EXPLOITING NON-REDUNDANT LOCAL PATTERNS AND PROBABILISTIC MODELS FOR ANALYZING STRUCTURED AND SEMI-STRUCTURED DATA

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of The Ohio State University

By

Chao Wang, B.E., M.S.

* * * * *

The Ohio State University

2008

Dissertation Committee:
Dr. Srinivasan Parthasarathy, Adviser
Dr. Joseph Verducci
Dr. Gagan Agrawal
Dr. P. Sadayappan

Approved by

Adviser
Graduate Program in
Computer Science and
Engineering
Dedicated to my beloved wife, Dandan Liu.
ABSTRACT

This work seeks to develop a probabilistic framework for modeling, querying and analyzing large-scale structured and semi-structured data. The framework has three components: (1) Mining non-redundant local patterns from data; (2) Gluing these local patterns together by employing probabilistic models (e.g., Markov random field (MRF), Bayesian network); and (3) Reasoning (making inference) over the data for solving various data analysis tasks. In more detail, our contributions are as follows:

• Mining non-redundant frequent itemset patterns on large transactional data. Often times in many real-world problems frequent pattern mining algorithms yield so many frequent patterns that the end-user is swamped when it comes to interpreting the results. We present an approach of employing probabilistic models to identify non-redundant itemset patterns from a large collection of frequent itemsets on transactional data. We show that our approach can effectively eliminate a large amount of redundancy from a large collection of itemset patterns.

• Employing local probabilistic models to glue non-redundant itemset patterns on large transactional or network data. We propose a technique of employing local probabilistic models to glue non-redundant itemset patterns together in tackling the link prediction task in co-authorship network analysis. The new technique effectively combines topology analysis on network structure data and frequency analysis on network event
log data. The main idea is to consider the co-occurrence probability of two end nodes associated with a candidate link. We propose a method of building MRFs over local data regions to compute this co-occurrence probability. Experimental results demonstrate that the co-occurrence probability inferred from the local probabilistic models is very useful for link prediction.

- Employing global probabilistic models to glue non-redundant itemset patterns on large transactional data. We explore employing global models, models over large data regions, to glue non-redundant itemset patterns together. To this end, we investigate learning approximate global MRFs on large transactional data and propose a divide-and-conquer style modeling approach. Empirical study shows that the models are effective in modeling the data and approximately answering queries on the data.

- Mining non-redundant tree patterns and employing probabilistic approaches to glue them on large XML data. We propose a technique of identifying non-redundant tree patterns from a large collection of structural tree patterns. We show that our approach can effectively eliminate redundancies from a large collection of structural tree patterns. Furthermore, we present techniques of employing these non-redundant tree patterns as summary statistics for the XML data to solve the XML twig selection estimation problem. We propose a probabilistic framework under which the selectivity of a twig query can be estimated from the information of its subtrees. Empirical results demonstrate the efficacy of our approach on real and synthetic datasets.
ACKNOWLEDGMENTS

Foremost, I would like to thank my advisor, Dr. Srinivasan Parthasarathy, for his guidance and assistance throughout my graduate career. His advice and research experience have helped me to start the data mining research smoothly and to shape my research skills in this field. Now, I am grateful that I have chosen this exciting research field for my Ph.D. work.

I would like to thank Dr. Joseph Verducci, Dr. Gagan Agrawal and Dr. P. (Saday) Sadayappan for serving on my committee, for reading through drafts of this dissertation, and providing me with valuable insights and comments. Specifically, I thank Dr. Verducci for making reference letters for me when I apply for summer internships.

My research has been supported in part by grants from the National Science Foundation (CAREER-IIS-0347662) and the Department of Energy (DE-FG02-04ER25611). I am very grateful to these sources of funding. I am also very thankful to the Ohio Supercomputer Center for the great computing power they make available to graduate students. Any opinions, findings, and conclusions or recommendations expressed here are those of the author and, if applicable, his adviser and collaborators, and do not necessarily reflect the views of the National Science Foundation and the Department of Energy.

I am fortunate to have interacted with a number of superb lab members during my study here at the data mining research lab: Amol Ghoting, Keith Marsolo, Matthew Otey, Hui Yang, Sameep Mehta, Sitaram Asur, Duygu Ucar, Gregory Buehrer, Shirish Tatikonda,
Venu Satuluri and many others. I am grateful to many useful discussions with Ruoming Jin who is now at Kent state University. Also, I am grateful to insightful discussions about $L_1$ regularization with Dr. Yoonkyung Lee here at the OSU statistics department.

Last, but not least, I would like to thank my parents, my sister for their supporting me to study abroad; and my wife, Dandan Liu, for the love and strength she is giving to me all the time in my life.
VITA


1998 ................................. B.E. Computer Science, Huazhong University of Science and Technology, Wuhan, China.

2001 ................................. M.E. Computer Science, Beijing University of Aeronautics and Astronautics, Beijing, China.

2006 ................................. M.S. Computer Science, The Ohio State University, Columbus OH, USA

PUBLICATIONS

Research Publications


**FIELDS OF STUDY**

Major Field: Computer and Information Science

Studies in:

Data Mining    Prof. Srinivasan Parthasarathy
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Vita</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiii</td>
</tr>
<tr>
<td>Chapters:</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Challenges in Processing Large Structured and Semi-Structured Data</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Organization</td>
<td>10</td>
</tr>
<tr>
<td>2. Background and Related Work</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Frequent Pattern Mining</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1 Frequent Itemset Pattern</td>
<td>12</td>
</tr>
<tr>
<td>2.1.2 Closed, Maximal and Non-Derivable Itemsets</td>
<td>13</td>
</tr>
<tr>
<td>2.1.3 Frequent Tree Pattern</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Probabilistic Graphical Models</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1 Undirected Graphical Model</td>
<td>17</td>
</tr>
<tr>
<td>2.2.2 Directed Graphical Model</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3 Inference</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Learning Markov Random Fields</td>
<td>22</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>General characteristics of the datasets. $k$ is the number of distinct items, $n$ is the number of records, $m$ is the number of total items and $d = \frac{m}{kn}$ is the density index.</td>
</tr>
<tr>
<td>3.2</td>
<td>Skewed itemset distribution on the Chess dataset when varying error threshold</td>
</tr>
<tr>
<td>3.3</td>
<td>Skewed itemset distribution on the Accidents dataset when varying error threshold</td>
</tr>
<tr>
<td>3.4</td>
<td>Skewed itemset distribution on the Mushroom dataset when varying error threshold</td>
</tr>
<tr>
<td>3.5</td>
<td>Skewed itemset distribution on the Web dataset when varying error threshold</td>
</tr>
<tr>
<td>4.1</td>
<td>Summary of the three datasets that were constructed</td>
</tr>
<tr>
<td>4.2</td>
<td>The number of predictions made by the Co-occurrence probability feature alone that were not made by either aKatz or semantic similarity. $c_{500}$, $k_{500}$ and $s_{500}$ refer respectively to the sets to top 500 predicted links using co-occurrence probability, the set of top 500 predicted links using aKatz and the set of top 500 predicted links using semantic similarity.</td>
</tr>
<tr>
<td>4.3</td>
<td>Link prediction classification results. The $K$ for calculating precision for each dataset is the number of positive instances (i.e. true links) in the test dataset.</td>
</tr>
<tr>
<td>4.4</td>
<td>Classification results when varying central neighborhood set size on the Genetics dataset</td>
</tr>
<tr>
<td>4.5</td>
<td>Timing results on co-occurrence probability feature induction</td>
</tr>
</tbody>
</table>
6.1 Dataset characteristics .................................................. 130
6.2 Workload characteristics (average no. of binding tuples) ......... 130
6.3 Summary construction time (in minutes) ............................... 130
6.4 Number of total and 0-derivable patterns on four datasets ........... 130
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An example of non-derivable itemset</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Iterative scaling algorithm</td>
<td>24</td>
</tr>
<tr>
<td>2.3</td>
<td>An MRF example</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Itemset pattern summarization algorithm</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>Results on the Chess dataset: (a) Restoration error (b) Summary size (c) Summarization time</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>Results on the Accidents dataset: (a) Restoration error (b) Summary size (c) Summarization time</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>Results on the Mushroom dataset: (a) Restoration error (b) Summary size (c) Summarization time</td>
<td>45</td>
</tr>
<tr>
<td>3.5</td>
<td>Results on the Web dataset: (a) Restoration error (b) Summary size (c) Summarization time</td>
<td>48</td>
</tr>
<tr>
<td>3.6</td>
<td>Approximate inference based summarization on the Mushroom dataset: (a) No. of patterns after summarization (b) Restoration error (all patterns)</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Enumerating all simple paths of length $K$ between two nodes</td>
<td>63</td>
</tr>
<tr>
<td>4.2</td>
<td>Selecting the central neighborhood set for two nodes</td>
<td>64</td>
</tr>
<tr>
<td>4.3</td>
<td>Inducing co-occurrence probability feature for a pair of nodes</td>
<td>66</td>
</tr>
<tr>
<td>4.4</td>
<td>An example of local model-based co-occurrence probability feature induction</td>
<td>67</td>
</tr>
</tbody>
</table>
4.5 Split of the datasets into Training and Testing data. T1, T2, . . , T10 represent the 10 time intervals the dataset spans.

4.6 Class conditional distributions for different features on the Genetics dataset: (a)Co-occurrence Probability (b) aKatz (c) Semantic Similarity

4.7 ROC Curve Comparison on the (a)DBLP (b)Biochemistry (c)Genetics dataset

5.1 Accumulative graph weighting scheme

5.2 Augmented variable-clusters

5.3 Learning approximate global MRF algorithm

5.4 Conditional independence

5.5 Varying $k$ ($g = 5$): estimation accuracy

5.6 Varying $k$ ($g = 5$): online time

5.7 Varying $k$ ($g = 5$): offline time

5.8 Varying $g$ ($k = 20$): estimation accuracy

5.9 Varying $g$ ($k = 20$): online time

5.10 Varying $g$ ($k = 20$): offline time

5.11 Varying $tw$ ($k = 25$): estimation accuracy

5.12 Varying $tw$ ($k = 25$): online time

5.13 Varying $tw$ ($k = 25$): offline time

5.14 Varying $tw$ ($k = 25$): estimation accuracy

5.15 Varying $tw$ ($k = 25$): online time

5.16 Varying $tw$ ($k = 25$): offline time
6.1 (a) Augmented twigs \( T_1 \cup T_2 \); (b) Growing \( T_1 \) from \( T \) .... 111

6.2 (a) Recursive decomposition scheme; (b) Fixed-sized decomposition scheme 116

6.3 Algorithm for recursive decomposition estimator .... 116

6.4 Fixed-sized decomposition algorithm .... 118

6.5 Algorithm TreeLattice-Build .... 120

6.6 Average estimation error on frequent-twigs workload: (a)Nasa (b)PSD (c)XMark (d)IMDB .... 128

6.7 Average estimation error on random workload: (a)Nasa (b)PSD (c)XMark (d)IMDB .... 128

6.8 Average estimation error distribution on frequent-twigs workload: (a)Nasa (b)PSD (c)XMark (d)IMDB .... 128

6.9 Average estimation error when varying summary size: (a)Nasa (b)PSD (c)XMark (d)IMDB .... 129

6.10 Average response time on frequent-twigs workload: (a)Nasa (b)PSD (c)XMark (d)IMDB .... 129

6.11 Average selectivity estimation error when varying delta (IMDB) .... 129
CHAPTER 1

INTRODUCTION

With advances in technology, the amount of data stored and accessible electronically has increased dramatically in recent years. Data can reside in different forms, ranging from unstructured data in text documents, semi-structured XML data, to highly structured data in relational database systems. Data is accessible through a variety of interfaces including web browsers, database query languages, application-specific interfaces, or data exchange formats. There is a pressing need to discover knowledge from such data, and then use the knowledge to guide decision making in all walks of human endeavor.

Accordingly, data can be classified into three main realms – unstructured data, structured data and semi-structured data. Unstructured data can be of any type, and does not necessarily follow any format or sequence, or rules. In this sense, it is not predictable. Examples of unstructured data include text, video, sound and images. In contrast, structured data is organized in semantic chunks (entities). Similar entities are grouped together (relations or classes) and entities in the same group have the same descriptions (attributes). Descriptions for all entities in a group form a schema. They have the same defined format, a predefined length, and are typically all present and follow the same order. Examples
of structured data include various relational data. Specifically in this thesis, we consider transactional data as an important class of structured data.

Transactional data consists of records of interactions between pairs of entities occurring over time. A transaction corresponds to one such record. As one can see, the derivation of a transaction follows a simple rule – it should consist of a set of items of a domain. A transactional dataset can be represented by a binary matrix, with columns corresponding to items, rows corresponding to transactions and matrix entries indicating the presence or absence of items in transactions. Therefore, we consider it to belong to the realm of structured data. Market basket data is a representative example of transactional data. Each basket is a transaction consisting of items purchased by the user. Furthermore, transactional data is a raw data format for many real-world graphs such as co-authorship social network data (each paper can be viewed as a transaction consisting of author items); movie-actor network data (each movie can be viewed as a transaction consisting of actor items); and friendship social network data (a person’s Yahoo! messenger’s buddy list can be viewed as a transaction consisting of person items). Arguably, any real-world graph can be thought of as derived from such a transactional dataset. Indeed, each transaction records an event – certain entities interact with each other at some time point. All transactions (events) together present a graph, where the nodes are the entities in the transactional data, and two nodes \((i, j)\) are connected to each other if they occur in the same transaction.

The third realm of data is the one that lies in between unstructured data and structured data – semi-structured data. Such data has certain structural information, but is not completely structured. Similar to structured data, semi-structured data is organized in semantic
entities and similar entities are grouped together. However in contrast to structured data, entities in the same group in semi-structured data may not have the same attributes, and the order of attributes is not necessarily important. Furthermore, not all attributes may be required, and size and type of the same attributes in a group may differ. XML data is perhaps the most important example of semi-structured data in use today. Structural information in XML data is specified by Document Type Definitions (DTDs) or XML schemas. However, they are not entirely structured. For example, they can contain unstructured components such as free text.

With the rapid growth of structured and semi-structured data, it is becoming increasingly clear that efficient modeling, querying, and analyzing mechanisms for such data is important to our decision making. In this thesis, we consider the problem of modeling, querying and analyzing large structured and semi-structured data. A wide variety of applications can benefit from progress made in this research. In the text below we list a series of examples.

