with different mean and standard deviation, but the overall mean and standard deviation of values from all clusters in one attribute are made very close to those in other attributes. For example, Figure 6.1. illustrates the value distributions of three attributes ($a_1, a_2$, and $a_3$) from one dataset and their corresponding counterparts

($a'_1, a'_2$, and $a'_3$) from another. It shows that the overall means and standard deviations for these six attributes are almost indistinguishable, and their ranges are similar as well. Previous methods using these whole-attribute-wise quantities as statistical characterization of attributes would have a hard time determining the matchings. However, as mentioned above and illustrated in the figure, the individual distributions underlying clusters in these attributes are distinct and, by using the segmented statistical characterization of attributes, the difference is significant enough to differentiate and identify matchings between attributes.

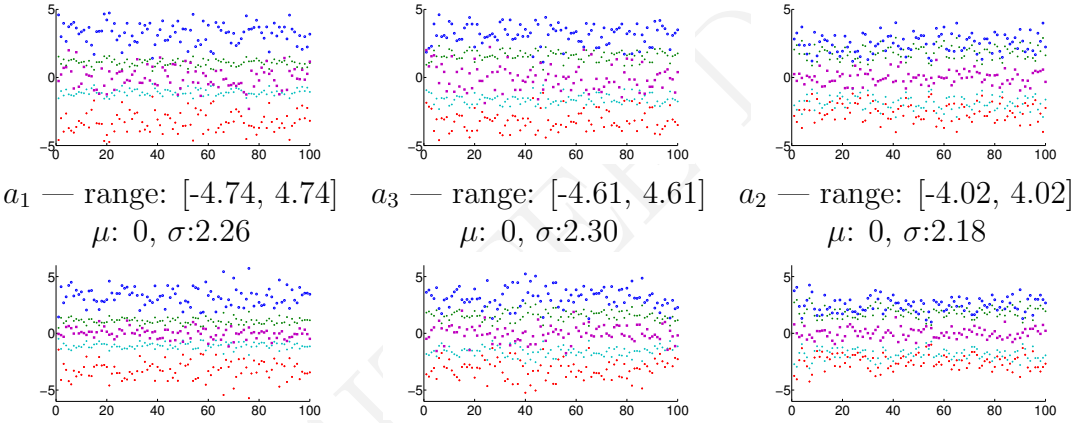



FIGURE 6.1. Scatter plots of data instances from three sample attributes in one synthetic dataset (upper frame) and those of their corresponding attributes from another (lower frame) are illustrated to show their respective value distributions.

#### 6.2.1.2. Results:

Figure 6.2. illustrates the Pareto front obtained from matching two synthetic datasets, each having 20 attributes and 5 clusters. Most notably, the gold standard results for both attribute matching and cluster matching are obtained from the left-most point on the Pareto front. In other words, given the decision variables ($X$) corresponding to

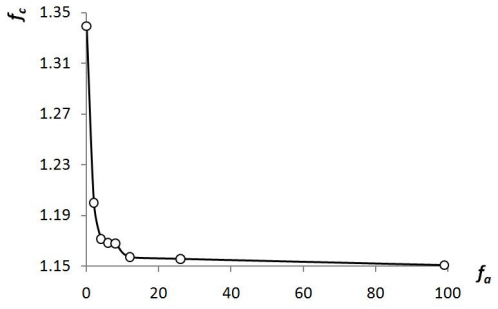



FIGURE 6.2. An example Pareto front obtained from matching two synthetic datasets with 20 attributes and 5 clusters.

that point, we obtained 100% correct matching results. We further observed that in our subsequent tests on other synthetic datasets with varied number of attributes and clusters, the derived Pareto fronts all contain gold standard result, and the point corresponding to the gold standard can always be found towards the minimum end of $f_a$. Given this, we propose the following method to reduce the Pareto-optimal set to a single point corresponding to the most favored choice ($X^*$) in the decision space. The idea is to find the decision with the minimum weighted sum of objective values in the obtained Pareto-optimal set, i.e., $X^* = \arg\min_X \left[\alpha f_a(X) + \beta f_c(X)\right]$, where $\alpha$ and $\beta$ are weights. We first conducted preliminary experiments to determine the best values for $\alpha$ and $\beta$ (0.8 and 0.2 respectively) and used them in all subsequent experiments. This method works markedly well on the synthetic datasets. For all the tests described in Table 6.1., 100% correct results for both attribute and cluster matchings are obtained (hence we omit the precision in the table).

#### 6.2.1.3. Running Time:

We systematically altered the number of attributes and clusters present in the data and conducted a series of tests to show the scalability of the proposed method.

The running time under different configurations is reported in Table 6.1.. The time is calculated by averaging over 5 runs of each test (on a 2.53GHz dual-core CPU with 4 gigabytes memory), each run having 1000 iterations in the simulated annealing process. The main computationally expensive part of the annealing process is the generation of new candidate solution phase (function $G$) in which an assignment problem is solved using the Hungarian method. The complexity of the Hungarian method is cubic and is already the most efficient algorithm for solving the assignment problem (a brute force algorithm has a factorial complexity). Fortunately, rarely is the case that the number of attributes or clusters is large in real-world scenarios where the proposed technique is needed. For reasonable configurations in most practical applications, the computation time is within a tractable range as shown in table 6.1..

| # attributes | # clusters | time (sec) |
|---:|---:|---:|
| 5 | 20 | 0.28 |
| 20 | 20 | 1.81 |
| 20 | 40 | 7.04 |
| 20 | 60 | 17.80 |
| 40 | 20 | 4.66 |
| 40 | 40 | 11.74 |
| 40 | 60 | 25.93 |
| 60 | 20 | 10.95 |
| 60 | 40 | 20.70 |
| 60 | 60 | 37.35 |
| 100 | 100 | 172.23 |

TABLE 6.1. Running time of the annealing process on synthetic datasets with varied configurations of attribute and cluster sizes. The time is obtained by averaging over results of 5 runs of each test.

### 6.2.2. Neuroscience Dataset

#### 6.2.2.1. Data Acquisition:

To address the problems of attribute and cluster matching in a real-world neuroscience application, we used a set of realistic simulated ERP (event-related potentials) datasets, which were designed to support evaluation of ERP analysis methods [**?** ]. The datasets were specifically designed to simulate heterogeneous data from different groups of subjects under different conditions (via distinct simulated brain activities), as well as distinct measurement methods (spatial and temporal metrics) and distinct patterns (reflecting two different pattern decomposition techniques). Real ERP data arise from superposition of latent scalp-surface electrophysiological patterns, each reflecting the activity of a distinct cortical network that cannot be reconstructed from the scalp-measured data with any certainty. Thus, real ERP data are not appropriate for evaluation of ERP pattern mapping. By contrast, simulated ERP data are derived from known source patterns and therefore provide the necessary gold standard for evaluation of our proposed methods.

The raw data for this study consist of 80 simulated event-related potentials (ERPs), in which each ERP comprises simulated measurement data for a particular subject ($n = 40$). The 40 simulated subjects are randomly divided into two 20-subject groups, SG1 and SG2, each containing 40 ERPs (20 subjects in 2 experimental conditions). Each ERP consists of a superposition of 5 latent varying spatiotemporal patterns. These patterns were extracted from the two datasets, SG1 and SG2, using two techniques: temporal Principal Components Analysis (tPCA) and spatial Independent Components Analysis (sICA), two data decomposition techniques widely used in ERP research [**?** ]. To quantify the spatiotemporal characteristics of the extracted patterns, two alternative metric sets,

m1 and m2, were applied to the two tPCA and the two sICA derived datasets. For a complete explanation of these alternative metrics, please see Appendix in [**?** ].

In summary, the simulated ERP data generation process yielded eight test datasets in total, reflecting a 2 (attribute sets) $\times$ 2 (subject groups) $\times$ 2 (decomposition methods) factorial design. Therefore, for each attribute sets there are 4 datasets generated from different combinations of subject groups and decomposition methods, resulting $4 \times 4 = 16$ cases for the studies of attribute matching and cluster matching. The reason to include such variabilities was to test the robustness of our matching method to different sources of heterogeneity across the different datasets. Within all test datasets, 5 major ERP spatiotemporal patterns are present. They are P100, N100, N3, MFN, and P300. These patterns can be identified in the datasets by clustering analysis. Pretending that the latent patterns underlying discovered clusters are unknown, we hope to match clusters across datasets to recover the fact that the same patterns are present in all datasets.

#### 6.2.2.2. Results:

We applied the weighted sum method as the post-process step after obtaining the Pareto-optimal solutions to determine the most favored choice using the parameters ($\alpha$ and $\beta$) discovered in the preliminary experiments on synthetic datasets (cf. Section 6.2.1.). The accuracy of attribute matching and cluster matching along with the number of points in the Pareto front are listed in Table 6.2. (all these results are obtained by taking average from 5 runs for each test case).

It can be observed from the results in Table 6.2. that more different factors involved in the acquisition of the two datasets for matching can negatively affect the matching performance. For example, in test case 1, the two datasets are drawn from the same subject group (SG1) and preprocessed using the same decomposition method (sICA);

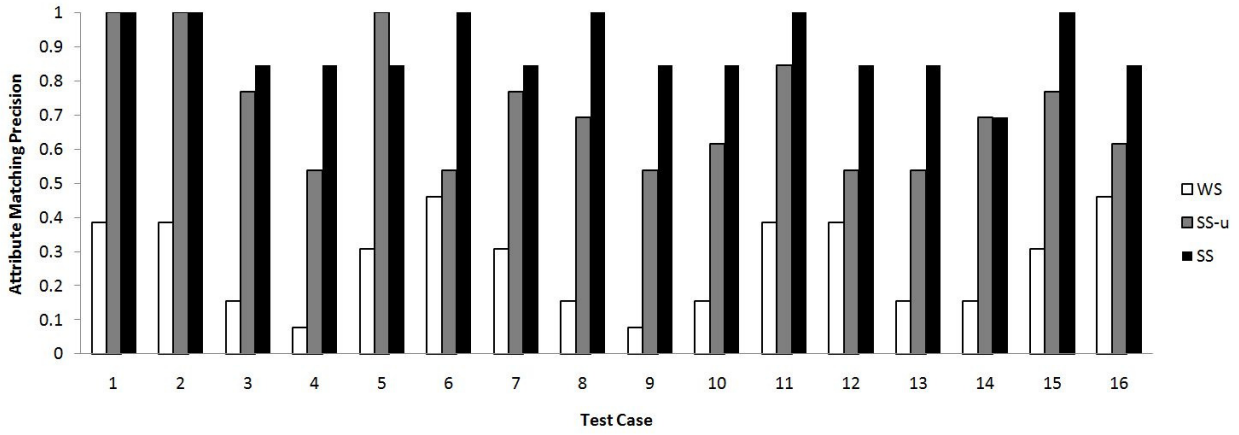



FIGURE 6.3. A comparison of the attribute matching accuracy of three methods on the 16 test cases of the neuroscience dataset. The three methods being compared are matching based on whole-attribute statistics (WS), segmented attribute statistics without knowing a priori cluster matching (SS-u), and segmented attribute statistics with expert-aligned clusterings (SS).

whereas in test case 4, the subject groups and decomposition methods are all different, resulting in greater variability and hence the performance is less satisfactory. However, it is worth noting that our method greatly outperforms traditional whole-attribute-based statistic characterization, as is shown in Figure 6.3.. In this figure we also demonstrate the accuracy of the segmented statistics characterization with expert-labeled patterns, meaning that the data is partitioned and aligned in the most accurate way, which marks the best achievable attribute matching performance. But it is not feasible because, as mentioned in Section I, manually recognizing patterns (partitioning data) and aligning them across datasets requires a priori knowledge of attributes in the datasets which is exactly what the problem of attribute matching tries to discover (the circular causality problem). On the other hand, our method does not require human involvement (except the specification of the number of clusters (patterns) present in the data in order to run the clustering analysis) in determining both the attribute matching and cluster matching and is able to achieve close-to-optimal results.

| Test case | Source params | Target params | $P_a$ | $P_c$ | $\lvert\Sigma\rvert$ |
|---|---|---|---|---|---|
| 1 | ⟨ SG1, sICA, m1 ⟩ | ⟨ SG1, sICA, m2 ⟩ | 13/13 | 5/5 | 5 |
| 2 | ⟨ SG1, sICA, m1 ⟩ | ⟨ SG2, sICA, m2 ⟩ | 13/13 | 5/5 | 6 |
| 3 | ⟨ SG1, sICA, m1 ⟩ | ⟨ SG1, tPCA, m2 ⟩ | 10/13 | 5/5 | 6 |
| 4 | ⟨ SG1, sICA, m1 ⟩ | ⟨ SG2, tPCA, m2 ⟩ | 7/13 | 3/5 | 8 |
| 5 | ⟨ SG2, sICA, m1 ⟩ | ⟨ SG1, sICA, m2 ⟩ | 11/13 | 3/5 | 7 |
| 6 | ⟨ SG2, sICA, m1 ⟩ | ⟨ SG2, sICA, m2 ⟩ | 13/13 | 5/5 | 7 |
| 7 | ⟨ SG2, sICA, m1 ⟩ | ⟨ SG1, tPCA, m2 ⟩ | 10/13 | 5/5 | 6 |
| 8 | ⟨ SG2, sICA, m1 ⟩ | ⟨ SG2, tPCA, m2 ⟩ | 9/13 | 2/5 | 8 |
| 9 | ⟨ SG1, tPCA, m1 ⟩ | ⟨ SG1, sICA, m2 ⟩ | 7/13 | 5/5 | 4 |
| 10 | ⟨ SG1, tPCA, m1 ⟩ | ⟨ SG2, sICA, m2 ⟩ | 8/13 | 5/5 | 6 |
| 11 | ⟨ SG1, tPCA, m1 ⟩ | ⟨ SG1, tPCA, m2 ⟩ | 11/13 | 5/5 | 6 |
| 12 | ⟨ SG1, tPCA, m1 ⟩ | ⟨ SG2, tPCA, m2 ⟩ | 7/13 | 3/5 | 5 |
| 13 | ⟨ SG2, tPCA, m1 ⟩ | ⟨ SG1, sICA, m2 ⟩ | 7/13 | 3/5 | 5 |
| 14 | ⟨ SG2, tPCA, m1 ⟩ | ⟨ SG2, sICA, m2 ⟩ | 9/13 | 5/5 | 6 |
| 15 | ⟨ SG2, tPCA, m1 ⟩ | ⟨ SG1, tPCA, m2 ⟩ | 10/13 | 3/5 | 8 |
| 16 | ⟨ SG2, tPCA, m1 ⟩ | ⟨ SG2, tPCA, m2 ⟩ | 8/13 | 3/5 | 8 |

TABLE 6.2. Matching performance of the proposed method with MOSA on the 16 test cases from the neuroscience dataset. The source and target parameter configuration of the data acquisition process of each test case are shown. $P_a$ and $P_c$ denote the accuracy of attribute matching and cluster matching respectively. $\Sigma$ is the number of points in the obtained Pareto-front. The quantities listed in the table are obtained by averaging over 5 runs of each test.