1.1 Motivation

- **Recommender Systems / Personalization**

  Recommender systems form a specific type of information filtering (IF) technique that attempts to present information items (movies, music, books, news, web pages) that are likely of interest to the user. Typically, a recommender system compares the user’s profile to some reference characteristics. These characteristics may be from the information item (the content-based approach) or the user’s social environment (the collaborative filtering approach). These predictions are specific to the user and
thus this is a personalization technique. Personalization tries to present unique products and service to each customer. It has seen practical use with the emergence of interactive media such as the Internet. A web site can track customers’ behavior patterns and then infer their interests and make suggestions for the future. Many sites help customers make choices by organizing information and prioritizing it based on the individual’s liking. In some cases, the product itself can be customized using a configuration system (e.g., My Yahoo!, iGoogle). Efficient and effective mechanisms of user modeling, mining associations existing in the data is indispensable to a successful recommender system and personalization practice.

• Social Network Analysis

A social network is a social structure consisting of nodes (which are generally individuals or organizations) that are tied by one or more specific types of interdependency, such as values, visions, idea, financial exchange, friends, kinship, dislike, conflict, trade, web links, or disease transmission (epidemiology) [1]. Social network analysis views social relationships in terms of nodes and their ties. Nodes are the individual actors within the networks, and ties are the relationships between the actors. Research in a number of academic fields has shown that social networks operate on many levels, from families up to the level of nations, and play a critical role in determining the way problems are solved, organizations are run, and the degree to which individuals succeed in achieving their goals. Common social network analysis tasks include: entity ranking – to determine the popularity (importance) of the entities in a network; entity classification – to label unlabeled entities in a network, whereby a small portion of entities are labeled; entity clustering (also known as the network community discovery) – to find highly connected components in a network where
entities share some common characteristics or are closely related to each other; link prediction – to make predictions on the future interactions among entities in a network; anomaly detection – to discover undergoing suspicious events in a network. An example of suspicious events could be two persons in a communication network who have never talked to each other begin to do so rather frequently recently, or two computers that have never communicated to each other exchange a significant amount of data recently, etc. Getoor et al. [2] have a good survey paper on various network analysis tasks.

• XML Query Optimization and Selectivity Estimation

Since its introduction in last decade, XML has evolved from a mark-up language for web documents to an emerging standard for data exchange and integration over the Internet. With the rapid growth of available XML data, it is becoming increasingly clear that the design of efficient high-level querying mechanisms have become necessary. To this end, path and tree-based querying mechanisms have been developed since XML data can be usually represented as rooted and labeled trees. Efficient support for accurately estimating their selectivity is crucial for the optimization of complex XML queries. This is analogous to selectivity estimation in relational database settings [3–6]. Accurate selectivity estimation is also desirable in interactive settings and for approximate querying. For instance, an end-user can interactively refine their query if they know it will return an overwhelmingly large result set. Similarly, the estimated value can be returned as an approximate answer to aggregate queries using the COUNT primitive. Efficient XML data modeling and selectivity estimation mechanisms are desirable for XML data processing.
1.2 Challenges in Processing Large Structured and Semi-Structured Data

We enumerate a series of great challenges in dealing with large structured and semi-structured data as follows:

• Data can be extremely large (e.g. web graphs). Data describing hundreds of thousands of entities are common in real life. Developing efficient and scalable mining algorithms is a challenging task one should address. Furthermore, such data is usually sparse. For example, in market basket data, a typical user buys only a few items from a large number of items offered by the grocery store. The high data dimensionality plus the extreme data sparsity make many traditional analytical methods infeasible. However, the sparsity may be advantageous from the computation perspective. We may employ this to come up with computationally efficient analytical methods.

• Data can be noisy. For example, in a network data, it is not clear whether an edge (interaction) has a true real-world correspondence. For example, there exist large amounts of false positive interactions in protein-protein interaction networks, making analysis on such data even more difficult. Analytical tools should be capable of modeling and reasoning in an environment with uncertainty.

• Data is usually dynamic in nature. Data evolves over time through the addition of fresh data chunks and deletion of stale ones, representing constant changes in the interactions among involved entities. Understanding the dynamics and evolution of these interactions in data is crucial to making meaningful and insightful inferences. Identifying the changing portions and the novel events in the data, characterizing the
type of change the data is undergoing, and predicting future events are just a few of the many challenges that need to be addressed.

There have been considerable efforts on studying the problem of modeling, querying and analyzing structured and semi-structured data, and there has been tremendous progress achieved (See Chapter 2). However, we believe that this problem still remains open and in-depth research is still needed. First, finding novel and interesting patterns from large data is not a fully solved task. In fact, Faloutsos and Megalooikonomou in a recent paper [7], relate pattern mining to data compression and Kolmogorov complexity; the latter being undecidable. They imply “pattern mining will always be an art, and specifically, the art for looking for better models”. Second, how to efficiently model a large dataset is not clear yet. Many learning approaches cannot scale to the size of the data we currently have. Progress in solving this problem will be very meaningful to many data analysis tasks. Third, exploration of the relationship between frequent pattern analysis and probabilistic learning has also not been fully studied.

In our study, we propose a framework for processing large structure and semi-structured data. The main idea of the framework is to leverage local patterns and probabilistic models for the data analysis. Local patterns are those patterns at a fine scale where each one only encodes a small amount of distributional information of the data. Probabilistic models are a general framework of modeling the data with uncertainty and reasoning in a coherent manner about events in the data. Our framework consists of three steps as follows.

**Step 1:** Mining non-redundant local patterns from data;

**Step 2:** Gluing these local patterns together by employing probabilistic models;

**Step 3:** Reasoning (making inference) over the data for solving various data analysis tasks.
**Thesis Statement:** We contend that an aggregation of local patterns mined from structured and semi-structured data provide key insights of the data and can be used to model the data. We argue that probabilistic models are an elegant tool to glue these local patterns together. Finally, we demonstrate that the learned models can be effectively employed for various data analysis tasks.

### 1.3 Contributions

In this thesis we explore the interesting relationship between frequent pattern analysis and probabilistic learning, and we address a number of specific research questions. The contributions of this thesis are summarized as follows:

- **Mining non-redundant itemset patterns on large transactional data.** Efficient algorithms to compute frequent patterns and rules associated with these patterns exist nowadays. However, often times in many real-world problems the number of frequent patterns mined for various parametric settings is extremely large leaving the end-user swamped when it comes to interpreting the results. We present an approach of identifying non-redundant itemset patterns from a large collection of frequent itemsets on transactional data. We show that our approach can effectively eliminate redundancies from a large collection of itemset patterns.

- **Employing local probabilistic models to glue itemset patterns together, to solve the link prediction problem in social network analysis.** The new technique effectively combines topology analysis on network structure data and frequency analysis on network event log data. For a given pair of nodes, the local common neighborhood of these two nodes in the network is first identified governed by the topology of the network. Then relevant local patterns are collected from the underlying event log
data and used to learn a local probabilistic model over this local common neighborhood. Finally, we use this model to induce a co-occurrence probability feature of the two nodes and use it to aid in link prediction. We demonstrate that this co-occurrence probability feature can be computed in a scalable manner and is highly discriminatory for link prediction when compared with state-of-the-art topological and semantic features on several real world datasets. Moreover, we demonstrate that the co-occurrence probability feature can be effectively combined with other features (semantic and topological features). One can then employ standard supervised learning algorithms to make link predictions.

- We propose a global modeling strategy to glue itemset patterns on large transactional data. We present an algorithm of learning approximate Markov random fields based on frequent itemset patterns from a large transactional data. The algorithm adopts a divide-and-conquer style learning approach based on graph partitioning. Empirical study shows that the models are effective in modeling the data and approximately answering queries on the data.

- Mining non-redundant tree patterns on XML data. The above idea of identifying non-redundant itemsets is generalized to deal with more complex structural tree patterns. We present an approach to identifying non-redundant tree patterns from a large collection of structural tree patterns. We show that our approach can effectively eliminate redundancies from a large collection of structural tree patterns. Furthermore, we present techniques of employing these non-redundant tree patterns as the summary statistics for the XML data to handle XML twig selectivity queries effectively.
1.4 Organization

In Chapter 2 we go over necessary background knowledge and related work on processing large structured and semi-structured data. We review frequent pattern mining techniques and probabilistic graphical models. Chapter 3 presents our work on identifying non-redundant itemset patterns from a large collection of itemset patterns on transactional data. The non-redundant itemsets together offer a concise summarization of the original collection of itemsets. Chapter 4 includes our efforts on employing a local modeling strategy to glue local patterns together to solve the link prediction problem in social network analysis. Chapter 5 addresses our work on employing a global modeling strategy to glue local patterns together to solve the approximate query answering on transactional data. Chapter 6 describes our efforts to identify non-redundant structural tree patterns on XML data and how we leverage them to model and query the XML data. In Chapter 7, we summarize our contributions and discuss extensions and directions for future work.
CHAPTER 2

BACKGROUND AND RELATED WORK

In our framework, we first apply frequent pattern mining techniques to mine frequently occurring patterns from data. Then we identify non-redundant patterns and employ probabilistic approaches to glue them together for tackling various data analysis tasks. In the text below, we first review necessary background knowledge on frequent pattern mining and probabilistic models, and then review related work on processing large structured data and semi-structured data.

2.1 Frequent Pattern Mining

The problem of mining frequent patterns was first introduced in Agrawal et al.’s pioneering work [8] for market basket analysis in the form of association rule mining. It analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets. For instance, if customers are buying milk, how likely are they going to also buy bread in the meanwhile? Such knowledge can lead to increased sales by helping retailers do personalized marketing, design advertisement campaigns, and arrange their shelf space. Mining frequent patterns has become an important data mining problem and has received considerable attention in recent years. The goal is
to discover frequently occurring patterns from data of various forms. In the text below, we briefly describe frequent itemset patterns and frequent tree patterns.

## 2.1.1 Frequent Itemset Pattern

Let $I$ be a set of items, $i_1, i_2, \ldots, i_d$. A subset of $I$ is called an itemset. The size of an itemset is the number of items it contains. An itemset of size $k$ is a $k$-itemset. A transactional dataset is a collection of itemsets, $D = \{t_1, t_2, \ldots, t_n\}$, where $t_i \subseteq I$. For any itemset $\alpha$, we write the transactions that contain $\alpha$ as $D_\alpha = \{t_i | \alpha \subseteq t_i \text{ and } t_i \in D\}$. An itemset $\alpha$ is said to be sub-itemset of an itemset $\beta$ if $\alpha \subset \beta$. Similarly, $\alpha$ is said to be super-itemset of $\beta$ if $\alpha \supset \beta$. In the probabilistic model context, each item can be modeled as a random variable.

**Definition 1 (Frequent itemset pattern).** For a transactional dataset $D$ and a user-specified non-negative threshold $\sigma$, an itemset $\alpha$ is frequent if $|D_\alpha| \geq \sigma$, where $|D_\alpha|$ and $\frac{|D_\alpha|}{|D|}$ are called the support and the relative support of $\alpha$ in $D$, respectively. The frequent itemset mining problem is to find all frequent itemsets given $D$ and $\sigma$.

We deem such frequent patterns to be local patterns since each pattern by itself only encodes a very small amount of information about the overall data distribution. There have been many efficient algorithms proposed to mine frequent itemsets from large transactional datasets. All of these algorithms heavily rely on the Apriori property [8] to effectively prune the space of infrequent patterns. This property states that any sub-itemset of a frequent itemset must also be frequent. Toivonen et al. [29] propose a sampling approach to mine frequent itemsets. Brin et al. [10] propose a dynamic itemset counting technique to mine frequent itemsets. Park et al. [9] and Zaki et al. [11] propose techniques to mine frequent itemsets in parallel and distributed settings. Han et al. [12] devise an FP-growth method.
that mines the complete set of frequent itemsets without candidate generation. Ghoting et al. [13] note that the primary performance bottlenecks of most extant mining algorithms are poor data locality and low instruction level parallelism (ILP). They propose a cache-conscious prefix tree to address this problem to improve the mining performance. More recently, researchers have examined the problem of mining compressed patterns. In the text below, we describe some of these approaches.

2.1.2 Closed, Maximal and Non-Derivable Itemsets

Definition 2 (Closed frequent itemset [30]). A frequent itemset \( \alpha \) is closed if there does not exist an itemset \( \beta \) such that \( \alpha \subseteq \beta \) and \( D_\alpha = D_\beta \).

Definition 3 (Maximal frequent itemset [15]). A frequent itemset \( \alpha \) is maximal if there does not exist an itemset \( \beta \) such that \( \alpha \subseteq \beta \) and \( \beta \) is frequent.

Definition 4 ((Non-)derivable frequent itemset [31, 32]). A frequent itemset \( \alpha \) is derivable if its support can be exactly inferred from the support of its sub-itemsets based on the inclusion-exclusion principle. Otherwise it is non-derivable.

A frequent itemset is closed if none of its super-itemsets have the same support. Let us look at an example: Itemset \( AB \) has support of 10, and itemset \( ABC \) has the same support. This implies that every time \( AB \) occurs in a transaction, \( C \) will occur in the same transaction. Then we say \( AB \) is not closed. On the other hand, if any \( AB \)'s super-itemset has a strictly lower support, we say \( AB \) is closed. A frequent itemset is maximal if none of its super-itemsets are frequent. Non-derivable itemsets (NDI) compress a collection of frequent itemsets by eliminating redundant patterns. More specifically, those patterns
that are exactly derivable from its sub-itemset patterns using the combinatorial inclusion-exclusion principle [31]) are deemed redundant and pruned. The following presents an example of the inclusion-exclusion principle and a non-derivable itemset in [31].

Consider the transactional dataset in Figure 2.1. The supports of the frequent itemsets we have collected are also shown in the figure. Now let us look at itemset \(ABCD\). From the inclusion-exclusion principle, we can derive a set of deduction rules on its support as follows:

\[
\begin{align*}
\text{Rule (1)} & \quad \text{support}(ABCD) \geq s_{ABC} + s_{ABD} + s_{ACD} + s_{BCD} - s_{AB} - s_{AC} - s_{AD} \\
& \quad - s_{BC} - s_{BD} - s_{CD} + s_{A} + s_{B} + s_{C} + s_{D} - S_{\emptyset} \\
\text{Rule (2)} & \quad \text{support}(ABCD) \leq s_{A} - s_{AB} - s_{AC} - s_{AD} + s_{ABC} + s_{ABD} + s_{ACD} \\
\text{Rule (3)} & \quad \text{support}(ABCD) \leq s_{B} - s_{AB} - s_{BC} - s_{BD} + s_{ABC} + s_{ABD} + s_{BCD} \\
\text{Rule (4)} & \quad \text{support}(ABCD) \leq s_{C} - s_{AC} - s_{BC} - s_{CD} + s_{ABC} + s_{ACD} + s_{BCD} \\
\text{Rule (5)} & \quad \text{support}(ABCD) \leq s_{D} - s_{AD} - s_{BD} - s_{CD} + s_{ABD} + s_{ACD} + s_{BCD}
\end{align*}
\]
After we plug in the concrete support values to the above rules, we can derive a lower bound on $ABCD$’s support as 1 (by Rule(7)). Furthermore, we can derive an upper bound on $ABCD$’s support as 1 again (by Rule(2)). Since these two bounds are equal, we know that $ABCD$’s support must be 1 without scanning the dataset. Accordingly, $ABCD$ is a derivable itemset.

Subsequently, the frequent itemset mining problem has been generalized to sequence, tree and graph transaction settings where a transaction of the dataset can represent a sequence, tree or graph, respectively. Correspondingly, the goal becomes mining sequence,
tree or graph patterns that occur frequently in these datasets. Many efficient and scalable methods for mining sequential patterns [16, 17], tree patterns [18–22] and graph patterns [23–28] have been proposed. In the text below, we take a closer look at frequent tree patterns.

2.1.3 Frequent Tree Pattern

The idea of mining frequent itemset patterns on transactional data can be extended to mining frequent tree patterns on tree transactional data where each transaction is a tree. Given a tree transactional dataset, \( D = \{T_0, T_1, \ldots, T_n\} \), the support of a tree pattern \( t \), \( \text{support}(t) \), denotes the number of transactions in \( D \) in which \( t \) is a subtree. The problem of mining frequent tree patterns is to find all tree patterns from \( D \) with support no less than a user-specified minimum support threshold.

Depending on whether \( T \)'s in \( D \) are ordered or unordered, one can mine frequent ordered tree patterns or frequent unordered tree patterns. Additionally, depending on different notions of subtree matching, one can mine frequent induced tree patterns or frequent embedded tree patterns. The induced tree pattern means that parental relations between vertices in the data tree must be the same as in the pattern tree. The embedded subtree pattern means that a parent in a pattern tree may be an ancestor in the data tree. The Apriori property usually still holds for such tree patterns, and existing mining algorithms again heavily rely on this property to effectively prune the space of infrequent patterns.

The tree pattern mining problem has received considerable attention in recent years. Asai et al. [18] propose an algorithm, Freqt, for mining frequent ordered induced tree patterns. The key of their algorithm is the notion of the rightmost expansion, a technique to grow a tree by attaching new nodes only on the rightmost branch of the tree. They use
this technique to effectively enumerate all ordered induced tree patterns. Zaki [21] propose
an algorithm, TreeMiner, for mining frequent ordered embedded tree patterns. TreeMiner
uses depth-first search and a novel scope-list vertical representation of trees to quickly com-
pute the candidate tree frequencies via scope-list joins based on interval algebra. Wang et
al. [22] propose two pattern-growth algorithms, Chopper and XSpanner, for mining fre-
quent ordered embedded tree patterns. Nijssen and Kok [20] extend the rightmost expa-
sion enumeration technique for ordered trees for mining frequent unordered induced tree
patterns. They show that the enumeration of unordered trees is not much more difficult than
the enumeration of ordered trees. Chi et al. [19] propose an algorithm, HybridTreeMiner,
for mining frequent unordered induced tree patterns. HybridTreeMiner uses a breadth-first
canonical form to enumerate all unordered induced tree patterns.

2.2 Probabilistic Graphical Models

In probability theory, statistics and machine learning, a probabilistic graphical model
represents dependencies among random variables by a graph in which nodes represent ran-
don variables and edges between nodes represent conditional dependencies. Depending
on whether graphs are directed or undirected, graphical models can be classified into undi-
rected graphical models and directed graphical models.

2.2.1 Undirected Graphical Model

Undirected graphical models [35], also known as Markov Random Fields (MRFs) or
Markov Networks, have an undirected graph structure. Such models have a simple defi-
nition of conditional independence (also known as global Markov property): two sets of
nodes $A$ and $B$ are conditionally independent given a third set $C$, if $A$ and $B$ are separated
by $C$. The joint distribution associated with an MRF factorizes as follows:
\[ p(X) = \frac{1}{Z(\psi)} \prod_{C_i \in \mathcal{C}} \psi_{C_i}(X_{C_i}) \]

where \( \mathcal{C} \) is the set of maximal cliques associated with the undirected graph; \( \psi_{C_i} \) is a potential function defined over the variables of clique \( C_i \) and \( \frac{1}{Z(\psi)} \) is a normalization term.

A clique is a subset of vertices\(^2\) in the graph that are fully-connected. A maximal clique is a clique that cannot have more vertices added and remain a valid clique. We associate with each maximal clique a non-negative and real-valued potential function.

2.2.2 Directed Graphical Model

Directed graphical models [36], also known as Bayesian Networks or Belief Networks (BNs), have a directed acyclic graph (DAG) structure and model causal relationships between random variables. Such models have a more complicated notion of independence that takes into account the directionality of the edges. Any two nodes that are not in a descendant or ancestor relationship are deemed conditionally independent given the values of their parents. Classic machine learning methods like hidden Markov models or neural networks can be considered as special cases of Bayesian networks. The joint distribution associated with a Bayesian network factorizes as follows (\( \pi_i \) is the set of parents of node \( i \)):

\[ p(X) = \prod_{i=1}^{n} p(X_i | X_{\pi_i}) \]

\(^2\)In this thesis we use these terms – node, vertex – interchangeably.
2.2.3 Inference

The goal of inference on a probabilistic model, is to compute the conditional probability of a variable or a set of variables, given the observed values of another set of variables. According to the Bayesian rule, computing a conditional probability can be converted to computing a ratio of two marginal probabilities, which requires one to marginalize (“summing out”) variables that are not in the query set. In fact, a directed graphical model needs to be converted into an undirected graphical model (moralization), before making inferences on it. A uniform treatment of inference on directed and undirected graphical models is thus possible [36]. When the model is relatively simple, we are able to compute exact marginal probabilities (exact inference) through the junction tree inference algorithm.

Junction Tree Inference Algorithm

The junction tree algorithm [37] is an exact inference framework for probabilistic models. It decomposes the original model into a hyper-tree, in which each tree node consists of a set of variables of the original graphical model. Two sets of variables associated with two tree nodes can overlap, and the overlapped part is called their separator. Each tree node corresponds to a unique maximum clique in the graph formed by triangulating the original model. Moreover, the junction tree satisfies the running intersection property, i.e., for every pair of cliques $V$ and $W$, all cliques on the path between $V$ and $W$ must contain $V \cap W$. Each tree node has an expectation on the marginal probability over its associated variables, called its belief. The beliefs of all the tree nodes propagate along all distinct paths respecting a two-phase message passing protocol in the junction tree. When the propagation terminates, the clique potentials and separator potentials are proportional to the local marginal probabilities. In other words, a global consistency has been achieved,
which implies that the inference problem within a tree node can be solved independently of the other tree nodes. The time complexity of the junction tree algorithm is exponential in the maximum number of variables contained in a tree node (also known as the *treewidth*, see Definition 5, of the original graphical model). See [37] for more details on the junction tree algorithm.

**Definition 5 (Treewidth).** The treewidth of a graph $G$ is the minimum $k$ such that $G$ can be decomposed into pieces forming a junction tree structure with at most $k + 1$ vertices per piece.

Treewidth is an important inherent property of a graph. Many NP-hard graph problems become polynomial or linear time solvable when restricted to graphs of bounded treewidth. Determining the treewidth of a general graph is NP hard and there exist approximation algorithms and heuristics for evaluating treewidth [38,39]. The *Maximum Cardinality Search* (MCS)-ordering is a simple and commonly used upper bound heuristic for treewidth [40].

If a model is relatively simple (with a relatively low treewidth), then the junction tree algorithm can yield exact inferences very efficiently. However, when a model is complex (with a large treewidth), the exact inference will be intractable. In such cases we have to resort to approximate inference algorithms.

**Approximate Inference Algorithms**

For many real-world applications of interest, it is quite common that the treewidth of the model will be very large, making the exact inference intractable. As a result, we have to resort to approximate inference schemes. There have been various approximation approaches proposed which often work reasonably well in practice. Below we list several popular techniques.
• Variational methods [41]. The variational methods yield approximations to marginal probabilities via the solution to an optimization problem derived from the corresponding inference problem. This optimization problem exploits the graphical structure of the model. The simplest variational method is the mean field algorithm [41], which exploits the law of large numbers to approximate large sums of random variables by their means.

• Loopy belief propagation [36]. The basic idea is to apply Pearl’s belief propagation algorithm [36] directly to the general loopy graphical models. This seemingly runs into the risk of repeated counting. However it usually works reasonably well in practice [42]. The explanation for this surprising fact that has been found is that this approach specifically targets on minimizing the Bethe free energy, and is able to achieve local minimum solutions [43].

• Markov chain Monte Carlo (MCMC) sampling methods [44]. MCMC sampling is the most widely used method of approximate inference. The idea behind MCMC is to approximate a distribution by forming an empirical estimate from samples. We construct a Markov chain with the approximate stationary distribution, and collect the samples from a chain which has converged. The simplest MCMC algorithm is the \textit{Gibbs sampler}. More specifically, we specify a full conditional distribution \( p(x_i|\neg x_i) \) for each variable \( x_i \) in the model, where \( \neg x_i \) is the set of variables in the model not including \( x_i \). Then we draw samples from it. Note that the Markov property indicates that it is sufficient to condition on the neighboring variables of \( x_i \) in the model. Gibbs sampling proceeds by sampling each \( x_i \) from its conditional distribution, given the current values of the other neighboring variables. Marginal
probabilities can be estimated by summing over the samples. The quality of the
approximate inference is reasonably good when the sample size is large enough. To
diminish the effect of the starting distribution, we generally discard a certain number
of early iterations, referred to as burn-in [45].

2.3 Learning Markov Random Fields

Markov random fields have been used in an ever-growing variety of applications, such
as computer vision, natural language processing, computational biology. However, learn-
ing such models from real world data is a challenging task. The learning task has two
components – structure learning and parameter estimation. Structure learning is to learn
the graphical structure of the model, while parameter estimation is to learn parameters of
potential functions in the model. Structure learning is important since it is the first step
of learning a model and also, a “good” model structure can provide meaningful insights
about the underlying structure of the domain. Structure learning is considered much harder
than parameter estimation, since the space of possible graphical structures is exponentially
large. The research efforts on structure learning include greedy local heuristic search-based
methods [46], explicitly searches over the space of low-treewidth models [47, 48] and re-
cently $L_1$ regularization-based learning algorithms [49]. We note that there have been no
theoretical justifications on the optimality of various structure leaning algorithms and we
believe the structure learning problem remains widely open.

Parameter estimation for a model of a fixed structure is also not easy, even though it’s
considered simpler than structure learning. The key difficulty is that the maximum likeli-
hood parameters of these models have no analytic closed from. Therefore, finding these
parameters requires an iterative procedure such as conjugate gradient [50, 51] or variations
of iterative scaling [46, 52]. Inevitably, during the learning process, we have to execute repeated inferences that can be computationally expensive. The research efforts on parameter estimation include variations of iterative scaling algorithms [46, 52], Newton’s method or Quasi-Newton’s method-based algorithms [50, 51], and sampling-based model fitting algorithms [53–55]. In our study, we focus on using local patterns from the data to build probabilistic models of the data.

Using Frequent Itemsets to Build an MRF

The idea of using frequent itemsets to build an MRF was first proposed by Pavlov et al. [56]. A \( k \)-itemset and its support represents a \( k \)-way statistic and can be viewed as a constraint on the true underlying distribution where the data comes from. Given a set of itemset constraints, a maximum entropy distribution satisfying all these constraints is selected as the estimate for the unknown true underlying distribution. The principle of maximum entropy instructs us to choose the model with maximal uncertainty, as measured by the Shannon entropy [57], since this is the least-committed model among all those consistent with our prior knowledge. A maximum entropy model has the following form [58]:

\[
p_\theta(X) = \frac{1}{Z(\theta)} \exp(f(X) \cdot \theta)
\]

As one can see, a maximum entropy distribution specifies an MRF. From the model learning perspective, one can see that we use frequent itemsets to learn the model structure. Frequent itemsets together specify the model structure – there is an edge between two nodes in the graph if they co-occur in one pattern. As we will show next, we can use a simple iterative scaling algorithm to fit all parameters in the model.
Input: $C$, a collection of itemsets;  
Output: an MRF $\mathcal{M}$;  

1: Obtain all involved variables $v$ and initialize parameters of $\mathcal{M}$  
(typically uniform over $v$);  
2: while Not all constraints are satisfied do  
3: for all constraint $c \in C$ do  
4: Update $\mathcal{M}$ to force it to satisfy $c$;  
5: end for  
6: end while  
7: return $\mathcal{M}$;  

Figure 2.2: Iterative scaling algorithm

The iterative scaling algorithm can be used to learn a maximum entropy distribution from a set of itemsets. Figure 2.2 presents a high-level outline of a computationally efficient version of the algorithm given by Jelinek [59]. It has been shown that the iterative process will converge if all the constraints are consistent. This condition naturally holds in this setting since all frequent itemsets are indeed “true” patterns from the data. The algorithm works as follows: All itemset constraints are examined in turn within each iteration (line 2). At one time, the expected value of one selected constrain is evaluated from the current model parameters. Then the algorithm will compare this expected value against the empirical value to adjust the parameter corresponding to that constraint to force the model to satisfy the constraint (line 4). This is always plausible since we have an additional free normalization parameter to use. It is worth pointing out that the iterative scaling algorithm is usually not tractable when learning very large and complex models.

The following gives an example of the maximal entropy distribution and the corresponding MRF inferred from a set of itemsets. Suppose we have collected the itemsets as $\{x_1, x_2, x_3, x_4, x_5, x_1x_2, x_1x_3, x_2x_3, x_3x_4, x_4x_5, x_1x_2x_3\}$. Let $\mathcal{X}_Q$ be $\{x_1, x_2, x_3, x_4, x_5\}$.  

24
Then the maximal entropy distribution on $x_Q$ has the following product form:

$$p(x_Q) = \mu_0 \cdot \mu_1 I(x_1=1) \cdot \mu_2 I(x_2=1) \cdot \mu_3 I(x_3=1) \cdot \mu_4 I(x_4=1) \cdot \mu_5 I(x_5=1) \cdot \mu_6 I(x_1=x_2=1) \cdot \mu_7 I(x_1=x_3=1) \cdot \mu_8 I(x_2=x_3=1) \cdot \mu_9 I(x_3=x_4=1) \cdot \mu_{10} I(x_4=x_5=1) \cdot \mu_{11} I(x_1=x_1=x_3=1)$$

where $I()$ is an indication function for the corresponding constraint and the constants $\mu_0, \ldots, \mu_{11}$ are model parameters that need to be estimated from the data by the iterative scaling algorithm. Figure 2.3 shows the corresponding MRF. In particular, $C_1, C_2, C_3$ are the three maximum cliques in the model, and the potential functions defined over these cliques are listed below.

$$\psi_{C_1}(X_{C_1}) = \mu_1 I(x_1=1) \cdot \mu_2 I(x_2=1) \cdot \mu_3 I(x_3=1) \cdot \mu_6 I(x_1=x_2=1) \cdot \mu_7 I(x_1=x_3=1) \cdot \mu_8 I(x_2=x_3=1) \cdot \mu_9 I(x_3=x_4=1) \cdot \mu_{10} I(x_4=x_5=1) \cdot \mu_{11} I(x_1=x_1=x_3=1)$$

$$\psi_{C_2}(X_{C_2}) = \mu_4 I(x_4=1) \cdot \mu_9 I(x_3=x_4=1)$$
\[ \psi_{C_3}(X_{C_5}) = \mu_{I}^{I(x_5=1)} \cdot \mu_{I}^{I(x_4=x_5=1)} \]

2.4 Related Work

The problem of processing large structured and semi-structured data has received tremendous attention from multiple research communities. In the text below, we briefly review some of these efforts.