### 6.2.3. Comparison with Multi-Objective Genetic Algorithm

The concept of Genetic Algorithm (GA) was developed by Holland and his colleagues [**?** ]. GA is first inspired by the evolutionary process in which weak and unfit species within their environment are faced with extinction and stronger ones have greater opportunities to pass their genes to next generation. Comparing to Simulated Annealing, Genetic Algorithm often offers a different perspective in the field of numerical optimization. Starts from a number of random generated population, cross over and evolve; GA has the ability to search in parallel around different and often fully scattered instances in the solution space, in contrast to the "single thread" search in Simulated

Annealing. In this paper, we also implemented the Multi-Objective Genetic Algorithm as the metaheuristics to solve the dual matching problem.

To compare the performance between GA and SA, we first carry out an experiment on the same set of Neuroscience Data, as shown in Table 6.3.. The iteration parameters of both algorithm are tuned so that the convergence time are about the same. The performance are then compared under such setting. We manually examine the Pareto front derived in each test case and find the solution that is closest to the gold standard and the accuracy of which is reported in Table 6.3. (averaged over 5 independent runs).

| Test Case | $P_a$ (%) | $P_c$ (%) | $\Sigma$ |
|---|---|---|---|
| 1 | 100 | 100 | 9 |
| 2 | 98.2 | 96.6 | 10 |
| 3 | 53.4 | 98.0 | 9 |
| 4 | 53.3 | 98.0 | 11 |
| 5 | 100 | 98.2 | 5 |
| 6 | 71.2 | 96.0 | 6 |
| 7 | 59.4 | 94.4 | 6 |
| 8 | 59.7 | 98.8 | 6 |
| 9 | 25.2 | 100.0 | 6 |
| 10 | 38.5 | 100.0 | 5 |
| 11 | 77.7 | 99.2 | 7 |
| 12 | 69.2 | 100.0 | 9 |
| 13 | 38.7 | 100.0 | 9 |
| 14 | 40.3 | 98.8 | 11 |
| 15 | 45.0 | 96.0 | 8 |
| 16 | 84.6 | 98.8 | 16 |

TABLE 6.3. Matching performance of the proposed method with MOSA on the 16 test cases from the neuroscience dataset. The source and target parameter configuration of each test case is the same as in Table 6.2..

The number of population kept in each generation is an important parameter regarding the complexity and performance in MOGA. Intuitively, the more instances we keep, the broader the search space we can explore in each generation. Table 6.3. shows the result with the number of population set to 4. We have also tested other settings and

found out that the accuracy in most cases increase with the number of population but in rare cases the performance deteriorates. The overall performance of MOGA is comparable to that of MOSA but appears to be less robust. It is worth noting that the metaheuristics (MOSA and MOGA) we employed in the experiments are simple algorithms. More modern and sophisticated methods that explore various fitness assignment procedure, elitisim, or diversification approaches will be very likely to improve the performance.

| | | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxid |
|---|---|---|---|---|---|---|---|
| mean | data1 | 6.86 | 0.28 | 0.34 | 6.35 | 0.05 | 35.58 |
| | data2 | 6.85 | 0.28 | 0.33 | 6.43 | 0.05 | 35.02 |
| stdev | data1 | 0.84 | 0.1 | 0.12 | 4.98 | 0.02 | 16.4 |
| | data2 | 0.86 | 0.1 | 0.12 | 5.16 | 0.02 | 17.61 |

| | | total sulfur dioxide | density | pH | sulphates | alcohol | quality |
|---|---|---|---|---|---|---|---|
| mean | data1 | 138.98 | 0.99 | 3.19 | 0.49 | 10.53 | 5.88 |
| | data2 | 137.68 | 0.99 | 3.19 | 0.49 | 10.49 | 5.88 |
| stdev | data1 | 41.86 | 0.02 | 0.16 | 0.11 | 1.25 | 0.89 |
| | data2 | 43.18 | 0 | 0.15 | 0.12 | 1.22 | 0.89 |

TABLE 6.4. Summary of the statistical characteristics of attributes in the Wine Quality dataset.

#### 6.2.3.1. Wine Quality Dataset

In order to further validate our method, we implement our method also on a real-world wine quality dataset [**?** ] which is available through the UCI machine learning repository[1]. This dataset has 12 attributes and 4898 records. We apply uniform sampling to split it into two equal-sized subsets. The attributes are anonymized and randomly reordered in each subset to generate artificial heterogeneity.

We then apply the proposed method with MOSA and MOGA as metaheuristics respectively. The test is focused on attribute matching because the gold standard is

[1] `http://archive.ics.uci.edu/ml/datasets/Wine+Quality`

known while the gold standard of cluster matching is unknown. Table 6.4. summarizes the statistics for each attributes in the dataset. For both MOSA and MOGA derived Pareto optimal solutions, we manually select the one that is closest to the gold-standard matching (e.g., the solution with 10 out 12 attributes matched correctly). Each metaheuristics is invoked 5 times and the matching accuracy is averaged over these runs. The performance for attribute matching is shown in Table 6.5.. The result demonstrate a markedly high accuracy for both MOSA and MOGA. It is worth noting that in most runs the Pareto fronts derived from MOSA and MOGA contain the gold standard matching (hence the high accuracy). It suggests a strategy to reduce the Pareto front in the matching problem by running MOSA or MOGA repeatedly after some times and only those "stable" points that appear more than certain proportion of the times are considered to be presented to decision makers.

| MOSA accuracy (%) | MOSA running time (ms) | MOGA accuracy (%) | MOGA running time (ms) |
|---|---|---|---|
| 95.5 | 517 | 92.3 | 3356 |

TABLE 6.5. Performance of the proposed method with MOSA and MOGA as metaheuristics respectively on the Wine Quality dataset.

### 6.3. Mining Semantically Associated Itemset

Because we are interested in understanding the differences between the $s_{CT}$ and $s_{L+}$ similarity measures for generating semantically associated itemsets, we conducted a series of experiments to highlight their tradeoffs. First, to illustrate the power of hypergraphs in finding associations via linking items, we synthesized a dataset for the *fish oil* example. Next, to illustrate the tradeoffs between the two methods, we evaluated both methods against a commonly used *shopping cart* dataset. Finally, encouraged by these results, we

applied these methods to actual *electronic health records* to highlight their scalability and applicability to the medical domain.

### 6.3.1. Fish Oil

#### 6.3.1.1. Dataset

As mentioned in Section **??**, *fish oil* and *Raynaud's syndrome* have been shown by Swanson [**?** ] to be linked together indirectly via various *blood changes*. He found these associations from examining biomedical texts. As a proof of concept, we replicated this situation by synthesizing a table of 50 rows, which is about the same scale as in Swanson's experiment. Each row represents a set of terms generated to represent biomedical text. Each set of terms was specifically generated so that *fish oil* and *Raynaud's syndrome* never appear together. The column headers include *fish oil, blood changes*, *Raynaud's syndrome*. Six other random variables acted as noise. We then applied the $s_{CT}$, $s_{L+}$ to the dataset. Specifically, we set a threshold for first generating top-15 2-itemsets using either similarity measure. Based on the generated 2-itemsets we used clique search to generate ($k > 2$)-itemsets.

#### 6.3.1.2. Results

The hypergraph approach finds significant links between *fish oil* and *Raynaud's syndrome*, as demonstrated particularly well by the $s_{CT}$ method as shown in Table 6.6.. Even the triplet was discovered by the clique search technique. Most notably, because their co-occurrence is zero, the association would never be discovered by traditional frequent itemset techniques such as the Apriori algorithm [**?** ].