2.4.1 Processing Transactional data

Apart from the work on mining frequent patterns from transactional data as we have presented, researchers have also been concerned with the problem of modeling transactional data with probabilistic models. Pavlov et al. [56, 60] investigate different modeling strategies for query approximation, such as mixture Bernoulli model, Chow-Liu tree model, inclusion-exclusion AD-Tree model, online Bayesian networks and online Markov random fields. They show that the online Markov random fields approach yields the best quality results on sparse data. Hollmen et al. [61] use frequent itemsets to learn mixture maximum entropy models of transactional data at a local scale. They only model the most frequent items. Goldenberg et al. [62] propose an algorithm, Screen-based Bayes Net Structure search (SBNS), for employing frequent itemsets to learn large Bayesian networks on transactional data.

2.4.2 Large Network Analysis

Real-world networks have emerged in a surprisingly diverse range of disciplines, spanning from information technology, sociology, physics, ecology to biology and many more. As a result, there have been many efforts on network analysis across multiple research communities. Network analysis has been traditionally studied by researchers in social sciences
and statistical physics field [63–65,65–67]. Many times, the objects under investigation are relatively small networks. Recently, it has gained a significant amount of attention from computer scientists, mainly because of the emergence of large real information networks, such as web graphs, friendship networks, email communication networks, etc. Traditional analytical methods are often not capable of processing such large networks. Computer scientists have focused on devising scalable computational methods for analyzing such large networks. These efforts have formed a distinct branch in the field – “graph mining”\(^3\), where the goal is to apply data mining and machine learning techniques to discover meaningful patterns from large graphs.

PageRank [68] and HITS [69,70] algorithms are two notable approaches to rank entities in a network. The basic idea is to take random walks in a network, and entities that are visited more often are deemed more popular. Recently, Domingos et al. [71] propose a probabilistic approach to mining the network value of customers for “viral” marketing. The network value of a customer is the expected sales increase to other customers that results from marketing to that customer [71]. The network value of a customer takes into consideration the influential factor of the customer in the network. Those customers with higher network values should be marketed to directly.

There has been much work on clustering entities in a network. The Markov clustering algorithm is a flow-based entity clustering algorithm proposed by Dongen et al. [72]. Newman [73] introduces the notion of betweenness centrality of nodes and based on this notion to cluster network entities. Betweenness [74] is a measure of the centrality of a node and its influence over data flows in a network. For a node \(v\), its betweenness value is calculated as the fraction of the shortest geodesic paths between node pairs that pass through node

\(^3\)in this thesis we use these terms – graph, network – interchangeably
Nodes of high betweenness values are broken successively until the underlying cluster structure is uncovered. Also, spectral clustering methods have been proposed [75] to cluster entities and detect community structures in a network.

Liben-Nowell et al. [76] use topological features (such as the shortest distance, the number of common neighbors between two nodes, etc.) from a network to predict future links in a network. Hasan et al. [77] and O’Madadhain et al. [78] find that semantic features, such as the number of matching keywords between two authors in a bibliographic setting, are useful in the link prediction task.

Recently, many researchers have been working on the anomaly detection task on network data. Sun et al. [79] examine the anomaly detection task on bipartite graphs, where they use the relevance score of nodes to detect anomalies. Rattigan and Jensen [80] propose the anomalous link discovery task. They equate anomalous links with those that are statistically unlikely.

Meanwhile, there have been many efforts on probabilistically modeling relational network data. The research efforts along this line include probabilistic relational models [81, 82], probabilistic dependency networks [83, 84], and Markov logic networks [85, 86]. Probabilistic relational models extend traditional graphical models such as Bayesian networks to relational domains, removing the assumption of independent and identically distributed instances that underlies conventional learning techniques. Probabilistic dependency networks extend dependence networks, proposed by Heckerman et al. [87], for relational data. They can represent and reason with the cyclic dependencies required to express and exploit autocorrelation during collective inference. Markov logic networks unify first order logic and probabilistic models and can be viewed as a template for constructing Markov networks.
2.4.3 XML Data Processing

There have been many research efforts on efficiently processing XML data. Much research has focused on the efficient retrieval of XML data from large databases. Several query languages, such as XPath, XQuery have been developed to specify complex structured traversal over the data. Much research has been devoted to solving the selectivity estimation problem of various XML queries, since it is crucial to evaluating an optimal query plan. The early work in this area has focused on determining the selectivity of XML path expressions [88–93]. The Lore system [88] adopts a Markov model-based approach for this purpose. The Markov table method [90] improves on the Lore system through the use of intelligent pruning and aggregation to reduce space requirements. Recently, Lim and Wang proposed XPathLearner [89], an on-line, tunable Markov table method which has been shown to be effective for path expression selectivity estimation.

More recently, researchers have focused on selectivity estimation for tree-like XML queries [94–98]. Examples include Correlated Sub-Trees [95], XSketches [96, 98] and TreeSketches [97]. Among these it has been shown that TreeSketches is the most accurate and efficient method [97]. TreeSketches [97], a successor of XSketches, clusters the similar fragments of XML data together to generate its synopsis. The granularity of the clustering depends on the memory budget. To estimate the selectivity of XML twig queries, the above approaches define a summary data structure that houses important statistics about the data from which the selectivity may be estimated. Important issues at hand include: the quality of estimation from the given summary; the time to construct the summary; and finally, the time to estimate the selectivity of queries from the summary.
CHAPTER 3

MINING NON-REDUNDANT FREQUENT ITEMSETS ON LARGE TRANSACTIONAL DATA

The problem of mining frequent itemset patterns in transactional datasets is an important one with many applications. Efficient algorithms to compute frequent itemsets and association rules associated with these patterns exist. However, often times in many real-world applications the number of frequent patterns mined for various parametric settings is extremely large leaving the end-user swamped when it comes to interpreting and using the results. One reason is that there usually exists a large amount of redundancy among these patterns. As a result, the problem of mining non-redundant patterns from a large collection of patterns becomes increasingly important. In this chapter, we present our work on mining non-redundant frequent itemset patterns on transactional data.

3.1 Related Work

The problem of mining non-redundant itemset patterns can be also viewed as concisely representing a large collection of itemset patterns (also known as pattern summarization). The pattern summarization is an important data mining task that has received considerable attention in recent years [30, 31, 33, 99, 100].
3.1.1 Itemset Pattern Summarization

We define the itemset pattern-summarization problem as follows: given a collection of frequent itemsets, we want to find a more concise representation such that the original collection of itemsets and their support information can be reasonably recovered. Additionally, the summarization should be tunable in terms of controlling the trade-off amongst often competing metrics, namely summarization quality, summary size or compactness, and efficiency.

3.1.2 Current Summarization Approaches

Researchers have proposed various strategies to summarize itemset patterns. This has led to research on problems such as mining of closed itemsets [30], maximal itemsets [99], non-derivable itemsets [31], and more recently pattern profiles [33].

Closed itemsets and non-derivable itemsets are lossless forms of compressing frequent itemset patterns, i.e. the full list of frequent itemsets and associated frequency counts can be exactly derived from the compressed representation. It is not clear which compressing scheme is better. Researchers have found that for some datasets and support thresholds we have $|NDI| < |Closed|$, while other datasets and support thresholds have $|Closed| < |NDI|$ [31]. Maximal itemsets allow greater compression when compared with closed itemsets, but the representation is lossy – the list of frequent itemsets can be exactly computed but the exact frequency counts associated with these frequent itemsets cannot be determined. There are some other lossy representations besides maximal itemsets. Top-$k$ patterns approach by Han et al. [101] presents the most frequent $k$ closed itemsets to the end-user. Xin et al. [102] extend this work to extract redundancy aware top-$k$ frequent itemset patterns. Error-tolerant patterns by Yang et al. [34] and Pei et al. [103]
allow certain amount of fluctuations in evaluating supports of itemset patterns. A recent approach by Afrati et al. [100] uses $K$ itemsets to recover a collection of frequent itemsets. However, it is not clear how to recover the support information with their approach.

Closed itemsets and non-derivable itemsets are two successful concise representations for a collection of frequent itemsets. In many cases, they can significantly reduce the number of itemsets in the representation without any information loss. However, the size of the resulting patterns can still be very large, which is why new summarization schemes are necessary.

**Pattern Profile Summarization Approach**

Recently, Yan et al. [33] propose a pattern summarization approach for frequent itemsets. The key notion is *pattern profile*, which can be viewed as a generalization of closed itemsets. Specifically, a pattern profile is a triple $< a, b, c >$ where $a$ is a set of items, $b$ is a distribution vector on these items and $c$ is the relative support of the whole pattern profile. A frequent itemset is a special pattern profile where the distribution vector entirely consists of “1.0”s. A pattern profile is a compressed representation of similar itemsets and can be used to summarize them. In the scheme proposed by Yan et al. [33], pattern profiles are compared based on the *Kullback-Leibler* (KL) divergence between their distribution vectors. The principle is that the pattern profiles having smaller KL divergence are more similar than those having larger KL divergence. Based on this similarity measure, the traditional $k$-means clustering algorithm is applied to cluster the itemsets into a user-specified number of groups. Then, a representative profile pattern is identified for each group and used as a compressed representation for that group of itemsets.
Yan et al. [33] demonstrate that the pattern profile approach can effectively summarize itemsets, resulting in good compression while retaining high recovery accuracy. However, from an efficiency perspective it is not clear how well this approach will scale to large datasets. The proposed strategy needs to repeatedly scan the original dataset in order to achieve good summarization quality. Obviously, this can become very expensive when dealing with large datasets. Furthermore, the resulting pattern profiles can be quite unbalanced in terms of their size and distribution, leading to poor interpretability of the results. Unfortunately, there is no clear way to alleviate this problem. Finally, the pattern profiles are not itemset patterns themselves. It’s not clear how to use them for data analysis. If we have to recover the original collection of itemsets from these pattern profiles before we can use them, the technique is much less attractive.

3.2 Proposed Approach

In our study, we explore and exploit the conditional independence in the patterns to compress them. To this end, we employ the undirected graphical models. We first present an important observation on non-derivable itemsets and probabilistic learning through the following lemmas.

**Lemma 1** Given a transactional dataset \( D \), the MRF \( M \) constructed from all of its \( \sigma \)-frequent itemsets is equivalent to \( M' \), the MRF constructed from only its \( \sigma \)-frequent non-derivable itemsets.

**Proof:** This is due to the universal applicability of the inclusion-exclusion principle. When we use all \( \sigma \)-frequent non-derivable itemsets to construct an MRF model, the model will maintain the exact support information for these itemsets. Later when we use the
model to infer the support for other itemset patterns, the estimation procedure has to satisfy the inclusion-exclusion principle, thus yielding exact estimations.

**Lemma 2** Given an itemset pattern $\alpha$ and all of its non-derivable sub-itemsets and their support estimations, i.e., $\hat{s}_1, \ldots, \hat{s}_l$ (true supports are $s_1, \ldots, s_l$, respectively). If these estimations are error bounded by $e_1, \ldots, e_l$, i.e., $|\hat{s}_1 - s_1| \leq e_1, \ldots, |\hat{s}_l - s_l| \leq e_l$, then the support estimation for $\alpha$, $\hat{s}(\alpha)$, derived from these sub-itemsets is error bounded by $e_1 + e_2 + \ldots + e_l$.

**Proof:** We infer the support of the itemset by applying the inclusion-exclusion principle based on its sub-itemsets’ support. If a sub-itemset is non-derivable, we use its support estimation directly, otherwise we recursively apply the inclusion-exclusion principle to derive its support estimation. It’s easy to see that $e_1 + e_2 + \ldots + e_l$ is the maximally possible error accumulation.

Motivated by the above lemmas, we focus on the task of summarizing non-derivable itemset patterns. These patterns capture non-redundant distributional information of the data according to Lemma 1. Furthermore, if we summarize these patterns well, the summarization quality for all other derivable patterns will be error bounded according to Lemma 2.

### 3.2.1 Summarizing Itemset Patterns Using MRFs

The idea behind our proposed summarization technique is simple. We use statistics of smaller itemset patterns to construct an MRF, and then use this model to infer the supports of larger itemset patterns. If the estimations are accurate enough (within a user-specified error tolerance), we bypass the corresponding patterns. Otherwise we use the extra information from them to augment the model. The summarization proceeds in a level-wise fashion. First, all 1-itemsets are collected and used to construct an MRF. Then we infer
the supports for all 2-itemsets. We bypass those 2-itemsets whose supports can be well estimated from the model and use the information of all “skewed” 2-itemsets to augment the model. We move on to process all 3-itemsets and so on. This process will be repeated level-by-level until we process all the itemset patterns. At the end of the process, all itemsets remaining in the resulting model afford a concise representation of the original collection of itemset patterns.

Essentially we select the “skewed” itemset patterns and add their information to the probabilistic model as it is constructed. Thus we expect that the final resulting model to be able to faithfully capture the most significant dependency information in the data, summarizing the original patterns well. In other words, we try to reduce the original collection of itemset patterns by eliminating redundancy. The MRF fully specifies the conditional independence in the data, thus if an itemset pattern does not introduce any extra significant dependency information to the current model, it will be pruned. Furthermore, we introduce a parameter $\delta$ to tune the granularity of the summarization. $\delta$ specifies the error tolerance during the summarization. If the estimation error of a pattern is within the tolerance, we bypass the pattern. Otherwise we label it as skewed. By specifying the error tolerance, the parameter $\delta$ provides a mechanism to trade-off summarization accuracy for compactness.

The formal summarization algorithm is presented in Figure 3.1. As we mentioned in Chapter 2, the iterative scaling algorithm can be used to learn an MRF from a set of itemsets. During the learning process, we need to iterate over all itemset constraints and repeatedly update the model to force it to satisfy the current itemset constraint. The model update relies on the support estimation for the current itemset constraint. Thus, we need to continuously make inferences on the current model. If the iterative scaling algorithm runs for $k$ iterations and there are $m$ itemset constraints, the time complexity of the algorithm
Input: $C$, a collection of frequent itemsets; 
$\delta$, an error tolerance threshold;
Output: $\mathcal{R}$, reduced collection of itemsets;

1: Obtain all 1-itemsets $\in C$ and their supports, 
   use them to initialize $\mathcal{R}$;
2: $k \leftarrow 2$;
3: while $k \leq MAX\_LEVEL$ do
4: Use itemsets in $\mathcal{R}$ to build an MRF $M$;
5: Obtain all $k$-itemsets in $C$ and their supports;
6: for all $k$-itemset $p$ do
7: Use $M$ to estimate $s(p)$, calculate estimation error $e$;
8: if $e > \delta$ then
9: add $p$ to $\mathcal{R}$;
10: end if
11: end for
12: $k \leftarrow k + 1$;
13: end while
14: return $\mathcal{R}$;

Figure 3.1: Itemset pattern summarization algorithm

will be $O(k \times m \times t)$, where $t$ is the average inference time over a constraint. Therefore, 
efficient inference is crucial to the running time of the learning algorithm. In our study, we 
exploit and evaluate two inference engines, the junction tree inference algorithm and the 
MCMC inference algorithm (See Chapter 2).

3.2.2 Generalized Non-derivable Itemsets

The probabilistic model-based summarization scheme returns a subset of the original 
collection of itemsets to the end-user. Similar to conclusions of Yan et al. [33] that pattern 
profiles can be viewed as generalized closed itemsets, the resulting itemsets in our summa-
rization approach can be viewed as generalized non-derivable itemsets. First, we construct 
the probabilistic models based on the non-derivable itemsets. As a result, all the itemsets
in the final summary are non-derivable. Second, we allow for a certain error tolerance when summarizing the itemset patterns. If a particular itemset follows the currently-known conditional independence structure specified by the model, we consider its support to be known. Note that there might be cases where we are not able to derive an itemset’s support based solely on the inclusion-exclusion principle. We are able to derive it according to the further conditional independence information, however. In contrast, previously an itemset is derivable only when its support can be completely determined from the support of its sub-itemsets based on the inclusion-exclusion principle alone. Essentially, we relax the requirement for an itemset to be “derivable”, which will significantly increase the number of derivable patterns. Furthermore, the greater the relaxation, the greater number of derivable patterns, which will result in a more compact summarization.

3.3 Experimental Results

In this section, we examine the performance of our proposed approach on real datasets. We compare our probabilistic model-based summarization (abbreviated as PM) against the state-of-the-art pattern profile summarization scheme (abbreviated as PP). The summarization algorithm is implemented in C++. The junction tree and Gibbs sampling inference algorithms are implemented using Intel’s Open-Source Probabilistic Networks Library\(^4\). Also, we implement the pattern profile summarization algorithm in C++ and we tune it to achieve performance similar to that reported in previous work [33].

3.3.1 Experimental Setup

Unless otherwise noted, all the experiments are conducted on a Pentium 4 2.66GHz machine with 1GB RAM running Linux 2.6.8. We use the implementation of Apriori\(^4\)https://sourceforge.net/projects/openpnl/
Table 3.1: General characteristics of the datasets. $k$ is the number of distinct items, $n$ is the number of records, $m$ is the number of total items and $d = \frac{m}{kn}$ is the density index.

<table>
<thead>
<tr>
<th></th>
<th>$k$</th>
<th>$n$</th>
<th>$m$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chess</td>
<td>75</td>
<td>3196</td>
<td>118252</td>
<td>0.493</td>
</tr>
<tr>
<td>Accidents</td>
<td>468</td>
<td>340183</td>
<td>11500870</td>
<td>0.0722</td>
</tr>
<tr>
<td>Mushroom</td>
<td>119</td>
<td>8124</td>
<td>186852</td>
<td>0.193</td>
</tr>
<tr>
<td>Web</td>
<td>294</td>
<td>32711</td>
<td>98654</td>
<td>0.0102</td>
</tr>
</tbody>
</table>

algorithm in [104] to collect the frequent itemsets and the corresponding closed itemsets. We use the implementation in [32] to collect frequent non-derivable itemsets. We detail the datasets and performance metrics considered in our evaluation in the text below.

Datasets: We use four publicly available datasets in our experiments. They are the Chess dataset with 3196 transactions and 75 distinct items; the Accidents dataset with 340183 transactions and 468 distinct items; the Mushroom dataset with 8124 transactions and 119 distinct items; and the Microsoft Anonymous Web dataset with 32711 transactions and 294 distinct items. The first three datasets are publicly available at the FIMI repository\(^5\) and the last Web dataset is publicly available at the UCI KDD archive\(^6\). The main characteristics of the datasets are summarized in Table 3.1. As can be seen, the Chess and Mushroom datasets are relatively dense. The Web dataset is quite sparse and the Accidents dataset is somewhere in between.

Summarization accuracy:

**Definition 6** Restoration error. Given a collection of itemset $\Phi = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}$, the quality of a pattern summarization can be evaluated by the following average relative error (called restoration error),

\[5\text{http://fimi.cs.helsinki.fi/}
\[6\text{http://kdd.ics.uci.edu/}

38
\[ E = \sum_{\alpha_k \in \Phi} \left( \frac{|s(\alpha_k) - \hat{s}(\alpha_k)|}{s(\alpha_k)} \right) \]

where \( s \) is the true support and \( \hat{s} \) is its estimation. Restoration error measures the average relative error between the estimated support of a pattern and its true support. If this measure is small, it means that the estimated support of a pattern is very close to its true support.

**Summary size:** In order to make a fair comparison between the two approaches, we need to consider the summary size. The comparison should be made between summarizations which are of the same size. A larger summary is expected to be more accurate. Overall, we prefer the summarizations with low sizes which however yield small restoration errors. In our study we use the number of bytes taken by a summarization to quantify its size. Specifically, we assume an item in the summary takes 2 bytes (a short integer) and a floating point support in the summary takes 4 bytes. For example, the following itemset takes 8 bytes, \( \{(item_1, item_2), 0.1\} \) and the following pattern profile takes 22 bytes. \( \{(item_1, item_2, item_3), (1.0, 0.8, 0.6), (0.1)\} \)

**Summarization time:** We also consider the time taken to summarize the itemsets. Making a fair timing comparison between the two summarization schemes is not easy. Both our approach and pattern profile approach are iterative processes. The running times are highly dependent of the convergence criteria, which can be rather subjective. We report the timing results of those summarizations from which we collect the accuracy results.

### 3.3.2 Results on the Chess Dataset

First, we report the experimental results on the Chess dataset. For this set of experiments, we set \( \sigma = 2000 \) to collect the frequent itemsets. As a result, there are 166581 frequent itemsets, from which 1276 itemsets are non-derivable. We also collect all the
Figure 3.2: Results on the Chess dataset: (a) Restoration error (b) Summary size (c) Summarization time

<table>
<thead>
<tr>
<th>Itemset Size</th>
<th>No. of Total Itemsets</th>
<th>No. of Skewed Itemsets (Varying $\delta$)</th>
<th>0.05</th>
<th>0.10</th>
<th>0.15</th>
<th>0.20</th>
<th>0.25</th>
<th>0.30</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>335</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>653</td>
<td>14</td>
<td>19</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>257</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sum</td>
<td>1276</td>
<td>53</td>
<td>50</td>
<td>33</td>
<td>32</td>
<td>31</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: Skewed itemset distribution on the Chess dataset when varying error threshold
closed frequent itemsets at this support level for the pattern profile summarization scheme.

Figure 3.2a presents the summarization quality as we vary the error tolerance threshold. For our approach we report both results on summarizing all itemsets and on summarizing all non-derivable itemsets. For the pattern profile approach, we only report the results on summarizing all itemsets. As a baseline, the results based on a naive independence model are also plotted in the figure.

We see that the probabilistic model-based summarization scheme effectively summarizes the itemsets. The restoration error of all frequent patterns is slightly higher than that of non-derivable patterns. This is expected, considering our approach focuses on summarizing non-derivable patterns. It is worth pointing out that the restoration error on all frequent itemsets is also very small, supporting our claim that non-derivable patterns play a key role in representing the whole collection of frequent itemsets.

Furthermore, it can be clearly seen that the restoration error increases as we raise the error tolerance threshold. This is due to the fact that we will lose more information with larger error tolerance thresholds. Particularly, the summarization with the threshold above 0.25 becomes equivalent to the naive independence model-based summarization. The advantage of our approach over the pattern profile approach is clearly demonstrated in the figure. For the pattern profiles of the same size, the restoration error is much higher than that of our approach and is actually quite close to that of the naive independence model.

Figure 3.2b presents the summary sizes with different error tolerance thresholds. The sizes of the original collection of patterns and the naive independence model are also plotted here for reference purpose. As one can see, our summaries use a very small amount of space to represent a much larger collection of itemsets. For example, the summary takes
398 bytes at an error threshold of 0.05 to summarize itemsets of size 12480 bytes, about a 30-fold reduction.

Figure 3.2c presents the timing performance of our approach. As one can see, our approach quickly summarizes the itemsets of this dataset. In all cases, the summarization takes less than 5 seconds. In contrast, the pattern profile approach does not finish before it exhausts the memory. When we submit the summarization job to a computer with more memory (4GB RAM) and the same CPU speed at the Ohio Supercomputer Center (OSC)\(^7\), the pattern profile approach takes about 40 minutes to finish. Another trend is that our approach takes more time when using a lower error tolerance threshold, since the models with lower thresholds are more complex.

Table 3.2 presents the distribution of the skewed itemsets at different levels with respect to different error tolerance thresholds. As can be seen from the table, the numbers of skewed itemsets are very small at all the thresholds. For example, at the threshold of 0.05, there are 6, 14 and 2 skewed 2, 3 and 4-itemsets, respectively. As we raise the threshold, the overall number of skewed itemsets decreases.

It’s worth noting that the Chess dataset satisfies the independence assumption quite well. Thus the MRF model based summarization scheme works extremely well. A relatively simple MRF model is able to faithfully capture the conditional independence existing in the data, which results in a very low restoration error and an extremely fast summarization.

### 3.3.3 Results on the Accidents Dataset

In this section, we report the experimental results on the Accidents dataset. In this set of experiments, we set \(\sigma = 150000\) to collect the frequent itemset patterns, which results in

\(^7\)http://www.osc.edu/
Figure 3.3: Results on the Accidents dataset: (a) Restoration error (b) Summary size (c) Summarization time

<table>
<thead>
<tr>
<th>Itemset Size</th>
<th>No. of Total Itemsets</th>
<th>No. of Skewed Itemsets (Varying $\delta$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>2</td>
<td>253</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>1071</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>2135</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1788</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>210</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Sum</td>
<td>5486</td>
<td>99</td>
</tr>
</tbody>
</table>

Table 3.3: Skewed itemset distribution on the Accidents dataset when varying error threshold
18175 frequent itemsets, out of which 18175 are closed patterns and 5486 are non-derivable patterns.

Figure 3.3a presents the summarization quality as we vary the error tolerance thresholds. We see that the probabilistic model-based summarization scheme works extremely well on this dataset as well. The restoration errors on both all frequent patterns and non-derivable patterns are very low. We note that the independence assumption is satisfied well on this dataset also (the naive independence model yields the error of 5.27% and 6.77% on all patterns and non-derivable patterns, respectively). Furthermore, it can be clearly seen that the restoration error increases as we increase the error tolerance threshold.

Note that the pattern profile approach can not deal with this dataset due to its large size. The algorithm runs out of the memory after running for hours, even on the computer at OSC. Repeatedly dataset scanning-based summarization is very computation and memory intensive.

Figure 3.3b presents the summary sizes with different error tolerance thresholds. Again, the summary sizes are much smaller than the size of the original itemset patterns.

Figure 3.3c presents the timing results of our approach. As one can see, the probabilistic model-based approach again summarizes all the itemsets of this dataset very fast. Furthermore, the summarization with a smaller error tolerance threshold takes much more time. For example, the summarization takes 80 seconds when the threshold is 0.05. In contrast, it takes less than 1 second when the threshold is above 0.2.

Table 3.3 presents the distribution of the skewed itemsets. As can be seen from the table, the numbers of skewed itemsets are also very small on this dataset, indicating that the MRF model captures the distribution information and represents all the itemsets quite well. For example, at the error tolerance threshold of 0.05, there are only 13, 54, 1 and
3 skewed 2, 3, 4 and 5-itemset patterns, respectively. Compared with the numbers of the original non-derivable itemsets, 253, 1071, 2135 and 1788, the numbers of skewed itemsets are much smaller. Again, as we raise the error tolerance threshold, overall the numbers of skewed itemsets decrease.

Note that both of the Accidents dataset and the Chess dataset are relatively dense and largely satisfy the independence assumption. For this kind of dataset, the MRF model-based summarization scheme works extremely well. Interestingly, we note that on these two datasets, the frequent non-derivable patterns are much fewer than the frequent closed patterns. Take the Accidents dataset as an example, none of its 18175 frequent patterns is closed. In contrast, only 5486 out of 18175 patterns are non-derivable. This is usually a good sign that the MRF-based summarization will perform better than the pattern profile approach. Next we examine the performance of our approach on the skewed datasets that do not satisfy the independence assumption.

### 3.3.4 Results on the Mushroom Dataset

![Graphs](image)

(a) (b) (c)

Figure 3.4: Results on the Mushroom dataset: (a) Restoration error (b) Summary size (c) Summarization time
Table 3.4: Skewed itemset distribution on the Mushroom dataset when varying error threshold

<table>
<thead>
<tr>
<th>Itemset Size</th>
<th>No. of Total Itemsets</th>
<th>No. of Skewed Itemsets (Varying δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>207</td>
<td>78</td>
</tr>
<tr>
<td>3</td>
<td>269</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>Sum</td>
<td>534</td>
<td>144</td>
</tr>
</tbody>
</table>

In this section, we report the experimental results on the Mushroom dataset. The Mushroom dataset is also a relatively dense dataset. In this set of experiments, we set $\sigma = 2031$ (a support threshold of 25%) to collect the frequent itemset patterns, resulting in 5545 frequent itemsets, from which 688 are closed and 534 are non-derivable.

Figure 3.4a presents the restoration error as we vary the error tolerance thresholds. We see that the independence assumption does not hold well on this dataset. The restoration errors for all itemset patterns and non-derivable itemset patterns are 20% and 39%, respectively. From the figure, we see that the probabilistic model-based summarization scheme again works very well on this dataset. The restoration errors for both all frequent patterns and non-derivable patterns are reasonably low, and are much lower than that of the pattern profile summaries of the same size. Note that both approaches work much better than the naive independence model. Furthermore, we can lower the restoration error by lowering the error tolerance thresholds, which is at the cost of more space usage.