The $s_{L+}$ method also picks-up the association, but it was fairly weak: the association is ranked 23rd among all 2-itemsets (column 3 in Table 6.6. lists the ranking of the $s_{CT}$

| $s_{CT}$ | $s_{L+}$ rank | Freq | Itemset |
|---|---|---|---|
| 0.83 | 2 | 25 | ⟨ *blood_change, fish_oil* ⟩ |
| 0.83 | 1 | 25 | ⟨ *blood_change, Raynaud_synd* ⟩ |
| **0.79** | – | **0** | ⟨ ***blood_change, fish_oil, Raynaud_synd*** ⟩ |
| 0.76 | – | 10 | ⟨ *blood_change, fish_oil, f* ⟩ |
| 0.76 | 7 | 16 | ⟨ *blood_change, f* ⟩ |
| 0.76 | 6 | 16 | ⟨ *blood_change, d* ⟩ |
| 0.76 | 3 | 16 | ⟨ *blood_change, b* ⟩ |
| 0.75 | 9 | 15 | ⟨ *blood_change, a* ⟩ |
| 0.75 | 4 | 15 | ⟨ *blood_change, e* ⟩ |
| 0.73 | 10 | 14 | ⟨ *blood_change, c* ⟩ |
| **0.72** | **23** | **0** | ⟨ ***fish_oil, Raynaud_synd*** ⟩ |
| 0.70 | 10 | 10 | ⟨ *fish_oil, f* ⟩ |
| 0.70 | – | 10 | ⟨ *fish_oil, d* ⟩ |
| 0.70 | 9 | 9 | ⟨ *fish_oil, b* ⟩ |
| 0.68 | 20 | 6 | ⟨ *Raynaud_synd, f* ⟩ |

TABLE 6.6. Top semantically associated itemsets generated by $s_{CT}$ from the Synthetic Fish Oil dataset.

results given by the $s_{L+}$). However, as our next evaluations suggest, the $s_{L+}$ demonstrates other favorable qualities.

### 6.3.2. Shopping Cart

#### 6.3.2.1. Dataset

To better understand how the $s_{CT}$ method compares against the $s_{L+}$ method, we tested them on a business shopping cart dataset. This dataset contains purchase information on 100 grocery items (represented by boolean column headers) for 2,127 shopping orders (corresponding to tuples). We applied $s_{L+}$ and $s_{CT}$ and set a threshold to include top-100 2-itemsets, based on which we subsequently used clique search to generate ($k > 2$) itemsets. The top-10 2-itemset results and ($k > 2$)-itemsets corresponding to maximum cliques generated by $s_{CT}$ and $s_{+}$ are reported in Table 6.7. and 6.8. respectively.

#### 6.3.2.2. Results

Unlike the experiment on the fish oil dataset, We do not have specific hypothesis to validate in this test. After examining the results from both measures, we can only conclude they make intuitive sense. However, we observe that the difference between the $s_{CT}$ and $s_{L+}$ becomes more significant in this experiment. The $s_{CT}$ tends to include itemsets with high support and the effect of indirect links is less pronounced. On the other hand, $s_{L+}$ promotes items with support values towards the lower end. We also observe one drawback of the $s_{CT}$ that the result is centered around items with large frequencies (i.e., many direct links to other nodes) and hence in a sense limiting the information (most itemsets are about *cheese*, *soup* and *cookie*). By contrast, the $s_{L+}$ produces more diversified itemsets.

Finally we tested our methods on the dataset of electronic health records of real patients. This dataset is different from the above two datasets not only in scale but also in practical importance as described in the following.

### 6.3.3. Electronic Health Records

#### 6.3.3.1. Dataset

In our final evaluation, we analyzed the electronic health records of real patients. Applying methods like the ones we have described to this kind of data is particularly relevant because of recent legislation aimed at increasing the meaningful use of electronic health records. Discovering meaningful semantically associated itemsets among the set of drugs and diseases identified in the patient's clinical note is a critical step toward identifying combinations of drug classes and co-morbidities, or risk-factors and co-morbidities that are common in patients with a certain outcome (for example, those

| | $s_{CT}$ | Freq | Itemset |
|---|---|---|---|
| 2-itemsets | 0.74 | 39 | ⟨ *Cheese, Soup* ⟩ |
| | 0.73 | 32 | ⟨ *Cheese, Dried Fruit* ⟩ |
| | 0.72 | 36 | ⟨ *Dried, Fruit Soup* ⟩ |
| | 0.72 | 38 | ⟨ *Cookies, Soup* ⟩ |
| | 0.71 | 24 | ⟨ *Cheese, Cookies* ⟩ |
| | 0.70 | 30 | ⟨ *Cookies, Dried Fruit* ⟩ |
| | 0.68 | 31 | ⟨ *Cheese, Preserves* ⟩ |
| | 0.67 | 24 | ⟨ *Cheese, Wine* ⟩ |
| | 0.67 | 21 | ⟨ *Preserves, Soup* ⟩ |
| | 0.67 | 28 | ⟨ *Soup, Wine* ⟩ |
| ($k$>2)-itemsets | 0.64 | 0 | ⟨ *Canned Vegetables, Cheese, Cookies, Dried Fruit, Frozen Vegetables, Nuts, Preserves, Soup, Wine* ⟩ |

TABLE 6.7. Top semantically associated itemsets generated by $s_{CT}$ from the shopping cart dataset.

| | $s_{L+}$ | Freq | Itemset |
|---|---|---|---|
| 2-itemsets | 10.17 | 3 | ⟨ *Sardines, Conditioner* ⟩ |
| | 8.17 | 6 | ⟨ *Toothbrushes, Nasal Sprays* ⟩ |
| | 6.70 | 6 | ⟨ *Yogurt, Anchovies* ⟩ |
| | 6.25 | 5 | ⟨ *Sports Magazines, Cottage Cheese* ⟩ |
| | 5.82 | 5 | ⟨ *Tofu, Sour Cream* ⟩ |
| | 5.79 | 3 | ⟨ *Toothbrushes, Acetominifen* ⟩ |
| | 4.77 | 4 | ⟨ *Sauces, Nasal Sprays* ⟩ |
| | 4.46 | 3 | ⟨ *Sports Magazines, Gum* ⟩ |
| | 4.43 | 4 | ⟨ *Sunglasses, Paper Dishes* ⟩ |
| | 4.05 | 5 | ⟨ *Tofu, Canned Fruit* ⟩ |
| ($k$>2)-itemsets | 4.51 | 2 | ⟨ *Canned Fruit, Sour Cream, Tofu* ⟩ |
| | 2.01 | 1 | ⟨ *Batteries, Cereal, Cooking Oil* ⟩ |
| | 1.75 | 5 | ⟨ *Canned Vegetables, Nuts, Waffles* ⟩ |

TABLE 6.8. Top semantically associated itemsets generated by $s_{L+}$ from the shopping cart dataset.

suffering from myocardial infarction), toward building predictive risk models, as well as toward providing probable hypotheses about the possible causes of that outcome.

We obtained the set of drugs and diseases for each patient's clinical note by using a new tool, the *Annotator Workflow*, developed at the National Center for Biomedical Ontology (NCBO). The patient notes are from Stanford Hospital's Clinical Data Warehouse (STRIDE). These records archive over 17-years worth of patient data comprising of 1.6 million patients, 15 million encounters, 25 million coded ICD9 diagnoses, and a combination of pathology, radiology, and transcription reports totaling over 9 million clinical notes (i.e., unstructured text).