Figure 3.4b presents the summary sizes with different error tolerance thresholds. We note that the summaries take relatively more space, compared with that on the previous two datasets. Again, with a lower error tolerance threshold, the summary size is larger.
Figure 3.4c presents the timing performance of the two approaches. Our approach is much faster than the pattern profile approach. We see that both approaches take more time when the error tolerance threshold decreases. However, for the pattern profile approach, the increase of the running time is not as significant.

Table 3.4 presents the distribution of the skewed itemsets. Compared with the previous two datasets, the proportion of the skewed itemsets is much higher on this dataset, which signifies that the independence assumption does not hold on this dataset as well as on the previous two datasets. But overall, the numbers of skewed itemsets are still much less than the numbers of all the original itemsets. We note on this dataset, there is a small fluctuation when we raise the error tolerance threshold from 0.30 to 0.40, leading to more skewed itemsets. This is because that the larger threshold 0.40 results in much more skewed 3-itemsets (46 vs. 25), though it indeed results in less skewed 2-itemsets (11 vs. 25). The increase of the former outweighs the decrease of the latter, resulting in more skewed itemsets overall. However, the overall trend is still that the numbers of skewed itemsets become fewer when we use larger error tolerance thresholds.

3.3.5 Results on the Microsoft Web Dataset

In this section, we report the experimental results on the Microsoft Web dataset, which is the sparsest dataset. For the sparse datasets, the independence assumption generally does not hold, since all single item patterns have very low support in sparse data, and any pattern containing more than one item is prone to have the support close to 0 if the dataset follows independence assumption. In this set of experiments, we set $\sigma = 100$ to collect the frequent itemset patterns, resulting in 998 frequent itemsets. Specifically, all itemset patterns are closed and non-derivable, which is a common phenomena on sparse datasets.
Figure 3.5: Results on the Web dataset: (a) Restoration error (b) Summary size (c) Summarization time

<table>
<thead>
<tr>
<th>Itemset Size</th>
<th>No. of Total Itemsets</th>
<th>No. of Skewed Itemsets (Varying $\delta$)</th>
<th>0.10</th>
<th>0.20</th>
<th>0.30</th>
<th>0.40</th>
<th>0.50</th>
<th>0.60</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
</tr>
<tr>
<td>2</td>
<td>329</td>
<td>287</td>
<td>249</td>
<td>202</td>
<td>160</td>
<td>123</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>346</td>
<td>192</td>
<td>108</td>
<td>72</td>
<td>66</td>
<td>47</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>168</td>
<td>39</td>
<td>8</td>
<td>7</td>
<td>13</td>
<td>9</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>49</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>998</td>
<td>623</td>
<td>470</td>
<td>387</td>
<td>343</td>
<td>283</td>
<td>246</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.5: Skewed itemset distribution on the Web dataset when varying error threshold
This also makes the summarization task more difficult since there does not exist much redundancy among the itemsets to exploit.

Figure 3.5a presents the summarization quality as we vary the error tolerance thresholds. Note that since that all itemsets are non-derivable, there is no separate results for non-derivable itemsets here. We see that the independence assumption indeed does not hold on this dataset. The restoration error for all itemset patterns is 57.5%.

We see that the probabilistic model-based summarization scheme works reasonably well on this dataset. The restoration error is reasonably low. When we use 283 patterns (less than one third of all patterns, $\delta = 0.5$), the restoration error is 9.62%. When we use 470 patterns (less than half of all patterns, $\delta = 0.2$), the restoration error is reduced to 3.72%. Furthermore, our approach consistently outperforms the pattern profile approach in terms of restoration error. Again, both approaches work much better than the naive independence model.

Figure 3.5b presents the summary sizes with different error tolerance thresholds. We see that the summarizations take relatively much more space on this dataset than that on the Chess and Accidents datasets.

Figure 3.5c presents the timing performance of the two approaches. We see on this dataset, our approach is overall much slower than the pattern profile approach. This is due to the complexity of the underlying MRF models. On this dataset, the proportion of skewed patterns is relatively high, resulting in more complex models, especially when the error tolerance threshold is small. Consequently, the summarization becomes much slower. For example, when the error tolerance threshold of 0.1 is used, the summarization takes more than 1 hour. In contrast, the pattern profile approach takes less than 10 minutes.
Table 3.5 presents the distribution of the skewed itemsets. The proportion of the skewed itemsets is relatively high on this dataset, which is similar to that on the Mushroom dataset. Still, we are able to reduce the numbers of skewed itemset patterns by raising the error tolerance threshold.

3.3.6 Results on Approximate Inference-Based Summarization

In this section, we report the results on the approximate inference-based summarization. We focus on the comparison of the summarization quality between the exact inference-based summarization and the approximate inference-based summarization. To this end, we use the approximate inference-based approach to summarize the same collections of frequent itemsets as in the previous sets of experiments. We report the results on the Mushroom dataset. The other results are similar and are thus omitted. Specifically, we set the sample size to be 4000 and the first 10% of the sample is used as burn-in data when we use the Gibbs sampling inference algorithm.

Figure 3.6a presents the numbers of itemsets in the resulting summarizations. As one can see, the approximate inference-based summarization scheme usually yields more patterns under the same parametric setting. This is expected, since its support estimating is not as accurate as that of the exact inference-based summarization scheme. Consequently, there are more skewed itemsets identified and placed into the model during the summarization. But we see that the difference is not significant.

Figure 3.6b presents the restoration errors of the approximate inference-based summarizations. As one can see, the approximate inference-based scheme usually yields higher restoration errors, which is expected. But overall, the approximate inference-based scheme
still yields reasonably good summarizations. It significantly outperforms the pattern profile approach.

The approximate inference-based summarization scheme takes hours to finish. It’s worth pointing out that on these datasets, there is no need to use the approximate inference-based summarization scheme. It yields worse summarization with much more time. However, we just want to show that the approximate inference-based summarization scheme can yield comparable summarizations as the exact inference-based summarization scheme. When the underlying MRF becomes more complex (the treewidth becomes larger), we have to use the approximate inference-based summarization scheme. We are currently investigating different approximate inference algorithms besides the Gibbs sampling approach.

Figure 3.6: Approximate inference based summarization on the Mushroom dataset (a) No. of patterns after summarization (b) Restoration error (all patterns)
3.3.7 Result Summary

The experimental results have shown that the probabilistic model-based summarization scheme overall is very efficient and effective in summarizing itemsets. In most cases, it outperforms the pattern profile summarization scheme.

When datasets are dense and largely satisfy the conditional independence assumption, there usually exists a large amount of redundancy in the corresponding itemset. In such cases, our approach will be extremely efficient and effective. On the other hand, when datasets become sparse and do not satisfy conditional independence assumption well, the summarization task for our approach becomes more difficult. As a result, we have to spend more space and time on summarizing the corresponding itemsets.

3.4 Discussion

Interestingly, we note that the task of identifying non-redundant local itemset patterns and learning probabilistic graphical models are two interactive procedures. From our proposed summarization scheme, we see that we tentatively learn a global model using lowest order local patterns first, then we use this model to select the higher-order local patterns that can not be well explained by the current model. In other words, these patterns are informative to the model. Then this information is incorporated to learn a new and more complex model. This iterative process will proceed until we finish processing all the local patterns. At the final stage, we have two products: a collection of non-redundant local itemset patterns and a probabilistic graphical model based on these patterns.

Furthermore, we note that our proposed itemset summarization technique is essentially a feature selection technique. Each itemset and its support is essentially a feature about the underlying data distribution, and our goal is to select the most “informative” features
from a large set of features. The informativeness of an itemset feature is determined by its contribution to modeling the underlying data distribution. Note that our feature selection problem is has an unsupervised setting since we are learning generative models of the data rather than discriminative models. This suggests us that we may exploit unsupervised feature selection techniques to solve this problem. In fact, the above level-wise non-redundant itemset selection mechanism is a heuristic of selecting important features. There has been much work on feature selection in supervised settings [105–109]. However, there has not been as much work on feature selection in unsupervised settings [110, 111]. We are specifically interested in using $L_1$ regularization techniques for our purpose. It has been known that using $L_1$ regularization over the model parameters – optimizing a joint objective that trades off fit to data with the sum of the absolute values of the parameters – tends to lead to sparse models, where many weights are exactly zero. More formally, the joint objective is expressed as follows:

$$f(w) = \text{loss}(w) + \lambda \cdot ||w||_1$$  \hspace{1cm} (3.1)

where $\text{loss}()$ is an arbitrary differentiable convex loss function, and $||w||_1$ is the $L_1$ norm of the parameter vector $w$ and $\lambda$ is the regularization coefficient. The objective function is convex and therefore we can apply standard Newton’s method or Quasi-Newton’s method-based techniques to optimize it. It has been shown that they are more efficient than variations of the iterative scaling method [51]. The key operation in (Quasi-)Newton’s method is to evaluate gradient of the objective, that can be expressed as follows (we only consider the loss term $\text{loss}(w)$ here):
\[
\frac{d(l(w))}{d(w_i)} = E(f_i) - k_i
\]  

(3.2)

where \(E(f_i)\) is the expected value of the feature \(f_i\) according to the model and \(k_i\) is the empirical value. \(E(f_i)\) can be computed much easier in supervised settings than unsupervised settings since the latter requires explicit summations over all events in the modeled domain, which is clearly infeasible. In the natural language processing community, there have been efforts on learning whole sentence language models that requires tackling a similar problem as here – explicit summations over all possible sentences in the language domain, also clearly infeasible. As a result, various sampling techniques have been proposed there to approximately compute \(E(f_i)\), including Gibbs sampling, independence sampling and importance sampling [54].

In the meanwhile, there has been progress in learning large-scale \(L_1\) regularized log-linear models. Andrew and Gao [112] propose an L-BFGS [113]-based algorithm, Orthant-Wise Limited-memory Quasi-Newton (OWLQN), that can efficiently optimize the \(L_1\) regularized log-likelihood of log-linear models with millions of parameters. The main idea is to do L-BFGS in an orthant [112] where the gradient of the \(L_1\) loss term (the second term in 3.1) does not change. Each time BFGS tries to step out of that orthant, the new solution point will be projected back to the old orthant, and a new orthant will be figured out for further exploration. As a near future work, we plan to investigate the combined use of sampling-based gradient evaluation and OWLQN-based \(L_1\) regularized model learning to solve our itemset feature selection problem. We note that our study is the first one, to the best of our knowledge, to relate the itemset pattern summarization problem with the feature selection problem.
3.5 Conclusions

In this chapter, we have presented a novel approach to selecting non-redundant itemset patterns using MRFs, which exploit conditional independence relations among the items in the transactional data. The success of our approach on all the tested real-world datasets indicates that the conditional independence structure is quite common. This is particularly true when dealing with relatively dense datasets, whose dense structure leads to significant redundancy in mined itemsets. As a result, our approach is a viable option for many real-world datasets. Given the same summary size, our approach can achieve up to a 4-fold improvement in accuracy compared with the pattern profile approach. On certain datasets, it is orders of magnitude faster than the pattern profile approach.

In the future, we would like to examine the scalability of our approach on truly large-scale collections of itemsets which can result in very complex MRFs. We intend to evaluate and adapt ideas from the approximate probabilistic inference field for this purpose. Also, it would be interesting to evaluate the viability of the proposed approach in a streaming or incremental setting.
CHAPTER 4

LOCAL PROBABILISTIC MODEL-BASED LINK PREDICTION IN SOCIAL NETWORK ANALYSIS

We have described our research on mining non-redundant frequent itemset patterns on large transactional data. Naturally the next question is how to leverage these patterns to analyze the data. We explore a strategy to glue local patterns together for data analysis. In this strategy, at a given point in time we focus on a small local region of the whole data (e.g., a small sub-network from a large social or biological network). We first collect local patterns relevant to this region, then learn a lightweight local probabilistic model over this region, and finally employ the model to predict events within this region. In this chapter we present our work along this line in the context of link prediction in social network analysis.

4.1 Introduction

In recent times, there has been significant interest in understanding and characterizing the properties of large-scale networks or graphs. Part of this interest is because of the generality of the graph representation. For example, the data in many domains, such as social networks, gene regulatory networks and the World Wide Web, can be naturally modeled as graphs. Link prediction is an important problem in this context. Informally link prediction is concerned with the problem of predicting the (future) existence of links among nodes in
Link prediction is useful in many application domains, ranging from the recommendation of new products to end users in recommender systems to the detection of unseen links in terrorism networks, from the prediction of protein interactions to the prediction of collaborations among scientists, and from the prediction of friendship formations to the prediction of web hyperlinks.

In this article we focus on the problem of link prediction specifically in the context of evolving co-authorship networks. This has been a hotbed of recent research activity where much of the focus has been on encapsulating the topological or semantic information embedded in such networks to address the link prediction problem. In contrast in this article we explore the realm of probabilistic models derived from frequency statistics and use the resulting predictions from the probabilistic models as additional features to further enhance predictions made by topological-based and semantic-based link prediction algorithms.

Our probabilistic model is driven by two aspects. First, given the candidate link (say between nodes $X$ and $Y$) whose probability is to be estimated, we identify the central neighborhood set (say $W, X, Y, Z$), which are the nodes that are deemed germane to the estimation procedure. The identification of the central neighborhood set is governed by the local topology of the social network as viewed from the perspective of the two nodes whose link probability is to be estimated.

Second, once the central neighborhood set ($W, X, Y, Z$) is identified we learn a maximum entropy Markov random field model that estimates the joint probability of the nodes comprising the central neighborhood set, i.e., $p(W, X, Y, Z)$. In this context one can leverage the fact that most co-authorship networks are computed from an event log (an event corresponding to a publication). Multi-way statistics (e.g., non-derivable frequent itemsets whose elements are drawn from $(W, X, Y, Z)$) on these event logs can be used to constrain
and learn the model parameters efficiently [114]. The resulting model can then be used to estimate the link probability between $X$ and $Y$ which we henceforth denote as the co-occurrence probability.

In our empirical results we demonstrate that the co-occurrence probabilities inferred from the resulting model can be computed in a scalable manner and is highly discriminatory for link prediction when compared with state-of-the-art topological and semantic features on several real world datasets. Moreover, we demonstrate that the resulting co-occurrence probability can also be effectively combined with these other features and then one can employ any supervised learning technique to predict if a link will be formed between two nodes. Specifically, we employ a simple yet novel variant of the Katz score as a topological feature, one that scales reasonably well at some cost to accuracy. Additionally we describe and use straightforward state-of-the-art methods to measure the semantic overlap among nodes based on the topics they work on, to further enhance the feature vector and improve overall link prediction performance.