From this set of 1.6 million patients, we extracted a cohort of patients that suffered from kidney failure. Out of those records, we applied our algorithms to all previous records in the patient's timeline, looking at just the set of drugs. Therefore, at a very simplistic level, the experiment result shows that semantically associated itemsets in this context could possibly represent sets of drugs that could lead toward kidney failure when used in combination.

#### 6.3.3.2. Results

The cohort dataset described above contains 467791 rows (corresponding to patients' clinical notes) and 10167 columns (corresponding to annotated terms appeared in the notes). With the help of the techniques described in Section 5.2.2.2., we are able to compute $L^+$ in a tractable amount of time (Equation Equation 5.1. and Equation 5.2. are calculated within 4 hours on a Quad-Core AMD Opteron(tm) Processor with 8 gigabyte memory), based on which we can efficiently derive the $s_{L+}$ itemsets. However, the calculation of $s_{CT}$ on this scale is intractable because an exact computation of all pair-wise $s_{CT}$ requires to fill in a $|V| \times |V|$ similarity table. In order to ameliorate the computational

| | Support | |
|---|---|---|
| | Shopping cart | Electronic health |
| $\mathbf{s_{CT}}$ | 0.58 | 0.82 |
| $\mathbf{s_{L+}}$ | 0.32 | 0.06 |

TABLE 6.9. The Kendall-$\tau$ score between rankings of itemsets generated by $s_{CT}$, $s_{L+}$ and support in the two experiments.

| | $\mathbf{s_{CT}}$ | **Freq** | **Itemset** |
|---|---|---|---|
| 2-itemsets | 0.80 | 39204 | ⟨ *Calcium Chloride, Amiloride* ⟩ |
| | 0.77 | 29325 | ⟨ *Calcium Chloride, Aspirin* ⟩ |
| | 0.76 | 28644 | ⟨ *Calcium Chloride, Probenecid* ⟩ |
| | 0.73 | 24805 | ⟨ *Calcium Chloride, Furosemide* ⟩ |
| | 0.72 | 34271 | ⟨ *Calcium Chloride, Calcium* ⟩ |
| | 0.71 | 21481 | ⟨ *Calcium Chloride, Disulfiram* ⟩ |
| | 0.70 | 16814 | ⟨ *Calcium Chloride, Amphetamine* ⟩ |
| | 0.66 | 19850 | ⟨ *Calcium Chloride, Prednisone* ⟩ |
| | 0.65 | 12231 | ⟨ *Aspirin, Amiloride* ⟩ |
| | 0.65 | 12106 | ⟨ *Probenecid, Amiloride* ⟩ |
| ($k$>2)-itemsets | 0.56 | 0 | ⟨ *Calcium Chloride, Disul-firam, Amphetamine, Aceta-minophen, Calcium, Aspirin, Probenecid, Amiloride, Prednisone, Furosemide* ⟩ |

TABLE 6.10. Top semantically associated itemsets generated by $s_{CT}$ from the kidney failure cohort of the electronic health dataset.

cost, we exploit domain knowledge to identify 582 terms of particular interest and then apply both $s_{CT}$ and $s_{L+}$ on the reduced dataset. The results are shown in Table 6.10. and 6.11. respectively, where we list top-10 2-itemsets and all ($k$ >2)-itemsets corresponding to the maximum clique.

It is clear that, continuing the trend shown in the FoodMart analysis, the $s_{CT}$ result becomes increasingly concordant with the support-based method. For illustrating this point of view, we calculate the Kendall-$\tau$ score between the ranking of itemsets generated by $s_{CT}$, $s_{L+}$, and support as shown in Table 6.9.. We observe from the table that as the

| | $s_{L+}$ | Freq | Itemset |
|---|---|---|---|
| 2-itemsets | 0.820 | 354 | ⟨ *sevoflurane, remifentanil* ⟩ |
| | 0.691 | 978 | ⟨ *frovatriptan, almotriptan* ⟩ |
| | 0.633 | 693 | ⟨ *Etomidate, Rocuronium* ⟩ |
| | 0.496 | 234 | ⟨ *Atazanavir, Pyrimethamine* ⟩ |
| | 0.420 | 3004 | ⟨ *ciclesonide, Fluorometholone* ⟩ |
| | 0.377 | 231 | ⟨ *naratriptan, Mefenamic Acid* ⟩ |
| | 0.373 | 1792 | ⟨ *ciclesonide, Vincristine* ⟩ |
| | 0.332 | 92 | ⟨ *Rocuronium, sevoflurane* ⟩ |
| | 0.325 | 1368 | ⟨ *tazarotene, halobetasol propionate* ⟩ |
| | 0.322 | 506 | ⟨ *Buprenorphine, alosetron* ⟩ |
| ($k$>2)-[redacted] itemsets[redacted] | 0.131 | 701 | ⟨ *Ketorolac, Flurbiprofen, Ketorolac, Etodolac, Sulindac, Piroxicam, Ketoprofen* ⟩ |

TABLE 6.11. Top semantically associated itemsets generated by $s_{L+}$ from the kidney failure cohort of the electronic health dataset.

$s_{CT}$ converges to support, the $s_{L+}$ becomes even more distinct from it. The result is that the itemsets discovered by $s_{CT}$ contain mostly general terms that are repeatedly found in the patients' notes. Although the association is reasonable but hardly interesting. On the contrary, the $s_{L+}$ result is not affected by the dimension of data as well as the presence of items with massive support. It identifies itemsets of relatively low support but more closely bonded by indirect links.

To demonstrate the scalability of the method based on the $s_{L+}$, we also conducted the same analysis on the data of the whole cohort after 2010. The data consisted 1 million rows and 10 thousand columns. We were able to produce the $s_{L+}$ based 2-itemsets in 6 hours. The top results are shown in Table 6.12..

The discovered $s_{L+}$ itemsets provide much valuable insights on the possible interrelationship between drugs. Some of them has been studied in the literature. For example, *sevoflurane/remifentanil* can be used for anaesthesia; *frovatriptan* and *almotriptan* are both oral treatment of migraine headache; *Etomidate* and *Rocuronium*

| $s_{L+}$ | Itemset |
|---|---|
| 0.0301 | ⟨ *White faced hornet venom, Yellow hornet venom* ⟩ |
| 0.0195 | ⟨ *Trichloroacetic Acid, Trichloroacetate* ⟩ |
| 0.0108 | ⟨ *Cloxacillin Sodium, benzathine cloxacillin* ⟩ |
| 0.0101 | ⟨ *Methacycline, Methacycline hydrochloride* ⟩ |
| 0.01 | ⟨ *Entamoebiasis, Hepatic, Liver Abscess, Amebic* ⟩ |
| 0.0086 | ⟨ *butenafine, Butenafine hydrochloride* ⟩ |
| 0.0085 | ⟨ *Acetone, Cantharidin* ⟩ |
| 0.0085 | ⟨ *ethyl cellulose, Cantharidin* ⟩ |
| 0.0085 | ⟨ *ethyl cellulose, Acetone* ⟩ |
| 0.0085 | ⟨ *Poloxamer 407, Eucalyptol* ⟩ |

TABLE 6.12. Top semantically associated itemsets generated by $s_{L+}$ from the whole electronic health dataset after 2010. The dataset contains 1 million rows and 10k columns.

can be used for rapid sequence intubation; etc. This area of research is still very new and there are no good gold standards to compare our results against. However, for single-item drugs that lead to kidney failure, SIDER[1] database lists drugs and their side-effects. Most notably, multi-itemsets are difficult to identify, but our methods have found not only *Ketoprofen* but it has also group other drugs like it (see the ($k > 2$)-itemset shown in Table 6.11., all of the items are anti-inflammatories). Our results are a matter of on-going evaluation with medical experts.

## 6.4. Mining Semantic Association with Ontologies

In this section, we evaluate the combined RDF bipartite graph for discovering semantic associations. We conducted a series of experiments to highlight the effect of the incorporation of ontologies in the mining framework, and to explore the impact of different ratios of weights assigned to various kinds of relationships in the graphs. First, to illustrate the power of combined RDF bipartite graph in finding semantic associations

[1] `http://sideeffects.embl.de/se/C0035078/all`

while taking into account seamlessly the ontological information, we evaluated our methods against a commonly used *shopping cart* dataset together with a manually created ontology describing the subsumption hierarchy for grocery items. Then, encouraged by the result, we applied our method to actual *electronic health records* to highlight its scalability and applicability to the medical domain. The sizes of the datasets are specified in Table 6.13..

| | # data stmts | # isa stmts | # other stmts* |
|---|---|---|---|
| Shopping cart | 8,481 | 127 | 0 |
| Electronic health | 10,000,257 | 1,048,604 | 43780 |

TABLE 6.13. Dataset overview ("stmts" stands for RDF statements). *In the electronic healths test, we explore the "may_treat" relationship between drugs and diseases defined in the National Drug File.

### 6.4.1. Shopping Cart

#### 6.4.1.1. Dataset

The shopping cart dataset contains purchase information on 100 grocery items (represented by boolean column headers) for 2,127 shopping orders (corresponding to tuples) from a Foodmart. We first construct an RDF bipartite graph from the dataset by transforming the table to 8481 RDF statements. Then we manually create an ontology to organize the grocery items into a subsumption hierarchy. In this process, we introduce 28 internal nodes (the 100 grocery items appeared in the data are mostly at the leaf level) and thus derive a total of 127 RDF statements. As the size of this dataset is fairly small, the calculation of similarity ranking for a given term is fast. In the following we highlight the effect of incorporation of ontology by comparing results obtained with and without ontologies.

#### 6.4.1.2. Results

In Table 6.14., results of items ranked by the strength of semantic association given a query term "Toothbrush" under various combinations of parameters are demonstrated side-by-side for comparison. First, we observe that without using ontology, performing random walk with restart on the data graph (Table 6.14..A) starting from "toothbrush" yields similar results to those reported in our previous work [**?** ] based on random walk commute time similarity. Items ranked high in this setting where only the data graph is considered are typically either hub nodes (with many edges linking to other items) or frequently co-occur with the query item (many edges connecting them). Second, applying the same similarity ranking method solely on the ontology graph (Table 6.14..B) gives a list of association based on the graph-configuration of the ontological structure (in this case, the rdfs:subClassOf lattice). The items that are considered most similar to the query term "Toothbrush" is its immediate parent class "PersonalHygiene," followed by some most derived classes at the same level of "PersonalHygiene" and then siblings of "Toothbrush" itself. Next, Table 6.14..C–E demonstrate the results of mining on the combined graph with different ratios of weights assigned to ontology edges and data edges respectively. It is obvious that these results can be seen as a mix of the data-only and ontology-only results with various emphasis on the data or ontology. We can observe that when $w_o/w_d = 20$ where $w_o$ is weight of ontology edge (i.e., rdfs:subClassOf) and $w_d$ is the weight of data edge, the ontology and data appear to have equal significance in determining the ranking. In a rough sense, it conforms to the ratio of the size of ontology graph and data graph as well (see Table 6.13.). In reality, the appropriate ratio for the edge weights is not only dependent on the size of graphs but also the graph configuration (depth, average degree, etc). Moreover, specifying the ratio of prior knowledge in ontologies and inductive evidences in data that one wants to employ for discovering new

patterns is a highly empirical process. Multiple pilot trials may need to be carried out for the optimal ratio before it is applied to the real application.

| w/ data only | | w/ onto only | | $w_o = 1$, $w_d = 1$ | | $w_o = 10$, $w_d = 1$ | | $o_w = 20$, $o_d = 1$ | |
|---|---|---|---|---|---|---|---|---|---|
| item | p(%) | item | p(%) | item | p(%) | item | p(%) | item | p(%) |
| Soup | 0.42 | PersonalHygiene | 12.55 | PersonalHygiene | 0.74 | PersonalHygiene | 3.97 | PersonalHygiene | 6.27 |
| Cookies | 0.41 | Snack | 0.86 | Soup | 0.41 | NasalSprays | 0.41 | NasalSprays | 0.5 |
| NasalSprays | 0.38 | Health | 0.64 | Cookies | 0.4 | Soup | 0.34 | Mouthwash | 0.41 |
| Popcorn | 0.32 | Sponges | 0.57 | NasalSprays | 0.37 | Cookies | 0.34 | Shampoo | 0.31 |
| PaperWipes | 0.29 | Soap | 0.57 | Popcorn | 0.31 | Mouthwash | 0.3 | Soup | 0.29 |
| FrozenVegetables | 0.29 | Shampoo | 0.57 | FrozenVegetables | 0.29 | Popcorn | 0.25 | Cookies | 0.29 |
| PersonalHygiene | 0.26 | NasalSprays | 0.57 | PaperWipes | 0.28 | FrozenVegetables | 0.24 | Sponges | 0.28 |
| DriedFruit | 0.25 | Mouthwash | 0.57 | DriedFruit | 0.25 | PaperWipes | 0.23 | Health | 0.27 |
| Milk | 0.25 | Conditioner | 0.57 | Milk | 0.25 | DriedFruit | 0.22 | Conditioner | 0.27 |
| Mouthwash | 0.24 | MealCourse | 0.54 | Mouthwash | 0.23 | Milk | 0.21 | Soap | 0.25 |
| (A) | | (B) | | (C) | | (D) | | (E) | |

TABLE 6.14. Foodmart items ranked by the strength of semantic association (i.e., $p(\%)$, the steady-state probability).

Note that without filtering the ranked semantic associations on the combined graph, the list will include items that never appear in the transactional data. It is because typically the semantic annotation process links table attributes to their most specific matching concept in the ontology which are close to the leaf level. The incorporation of ontology is to aid the mining process, therefore including in the result those internal items (e.g., "PersonalHygiene") that never appear in the data is counterintuitive. To overcome this, we can simply filter out those items exclusive to the ontology. Table 6.15. shows an example where the query term is "soup" and both test settings yield rankings taking value on the same set of items.

### 6.4.2. Electronic Health Records

#### 6.4.2.1. Dataset

In our second evaluation, we analyzed the electronic health records of real patients. The patient clinical note data are from Stanford Hospital's Clinical Data Warehouse (STRIDE). These records archive over 17-years worth of patient data comprising of

| w/ data only | | | | w/ onto only | | | |
|---|---|---|---|---|---|---|---|
| item | p(%) | item | p(%) | item | p(%) | item | p(%) |
| Cheese | 0.38 | Preserves | 0.19 | TVDinner | 0.46 | Sponges | 0.06 |
| Cookies | 0.32 | Juice | 0.17 | Pizza | 0.46 | Soap | 0.06 |
| DriedFruit | 0.32 | Lightbulbs | 0.17 | Pasta | 0.46 | Shampoo | 0.06 |
| Wine | 0.24 | PaperWipes | 0.16 | HotDogs | 0.46 | NasalSprays | 0.06 |
| CannedVegetables | 0.23 | Pizza | 0.16 | Hamburger | 0.46 | Mouthwash | 0.06 |
| FrozenVegetables | 0.23 | Nuts | 0.16 | FrenchFries | 0.46 | Conditioner | 0.06 |
| Cereal | 0.22 | Popcorn | 0.16 | DeliSalads | 0.46 | Ibuprofen | 0.06 |
| Milk | 0.22 | Chips | 0.16 | DeliMeats | 0.46 | ColdRemedies | 0.06 |
| ChocolateCandy | 0.19 | Eggs | 0.16 | Sunglasses | 0.07 | Aspirin | 0.06 |
| Waffles | 0.19 | TVDinner | 0.15 | Toothbrushes | 0.06 | Acetominifen | 0.06 |

TABLE 6.15. Semantically associated items by filtering out those items exclusive to the Foodmart ontology.