4.2 Related Work

The seminal work of Liben-Nowell and Kleinberg [76] was the first comprehensive study on the utility of topological features derived from graphs for predicting links in social networks. They examine various topological features, including graph shortest distance, common neighbors, preferential attachment, Adamic-Adar, Jaccard, SimRank, hitting time, rooted PageRank, and Katz. They find that topological information is quite useful when compared to a random predictor. In particular, Adamic-Adar and the Katz measure appear to be more effective than the other topological features. Recently, Huang [115] proposes
to use another topological feature – generalized clustering coefficient – to solve the link prediction problem.

An important limitation of these methods is that they only use a single (topological) feature for the link prediction task. Intuitively, it seems that one can achieve better performance by utilizing other sources of information, such as the content or semantic attributes of the nodes. A natural way to do this would be to use the multiple sources of information as features to be fed into a classifier that is trained to discriminate between positive instances (i.e. links that form) and negative instances (links that do not form) by making use of all the features. This is the approach adopted by Hasan et al. [77] and O’Madadhain et al. [78]. Hasan et al. [77] have used topological features (such as the shortest distance between the two nodes), aggregated features (such as the sum of neighbors) and semantic features (such as the number of matching keywords). They report keyword match count to be their most useful feature on one dataset, which indicates that a lot is to be gained by taking into consideration the semantic similarity in the publications of the two authors. O’Madadhain et al. [78] also have investigated the use of content-based attributes such as the KL-divergence of the topic distributions of the two nodes, their geographic proximity, and similarity of journal publication patterns.

The work of Popescul et al. [116] is another interesting approach to integrating different kinds of information. They represent the data in a relational format, generate candidates for features through database join queries, select features using statistical model selection criteria and use Logistic Regression using the selected features for classification. A potential problem with this approach is that the features so generated are simple aggregation functions of the column values in the result set of the join queries (such as count and average).
A more complex feature such as cosine similarity between bag-of-words representations cannot be easily expressed using simple SQL aggregation functions.

Researchers have also examined the use of probabilistic models for solving the link prediction problem. Taskar et al. [117] use discriminatively trained relational Markov networks to define a joint probabilistic model over the entire graph (i.e. over the links as well as the content attributes of the nodes). The trained model is used to collectively classify the test data. Kashima and Abe [118] propose a parameterized probabilistic model of network evolution and then use it for link prediction. They assume the network structure is in a stationary state and propose an EM algorithm to estimate model parameters. They report encouraging results on two small biological datasets. However, both collective classification and training global probabilistic models can be expensive to compute and typically do not scale well to medium and large scale networks.

In related work, Rattigan and Jensen [119] argue that the link prediction problem is too hard to solve because of the extreme class skew problem. Social networks are usually very sparse and positive links only hold a very small amount of all possible pairs of nodes. As an alternative, they propose a simpler problem – anomalous link discovery. Specifically, they constrain their focus on the links that have been formed and infer their anomaly (suprisingness) scores from the previous data.

4.3 Methods

We consider three sources of information of the network data for link prediction. We have a large number of local events that are accumulated along time, where by local event we mean an interaction among a set of objects in the network. For example, the publication of a paper would represent a local event involving all the authors of the paper. We refer to
the collection of such local events as the event log. This is the raw format of the network data and provides the first source of information for link prediction. Such an event log is typically converted to a graph representation, in which nodes represent objects in the network and two nodes are connected to each other if they co-occur in at least one local event. This graph provides the second source of information for link prediction. Finally, we have access to other attributes of an object in the network, such as the research areas of the authors in an author collaboration network, usually referred to as content or semantic information. This semantic information provides the third source of information for link prediction.

It is difficult to capture all information from different sources with one single feature. For this reason, we examine three types of features – co-occurrence probability features, topological features and semantic features - coming from the first, second and third source, respectively. In the text below, we discuss how we derive these features in turn.

### 4.3.1 Deriving Co-occurrence Probability Features

For a pair of nodes that have never co-occurred in the event log, our aim is to estimate the chances of their co-occurring in the future, i.e. of a link forming in the future between those two nodes. In order to estimate the co-occurrence probability of the given two nodes in a principled manner, we use probabilistic graphical models. Specifically, we employ undirected graphical models, also called Markov Random Fields (MRFs), to model the local neighborhood containing the two nodes. We stress the fact that we build a local model, rather than a global model, as building global models can become prohibitively expensive for large scale networks.
There are two main stages in our approach to use graphical models in this context - (a) determining the nodes that will be included in the local model, and (b) using frequent non-derivable itemsets to determine the structure of the graphical model as well as to learn the parameters of the model. Once the model is learned, we use exact inference techniques to determine the co-occurrence probability of the pair of nodes under consideration. We need not resort to approximate inference, as we build a local model, leading to a low treewidth (maximum clique size in the graph formed by triangulating the model subtracted by one) for the model.

**Determining the Central Neighborhood Set of Two Nodes**

For a given pair of nodes, we retrieve a small set of nodes that we believe to be most relevant to estimating the co-occurrence probability of the given pair of nodes. We refer to this set of nodes as the *central neighborhood set* of the two nodes. At one extreme, we can include any node that lies on any path between the two nodes as belonging to the central neighborhood set. However, this would lead to a large probabilistic model, over which learning can be expensive. For this reason, we set a parameter *size* that specifies the number of nodes to be present in the central neighborhood set.

We need a mechanism of selecting the nodes to include in the central neighborhood set. Intuitively, the nodes that lie along paths of shorter length are more relevant. Hence, we propose a method of enumerating simple paths lengthwise, i.e. we first collect all nodes that lie on length-2 simple paths, and then those on length-3 simple paths and so on. (A simple path is a path without cycles.) The algorithm for enumerating all simple paths of a given length is presented in Figure 4.1. However, this order of enumerating nodes may not be enough as there may be many nodes that lie on paths of a given length. Hence, we need a way of ordering paths of the same length. For this purpose, we define the *frequency*
Input: $G$, a graph; $s$, starting node; $t$, ending node; $K$, path length
Output: $\mathcal{P}$, a set of simple paths of length $K$

1: $N = \text{Breadth-first-search}(G, s, K - 1, t)$;
   \hspace{1em} \{ * Find distance-$(K - 1)$ neighbors of $s$ without visiting $t$ and bookkeeping all path information * \}

2: for all $e \in N$ do
3: \hspace{1em} if $(e.\text{Connect}(t, G))$ then \{ * If $e$ and $t$ are connected in $G$ * \}
4: \hspace{2em} add path($s \rightarrow e \rightarrow t$) to $\mathcal{P}$;
5: \hspace{1em} end if
6: end for

7: return $\mathcal{P}$

Figure 4.1: Enumerating all simple paths of length $K$ between two nodes

score of a path as the product of the occurrence counts of all nodes along the paths. Now, among paths of the same length, we enumerate paths with higher frequency scores before paths with lower frequency scores. The pseudo-code of the full algorithm for selecting the central neighborhood set of a pair of nodes is presented in Figure 4.2.

We also use a path length threshold in the algorithm, because in practice, we cannot afford to enumerate all simple paths between two nodes in a large graph. In our study, we consider paths up to length four because we found that this threshold works well in capturing the contextual information between two nodes. In situations where there does not exist such paths between two nodes, we define the central neighborhood set to be the two nodes themselves. Interestingly, we note that in this case, the local probabilistic model reduces to a simple independence model.
Learning Local Markov Random Fields

For a given pair of nodes and their corresponding central neighborhood set, how should one learn a local probabilistic model for it? We adopt an approach of using non-derivable frequent itemsets from the underlying network log events data to learn local probabilistic models. The event log is essentially a transactional dataset and we apply widely-used frequent itemset mining techniques on it to collect occurrence statistics of network objects. These statistics can be leveraged afterward to learn local probabilistic models on the central neighborhood set. The main idea of using a set of itemsets to learn a model is as follows. Each itemset and its occurrence statistic can be viewed as a constraint on the underlying unknown distribution. A model that satisfies all present occurrence constraints and in the meanwhile has the maximum entropy (“as uniform as possible”) is used as the estimate.
of the underlying unknown distribution. This maximum entropy distribution specifies a Markov Random Field.

In our study we pre-compute all frequent itemsets from the underlying log events. Social networks are usually very sparse – the proportion of formed links is very low as opposed to the number of all possible pairs of nodes. For this reason we use a support threshold of one to collect frequent itemset patterns. As a result, all positive occurrence evidence will be captured. We note however, at this support threshold level, frequent itemsets are too many to use. Fortunately, we only need to mine and use non-derivable itemsets for model learning. Simply speaking, non-derivable itemsets are those itemsets whose occurrence statistics can not be inferred from other itemset patterns. As such, non-derivable itemsets provide non-redundant constraints and we can employ them to learn probabilistic models without any information loss. Calders et al. [32] propose an efficient depth-first search method to mine non-derivable itemsets. We use their implementation to mine non-derivable itemsets.

To predict if two nodes will be linked, we first identify from the network the central neighborhood set of the two involved nodes. We then select all itemsets that lie entirely within this set and use them as evidence to learn an MRF. The learned MRF is local in that it specifies a joint distribution over only those nodes in this set. Subsequently, we estimate the joint co-occurrence probability feature of the link through inference over the local model. The formal algorithm is presented in Figure 4.3. We illustrate this whole process with a simple example presented in Figure 4.4. Assume we want to predict the link between nodes $a$ and $b$ and there are two paths connecting them in the graph: $p_1 = a \rightarrow c \rightarrow b$ and $p_2 = a \rightarrow d \rightarrow e \rightarrow b$. Also assume that we use both $p_1$ and $p_2$ to identify the central neighborhood set between $a$ and $b$. As a result, the central neighborhood set $C$ is given
Procedure: Co-occurrence_Probability_Feature_Induction
Input: C, central neighborhood set between s and t;
       s, starting node; t, ending node;
       NDI, a collection of non derivable itemsets
Output: f, co-occurrence probability feature of s and t
1: R ← ∅;
2: for all ndi ∈ NDI do
3:   if ndi ∈ C then {* Retrieve ndi patterns relevant to C *}
4:     add ndi to R;
5: end if
6: end for
7: M = learn_MRF(C, R); {* Learn an MRF on C using R *}
8: f = Inference(M, s, t); {* Infer co-occurrence prob. of s and t from M *}
9: return f

Procedure: Learn_MRF
Input: C, a central neighborhood set;
       R, a collection of itemsets;
Output: MRF M
1: Initialize_Parameters(M); {* Obtain all variables in C and initialize M *}
2: while (Not all constraints are satisfied) do
3:   for all constraint r ∈ R do
4:     Update M to force it to satisfy r;
5:   end for
6: end while
7: return M;

Figure 4.3: Inducing co-occurrence probability feature for a pair of nodes
Figure 4.4: An example of local model-based co-occurrence probability feature induction

by: $C = \{a, b, c, d, e\}$. Next we retrieve all non-derivable itemsets that lie entirely within this set. Let us assume that the itemsets retrieved are: \{a, b, c, d, e, ac, ad, bc, be, de\}. Their occurrence statistics are collected from the log events and are presented in the figure.

We employ all of these patterns and their occurrence statistics to learn a local probabilistic model $M$ over $C$: $M = P(a, b, c, d, e)$. $M$ specifies a joint distribution on all variables in $C$ and its clique potential functions are listed as follows ($\mu$’s are model parameters to be learned and $I()$ is indicator function). The shaded area in Figure 4.4 shows a clique of $M - \{a, c\}$.

$$
\psi_{a,c} = \mu_1 I(a=1) \cdot \mu_2 I(c=1) \cdot \mu_3 I(a=c=1)
$$

$$
\psi_{a,d} = \mu_4 I(d=1) \cdot \mu_5 I(a=d=1)
$$

$$
\psi_{b,c} = \mu_6 I(b=1) \cdot \mu_7 I(b=c=1)
$$

$$
\psi_{b,e} = \mu_8 I(e=1) \cdot \mu_9 I(b=e=1)
$$

$$
\psi_{d,e} = \mu_{10} I(d=e=1)
$$
Then we derive the co-occurrence probability of $a$ and $b$ by computing the marginal probability of $p(a = 1, b = 1)$ on $M$.

We use the iterative scaling algorithm $Learn\_MRF()$ to learn a local MRF for the central neighborhood set. The idea is to iterate over all itemset constraints and repeatedly update the model to force it to satisfy the current itemset constraint, until the model converges. After the model is constructed, we apply an inference procedure over it to estimate the joint co-occurrence probability of $s$ and $t$. For the $Inference()$ procedure in the algorithm, we can plug in exact inference algorithms for it since our model is local. In our study, we use the junction tree inference algorithm.

### 4.3.2 Deriving Topological Features

The Katz measure is a weighted sum of the number of paths in the graph that connect two nodes, with shorter paths being given the more weight. This leads to the following measure:

$$Katz(s, t) = \sum_{i=1}^{\infty} \beta^i p_i$$

Here $p_i$ is the number of paths of length $i$ connecting nodes $s$ and $t$, while $\beta$ is a damping factor. It has been shown that Katz is among the most effective topological measures for the link prediction task [76]. It outperforms shortest distance, hitting time and many others. It can be verified that the matrix of Katz scores can be computed by $(I - \beta M)^{-1} - I$, where $M$ is the adjacency matrix of the graph [76].

However, this method does not scale well to handle the network data under the consideration since computing matrix inverse for large graphs is very expensive. As such, we
come up with a way to approximately compute the Katz score. Specifically, we only consider paths of length up to certain threshold to compute the Katz score. The new measure is as follows:

\[
 aKatz(s, t) = \sum_{i=1}^{k} \beta^i p_i
\]

where \( p_i \) and \( \beta \) have the same meaning as above, while \( k \) is a new input parameter specifying the maximum path length we consider. Since the score terms damp exponentially with the longer length, this new measure captures the most significant portion of the exact Katz score. We find that \( k \) of 4 can give a good approximation of the Katz scores in practice. We design graph algorithms to evaluate this new measure. To this end, we follow the similar process of identifying the central neighborhood set for two nodes shown above, enumerate all simple paths up to length \( k \) from \( s \) to \( t \) and use the above formula to compute an approximate Katz score. We execute breadth-first-search from \( s \) up to \( k \) levels without visiting \( t \), while keeping track of all paths formed so far. We will denote this approximate Katz measure as \( aKatz \) throughout the rest of the chapter.

### 4.3.3 Deriving Semantic Features

The degree of semantic similarity among entities is something that can be useful to predict links that might not be captured by either topological or frequency-based features. For example, in the context of co-authorship networks, we use the following method to compute the semantic similarity for two authors:

1. Collect the words in the titles of each author (removing stop words), so that an author is represented as a set of words (akin to a text document).
2. Derive a bag of words representation for each author, weighting each word by its TFIDF (Term Frequency - Inverse Document Frequency) measure.

3. Compute the cosine between the TFIDF feature vectors of the two authors whose semantic similarity we need to determine.