1.6 million patients, 15 million encounters, 25 million coded ICD9 diagnoses, and a combination of pathology, radiology, and transcription reports totaling over 9 million clinical notes (i.e., unstructured text). We obtained the set of drugs and diseases for each patient's clinical note by using a new tool, the *Annotator Workflow*, developed at the National Center for Biomedical Ontology (NCBO), which annotates clinical text from electronic health record systems and extracts disease and drug mentions from the electronic health records.

From this set of 1.6 million patients, we extracted a subset whose clinical notes mention the terms of interest (including both drugs and diseases, such as vioxx and myocardial infraction). Out of those annotated records, we vectorize texts and turned them into a huge bag-of-word representation, from which a RDF bipartite graph is constructed (including 10 million RDF statements, see Table 6.13.). we applied our algorithms to all previous records in the patient's timeline, looking at just the set of drugs and their semantically related diseases. Therefore, at a very simplistic level, the experiment result shows that strong semantically associated items in this context could possibly represent sets of drugs that could lead toward certain diseases.

One strength of the Annotator is the highly comprehensive and interlinked lexicon that it uses. It can incorporate the entire NCBO BioPortal ontology library of over 250

ontologies to identify biomedical concepts from text using a dictionary of terms generated from those ontologies. Terms from these ontologies are linked together via mappings. For this study, we specifically configured the workflow to use a subset of those ontologies that are most relevant to clinical domains, including Unified Medical Language System (UMLS) terminologies such as SNOMED-CT, the National Drug File (NDFRT) and RxNORM, as well as ontologies like the Human Disease Ontology. The resulting set of ontologies contains 1 million subsumption statements. To highlight the capability of our method for incorporating multiple types of relationships, we also explore the "may_treat" relationship between drugs and diseases defined in NDFRT, for example, Thiabendazole may_treat Larva Migrans. Since we are interested in learning the interaction between drugs and diseases, may_treat is naturally a better indicator relationship to include while mining semantic associations than the subsumption relationship. Our results below prove this point.

#### 6.4.2.2. Results

Before studying the drug-disease association, we carried out a similar test to that on the shopping cart dataset, in which we focus on studying the drug-drug and disease-disease association. To this purpose, we combine the subsumption hierarchy in the ontology graph with the data graph. Table 6.16. shows the ranked semantic association for the query term "ROFECOXIB" (an active ingredient of some anti-inflammatory drugs) given different configuration of graphs. Without any preprocessing and prior knowledge about how the clinical notes are prescribed, the incorporation of subsumption relationship can be seen as a mean for denoising and enhancement of the data. Given the ratio of the size of the ontology to the size of data, the ontology in this test is more dominant in

determining the ranking than in the shopping cart experiment. One can gradually change the ratio of $w_o$ to $w_d$ to strike a balance and achieve the optimal result.

| w/ data only | | $w_o = 1, w_d = 1$ | | $w_o = 1, w_d = 2$ | |
|---|---|---|---|---|---|
| p(%) | item | p(%) | item | p(%) | item |
| 1.1132 | pseudoisocytidine | 1.4687 | oral form rofecoxib | 1.3535 | oral form rofecoxib |
| 1.1113 | dimetpramid | 1.1638 | rofecoxib 50 mg oral tablet | 1.0705 | rofecoxib 50 mg oral tablet |
| 1.1112 | quinalphos | 1.1638 | rofecoxib 12.5 mg oral tablet | 1.0705 | rofecoxib 12.5 mg oral tablet |
| 1.1103 | retinol o-fatty-acyltransferase | 1.1638 | rofecoxib 25 mg oral tablet | 1.0705 | rofecoxib 25 mg oral tablet |
| 1.1102 | trimethylaminobutyrate | 1.0762 | cyclooxygenase 2 inhibitors | 0.8561 | cyclooxygenase 2 inhibitors |
| 1.1100 | fungiqual | 0.7812 | duplicate concept | 0.7952 | social control policies |
| 1.1090 | chemoembolization, therapeutic | 0.6390 | rofecoxib 2.5 mg/ml oral suspension | 0.7922 | dimetpramid |
| 1.1088 | social control policies | 0.6390 | rofecoxib 5 mg/ml oral suspension | 0.7920 | quinalphos |
| 1.1088 | insulin | 0.5682 | cyclooxygenase inhibitors | 0.7917 | trimethylaminobutyrate |
| 1.1082 | 2-...hexanoic acid | 0.5487 | r [preparations] | 0.7916 | pseudoisocytidine |

TABLE 6.16. Results of Health items ranked by the strength of semantic association.

Table 6.17. illustrates the rankings of three associations (one per row) under different settings. The first element in the pair is the query item, which are all active ingredients of some prescription drugs, and the ranking shown in the table is for the second item, which are diseases. For example, arthritis is ranked as the 527th semantically associated item to ROFECOXIB according to similarity ranking based only on data graph. All these item pairs are actually backed by known relationships, i.e., as a golden standard, we know the semantic associations between them should be strong.

| | w/ data only | | w/ data and "isa" | | w/ data and "may_treat" | |
|---|---|---|---|---|---|---|
| | p(%) | rank | p(%) | rank | p(%) | rank |
| $\langle ROFECOXIB, arthritis \rangle$ | 0.006 | 527 | 0.004 | 632 | 0.51 | 13 |
| $\langle valdecoxib, arthritis \rangle$ | 0.007 | 613 | 0.005 | 695 | 0.63 | 17 |
| $\langle troglitazone, diabetes \rangle$ | 0.006 | 478 | 0.005 | 514 | 0.44 | 11 |

TABLE 6.17. The rankings of three semantic associations in health data under different settings.

We first observe that the ranking based on data graph alone is fairly high already, consider there are approximately 1 million distinct terms in the ontology. However, the results based on the combination of data and subsumption ("isa") graph are worse. It is because the subsumption hierarchy for drugs and diseases are largely separate structures.

While on the one hand, the incorporation of subsumption relationship boosts the association within closely related drugs and diseases, it on the other hand in fact obfuscate the association between drugs and diseases across their respective subsumption hierarchies. On the other hand, the association between these pairs can be exactly captured by the NDFRT "may_treat" relationship (e.g., NDFRT explicitly defines that ROFECOXIB may_treat arthritis). Therefore, when the "may_treat" graph is incorporated into the mining process, the ranking for the association is greatly boosted.

We believe this result demonstrates the power of domain knowledge in data mining. Using the proposed combined RDF bipartite graph for both data and ontology, the incorporation of domain knowledge can be achieved by simply encoding it in the ontology and choose appropriately relevant part of the ontology that best suits the mining task.

# CHAPTER VII

# DISCUSSION AND FUTURE WORK

## 7.1. Graph-based Methods for Clustering and Classification

**Classification**: In classification, the goal is to find the correct labels for the unlabeled instances given a finite set of labeled instances. In Figure 3.7., the leftmost partition corresponds to rows in the underlying relational table. Suppose a subset of vertices are labeled (e.g., $r_1$ is labeled $l_1$ and $r_4$ is labeled $l_2$), then the classification problem can be formulated as to discover correct labels for unlabeled vertices (e.g., $r_1$ and $r_3$). Conceptually, the pair of nodes being "close to" one another shall share the same labels, which are consistent to one basic principle of semi-supervised learning. However, the challenge is how can we define the closeness between two nodes. We can still rely on the paths linking them where the path length becomes an important measure as well as the node labels.

**Clustering:** Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters. It is easy to observe that in Figure 3.7. the clustering of rows of the underlying relational table can be viewed as a neighborhood formation for vertices on the leftmost partition. Basically, for any closely related nodes, we shall group them together. Especially, if we can derive an explicit similarity/distance measure, then classical distance-based clustering algorithms, such as k-means, can be directly applied to cluster the vertices.

### 7.1.1. Clustering Analysis

#### 7.1.1.1. Distance-based Clustering

Given the hypergraph-based technique for mining semantically associated itemsets presented in the last section, we can apply a "turn around" of the perspective in the hypergraph model and extend it to cluster analysis.

| | A | B | C | D |
|---|---|---|---|---|
| $e_1$: | 1 | 1 | 1 | 0 |
| $e_2$: | 0 | 1 | 1 | 1 |
| $e_3$: | 1 | 0 | 1 | 1 |

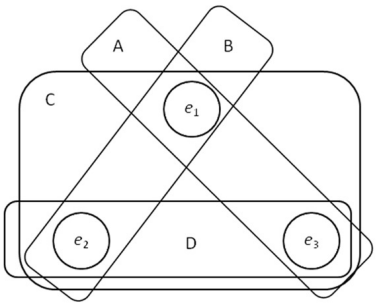



FIGURE 7.1. .

In Figure. 7.1. we show the same motivating example as in Section 5.2.2.. Instead of creating a hyperedge for each row, we create one for each column. A row node is incident on the corresponding hyperedge if it has a value '1' in that column.

#### 7.1.1.2. Partition-based Clustering

Given a set of data points $x_1, \ldots x_n$ and some notion of similarity $s(i, j) > 0$ between all pairs of data points $x_i$ and $x_j$ , the common definition of clustering is to divide the data points into several groups such that points in the same group are similar to each other than ones taken from other groups. The relationships among $n$ data points can be specified by means of an $n \times n$ similarity matrix, whose elements $s_{ij}$ measure the similarity of data points $x_i$ and $x_j$.

Partitional clustering algorithms obtain a single partition of the data that optimizes a certain criterion. The most widely used criterion is minimizing the overall squared distance between each data point and the center of its related cluster.

$$argmax_s \sum_{i=1}^{k} \sum_{x_j \in S} s(j, u_i)^2 \ , \qquad \text{(Equation 7.1.)}$$

where $u_i$ is the mean of points in $S_i$. If we restrict the centroids to members of the data, i.e., $u_i \in S_i$, the resulting clustering algorithm is called $k$-medoids algorithm. The optimization problem in Equation Equation 7.1. itself is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximative method is Lloyd's algorithm [**?** ].

There are several graph-based similarity measures that can be used in the partitional clustering algorithms. The shortest path distance on a graph (also known as the geodesic distance) can be used as the most straight forward measure. The hypergraph commute time similarity introduced in Section 5.2.2. is more appealing to be used for clustering purposes, because as opposed to the shortest path distance, the commute distance between two vertices decreases if there are many different short ways to get from vertex $v_i$ to vertex $v_j$ . So instead of just looking for the one shortest path, the commute distance looks at the set of short paths. Points which are connected by a short path in the graph and lie in the same high-density region of the graph are considered closer to each other than points which are connected by a short path but lie in different high-density regions of the graph.

To illustrate this point of view, the following example illustrates a comparison between Euclidean distance and commute time distance on a graph shown in Figure 5.1..

#### 7.1.1.3. Spectral Clustering

If we do not have more information than similarities between data points, a nice way of representing the data is in form of the similarity graph $G = (V, E)$. Each vertex vi in this graph represents a data point $x_i$. Two vertices are connected if the similarity $s_{ij}$ between the corresponding data points $x_i$ and $x_j$ is positive or larger than a certain threshold, and the edge is weighted by $s_{ij}$. The problem of clustering can now be reformulated using the similarity graph: we want to find a partition of the graph such that the edges between different groups have very low weights (which means that points in different clusters are dissimilar from each other) and the edges within a group have high weights (which means that points within the same cluster are similar to each other). To be able to formalize this intuition we first want to introduce some basic graph notation and briefly discuss the kind of graphs we are going to study.

The $\epsilon$-neighborhood graph: Here we connect all points whose pairwise distances are smaller than $\epsilon$. As the distances between all connected points are roughly of the same scale (at most $\epsilon$), weighting the edges would not incorporate more information about the data to the graph. Hence, the $\epsilon$-neighborhood graph is usually considered as an unweighted graph.

$k$-nearest neighbor graphs: Here the goal is to connect vertex $v_i$ with vertex $v_j$ if $v_j$ is among the $k$-nearest neighbors of $v_i$. However, this definition leads to a directed graph, as the neighborhood relationship is not symmetric. There are two ways of making this graph undirected. The first way is to simply ignore the directions of the edges, that is we connect $v_i$ and $v_j$ with an undirected edge if $v_i$ is among the k-nearest neighbors of $v_j$ or if $v_j$ is among the $k$-nearest neighbors of vi. The resulting graph is what is usually called the k-nearest neighbor graph. The second choice is to connect vertices $v_i$ and $v_j$ if both $v_i$ is among the k-nearest neighbors of $v_j$ and $v_j$ is among the $k$-nearest neighbors

of $v_i$. The resulting graph is called the mutual $k$-nearest neighbor graph. In both cases, after connecting the appropriate vertices we weight the edges by the similarity of their endpoints.

The fully connected graph: Here we simply connect all points with positive similarity with each other, and we weight all edges by $s_{ij}$ . As the graph should represent the local neighborhood relationships, this construction is only useful if the similarity function itself models local neighborhoods. An example for such a similarity function is the Gaussian similarity function $s(x_i, x_j) = e^{-\|x_i - x_j\|^2/(2\sigma^2)}$, where the parameter $\sigma$ controls the width of the neighborhoods. This parameter plays a similar role as the parameter $\epsilon$ in case of the $\epsilon$-neighborhood graph.

### 7.1.2. Classification

The hypergraph representation for the problem of classification is similar to the one for clustering. We can create hyperedges for columns in the same way as shown in Figure 7.1. with the only difference being that a portion of row nodes is labeled and rest unlabeled. The goal for classification is to utilize the information in labeled nodes to come up with correct labels for unlabeled ones.

This problem can be solved by hypergraph-based label propagation. Let $f$ be the objective function (vector) of labels to be learned. Intuitively, there are two criteria for learning the optimal $f$ : (1) we want to assign the same label to vertices that share many incidental hyperedges in common; and (2) assignment of the labels should be similar to the initial labeling $y$. For criteria (1), we define the following cost function

$$\Omega(f, w) = \frac{1}{2} \sum_{e \in E} \sum_{u,v \in e} \frac{w(e)h(u, e)h(v, e)}{d(e)} \ . \qquad \text{(Equation 7.2.)}$$

where $I$ is the identity matrix. Here, $D_v$ and $D_e$ are used for computing the normalization of the hypergraph Laplacian, and the unnormalized hypergraph Laplacian is $diag(HD_ew)HWH^T$ . Empirical results showed that the normalized form gives better classification performance for graphbased learning [**?** ]. If the predicted labels on the vertices are consistent with the incidences with the hyperedges, the value of $\Omega(f, w)$ should be minimized. For criteria (2), we directly calculate the squared-loss between the predicted labeling f and the original labeling y as follows,

$$\sum_{u \in V} (f(u) - y(u))^2 = \|f - y\|^2$$