Previously, Hasan et al. [77] have used keyword match count between two authors as a feature and have reported the feature to be the most useful feature. Our method for computing semantic similarity makes use of the well-known techniques such as TFIDF feature vector representation and the cosine measure to compute similarity - the former can weight words automatically and the latter is a widely-used and an effective measure for computing similarity between text documents represented in the vector space model.

4.3.4 Combining Different Features Using Supervised Learning Framework

Since we have three types of features - the co-occurrence probability feature, the topological similarity feature and the semantic similarity feature - we need an effective way to combine these features. For this we resort to supervised learning. In order to do this, we need to come up with a way to partition the original dataset into training and testing datasets. A supervised learning approach to the link prediction problem has been taken previously by Hasan et al. [77] and Madadhain et al. [78], and the authors have taken different approaches to partitioning the dataset into training and testing sets. We find the approach taken by Madadhain et al. [78] to be cleaner and follow the same, which we describe below. An illustration of our approach can be found in Figure 4.5.

We form a labeled training dataset as follows. We take all the links that are formed in the 9th year (T9 in Figure 4.5) and label them as positive training instances. Of the links
that are not formed in the first 9 years, we randomly sample a subset and label them as negative training instances. We sample 10 times as many negative instances as positive instances. The features for each of these instances are constructed from the first 8 years of data. A classifier is then trained on the labeled training set - any off-the-shelf classifier can be used, we chose to use Logistic Regression\textsuperscript{8}, since it is computationally efficient, and produces well-calibrated class probabilities that can be used to rank predictions.

The testing dataset is formed in a similar fashion: the links that are formed in the 10th year (T10 in Figure 4.5) are treated as testing instances that need to be predicted as positive, and we include a sample of the links that are not formed in the whole of the dataset as testing instances whose ground truth labeling is negative. The features that are used by the classifier trained previously are formed from the first 9 years of data.

4.4 Evaluation

In this section, we report the experimental results of our proposed approach.

\textsuperscript{8}We used the Logistic Regression implementation in the popular WEKA suite of Data Mining algorithms.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of authors</th>
<th>No. of papers</th>
<th>No. of edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>23136</td>
<td>18613</td>
<td>56829</td>
</tr>
<tr>
<td>Genetics</td>
<td>41846</td>
<td>12074</td>
<td>164690</td>
</tr>
<tr>
<td>Biochemistry</td>
<td>50119</td>
<td>16072</td>
<td>191250</td>
</tr>
</tbody>
</table>

Table 4.1: Summary of the three datasets that were constructed

4.4.1 Datasets

We evaluated performance on three real datasets in all, described below. The details of the datasets are summarized in Table 4.1.

- The DBLP dataset was generated using the DBLP collection of Computer Science articles. This dataset contains the publication details of the proceedings of 28 conferences related to Data Mining, Databases and Machine Learning from the years 1997 to 2006.

- The Genetics dataset contains articles published from 1996 to 2005 in 14 journals related to genetics and molecular biology.

- The Biochemistry dataset contains articles published from 1996 to 2005 in 5 journals related to biochemistry.

The Genetics and the Biochemistry datasets were generated from the PubMed database.

---

9 DBLP is located online at http://dblp.uni-trier.de/

4.4.2 Class Conditional Distributions of the Different Features

First we examine the distribution of the features among the positive and negative examples. Figure 4.6a-c plot the distribution for three features on the Genetics dataset. The results on the other two datasets are similar and are not plotted here due to space constraints.

One can see that both the co-occurrence probability and the aKatz measure can discriminate among the negative and positive instances. The main difficulty in the link prediction task occurs because the number of links that do not form far outweighs the number of links that do form. The ratio of negative instances to positive instances in our workload is $10 : 1$.

4.4.3 The Additional Information Captured by Co-occurrence Probability Feature

We believe that the co-occurrence probability feature captures information a large chunk of which is not captured by either topological metrics such as aKatz or content-based metrics such as semantic similarity. To test this conjecture, we examined the number of correct predictions that were made by the co-occurrence probability feature that were not made by either aKatz or semantic similarity. The results are shown in Table 4.2. As can be observed, in all three of the datasets there exists a significant percentage - up to 75% on the Genetics dataset - of correct predictions in the top 500 that are captured only by the co-occurrence probability feature and not by the other features. This confirms our hypothesis that the co-occurrence probability feature uses information about the domain that is not captured by other types of features.

4.4.4 Results on Link Prediction as Classification

We report the classification results when we vary the features used for classification. First, we report the results when the three features – co-occurrence probability, aKatz and
Figure 4.6: Class conditional distributions for different features on the Genetics dataset: (a) Co-occurrence Probability (b) aKatz (c) Semantic Similarity

<table>
<thead>
<tr>
<th></th>
<th>#correct in $c_{500}$</th>
<th>#correct in $c_{500} - k_{500} - s_{500}$ (percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>323</td>
<td>125 (38.7)</td>
</tr>
<tr>
<td>Genetics</td>
<td>425</td>
<td>321 (75.5)</td>
</tr>
<tr>
<td>Biochemistry</td>
<td>474</td>
<td>246 (51.9)</td>
</tr>
</tbody>
</table>

Table 4.2: The number of predictions made by the Co-occurrence probability feature alone that were not made by either aKatz or semantic similarity. $c_{500}$, $k_{500}$ and $s_{500}$ refer respectively to the sets to top 500 predicted links using co-occurrence probability, the set of top 500 predicted links using aKatz and the set of top 500 predicted links using semantic similarity.
semantic similarity – are used in isolation. Then we examine the results when we use all three features. Unless otherwise specified, we use the length threshold of 4 for the aKatz feature and 6 for central neighborhood set size of the local probabilistic model-based co-occurrence probability feature.

**Baseline Approaches**

For the sake of comparison, the results on two baseline approaches – the *Adamic-Adar* measure [120] and *Preferential Attachment* measure [121] are also presented.

The Adamic-Adar measure was originally meant for computing the similarity of two homepages, but has been adapted for computing the similarity between two nodes in a graph by [76]. Let $\Gamma(x)$ be the set of all neighbors of node $x$. Then the similarity between two nodes $x, y$ is given by

$$score(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$$

The intuition behind the score is that instead of simply counting the number of neighbors shared by two nodes, we should weight the hub nodes less and rarer nodes more.

Preferential Attachment is a measure based on a generative model for graphs that has been well received [122]. Based on the generative model that specifies that new nodes are more likely to form edges with nodes that have a large number of neighbors, Newman [121] has proposed the score for two nodes $x, y$ as $score(x, y) = |\Gamma(x)| \cdot |\Gamma(y)|$.

**Evaluation Metrics**

Previous literature has mainly used precision of top-K predictions (K is a user specified parameter - usually the number of true links formed in the testing period) as a metric for
Table 4.3: Link prediction classification results. The K for calculating precision for each dataset is the number of positive instances (i.e. true links) in the test dataset.

<table>
<thead>
<tr>
<th>Feature</th>
<th>DBLP (AUC, Prec. (top-K, K=1500))</th>
<th>Genetics (AUC, Prec. (top-K, K=2912))</th>
<th>Biochemistry (AUC, Prec. (top-K, K=3991))</th>
</tr>
</thead>
<tbody>
<tr>
<td>co-occur. prob. (c)</td>
<td>0.8229 45.80 0.7904 46.77</td>
<td>0.8331 52.59</td>
<td></td>
</tr>
<tr>
<td>aKatz(k)</td>
<td>0.7501 54.87 0.5888 32.21</td>
<td>0.7644 54.93</td>
<td></td>
</tr>
<tr>
<td>semantic(s)</td>
<td>0.7148 35.93 0.5738 16.79</td>
<td>0.6732 20.90</td>
<td></td>
</tr>
<tr>
<td>k+c</td>
<td>0.8665 56.06 0.7904 46.77</td>
<td>0.8526 56.38</td>
<td></td>
</tr>
<tr>
<td>c+k+s</td>
<td>0.8722 57.66 0.7886 47.08 0.8528 56.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adamic-adar</td>
<td>0.6148 31.26 0.4864 15.79 0.5384 18.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prefer. att. (p)</td>
<td>0.7482 36.67 0.7194 35.03</td>
<td>0.8200 51.12</td>
<td></td>
</tr>
<tr>
<td>p+k+s</td>
<td>0.8387 52.53 0.7332 37.36 0.8359 54.92</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Classification results when varying central neighborhood set size on the Genetics dataset

<table>
<thead>
<tr>
<th>Neighborhood Size</th>
<th>AUC</th>
<th>Distance 2 Precision (for top 173)</th>
<th>AUC</th>
<th>Distance 3 Precision (for top 247)</th>
<th>AUC</th>
<th>Distance 4 Precision (for top 382)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.8181 68.78 0.9361 81.78</td>
<td>0.8877 60.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.9932 98.26 0.9855 93.11</td>
<td>0.8854 59.94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.9373 89.01 0.9943 90.28</td>
<td>0.9806 84.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.9621 92.48 0.9942 89.06</td>
<td>0.9819 84.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

evaluation. While this metric has its merits, it has some problems too. Some link prediction methods have relatively high precisions for their top-K predictions when K is small, because they are good at predicting the “easy” links, but the precision drops off dramatically as one increases K. It seems desirable to have an additional evaluation metric that can measure the precision of the classifier without reference to any arbitrary cut-off point. For this reason, we also use AUC (Area Under the ROC Curve) to compare different classifiers, which is a metric that does not need the specification of arbitrary cut-off points and is widely used to evaluate rankings output by a classifier. Huang [123] has previously used this as an evaluation metric in the context of link prediction. An AUC score of 1.0 represents a perfect classifier, and a score of 0.5 is a random classifier. Visually, the closer the ROC curve is to the top left corner of the graph, the better the classifier.
Discussions

Table 4.3 presents the AUC scores and precision of top-K predictions for different features on different datasets. Following [76], the K for the precision metric has been chosen to be the number of true links in the testing period. We also plot the ROC Curves for the different features and the ensemble method considering all the three features on the three datasets in Figures 4.7.

The main point to note is that the co-occurrence probability feature consistently outperforms all other features on AUC scores and performs comparably or better on precision. We discuss the results for each dataset in detail below.

One can see that on the DBLP dataset, the co-occurrence probability feature yields the best AUC score (0.8229) when the three features are used in isolation. The aKatz feature yields the second best AUC score 0.7501 and it has the best precision for K=1500 (however the precision for this feature drops unexpectedly at some point after K=1500, leading to the lower AUC score). The semantic feature is inferior to the previous two features, yielding 0.7148 AUC score and 35.93% precision. We note that all three features outperform the Adamic-Adar measure significantly in terms of both the AUC score and precision. Preferential Attachment outperforms the semantic feature but is worse than the other two features. Furthermore, when we combine the aKatz and the co-occurrence probability features, we improve the AUC score to 0.8665 as well as the precision to 56.06%. We get the best results when we use all three features together – 0.8722 AUC score and 57.66% precision.

The results are even better on the Genetics dataset, where again the co-occurrence probability feature performs significantly better than the other features. This feature alone can give 0.7904 AUC score and 46.77% precision. The other two features do not perform very
well on this dataset when used in isolation, with AUCs dropping to around 0.58 and precisions at 32.21% and 16.79%. When we combine all the three features together, there is not much improvement in the AUC over the co-occurrence probability alone, but there is a slight improvement in the precision to 47.08%. This is because the co-occurrence probability feature has been able to predict a majority of the links that were correctly predicted by the other two features, and predict additional links, leading to not much improvement when using the three features together. Among the baseline methods, Preferential Attachment performs better than aKatz, semantic as well as Adamic-Adar, giving an AUC of 0.71 and a precision of 35.03%.

On the Biochemistry dataset, we again observe the same trend of co-occurrence probability being the most useful feature with an AUC of 0.83 followed by Preferential Attachment with an AUC of 0.82. Precision-wise, aKatz has a slight edge over co-occurrence probability and Preferential Attachment, with aKatz slightly better at 54.9% whereas the latter two have scores of 52.6% and 51.12%. Combining aKatz and co-occurrence probability improves the AUC and the precision to 0.8526 and 56.4%, with additionally combining the semantic similarity giving essentially no improvements. The reason the performance of Preferential Attachment is better on this dataset than Genetics is that the latter is a sparser dataset (it was prepared from 14 journals), which meant that it gave high scores to pairs of prolific authors even though they happened to be in different sub-fields, whereas that would happen less on the Biochemistry dataset.

4.4.5 Results on Varying Central Neighborhood Size

In this section we report the results on varying central neighborhood size for local probabilistic models. We use the Genetics dataset as an example for this set of experiments.
The results on other two datasets are consistent. To better validate the use of contextual information for co-occurrence probability estimation, we divide the positive examples (true links) into different classes by their shortest distance. We first examine the case where true links are formed between nodes with shortest distance 2, followed by shortest distance 3 and so on. For each class of links, we generate correspondingly negative examples using the same ratio (1 to 10). We train a separate classifier for each class. We examine the true links up to distance 4. Specifically on this dataset, we have 173 true links within testing period that are of distance 2, 247 true links of distance 3 and 382 true links of distance 4. Table 4.4 presents the classification results when we use co-occurrence probability feature alone. We vary the size threshold of the central neighborhood set. The larger threshold is, we tend to use more contextual information when estimating the joint probability for a link. Note that threshold 2 is essentially the independence model. From the results, one can see that overall the local probabilistic model-based approach outperforms the independence model by taking into consideration contextual information of pairs of nodes. For the links of distance 2, the AUC score can be improved from 0.8181 to 0.9621 and we can identify up to 51 more true links. For the links of distance 3, the AUC score can be improved from
<table>
<thead>
<tr>
<th>Size Threshold</th>
<th>Time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>6.3</td>
</tr>
<tr>
<td>6</td>
<td>8.6</td>
</tr>
<tr>
<td>8</td>
<td>10.6</td>
</tr>
</tbody>
</table>

Table 4.5: Timing results on co-occurrence probability feature induction

0.9361 to 0.9943 and we can identify up to 28 more true links. Finally for the links of distance 4, the AUC score can be improved from 0.8877 to 0.9819 and we can identify up to 92 more true links. We see that the contextual information does indeed improve predictions.

### 4.4.6 Results on Timing Performance

Now we report the timing performance on co-occurrence probability feature induction when we vary the size of the central neighborhood set. We use the DBLP dataset as an example for this set of experiments. The results on the other two datasets are similar. Table 4.5 presents the average time used to induce the co-occurrence probability feature for one link. As one can see, when we increase the size of the central neighborhood set, it takes more time to compute co-occurrence probability feature. This is expected since the cost of learning a local model is higher as we increase the size of the central neighborhood set. Overall, one can see that inducing the co-occurrence probability feature is computationally efficient.

### 4.5 Conclusions

In this chapter, we have presented our efforts on employing local probabilistic models to glue local patterns together, to tackle the link prediction task. Our efforts produce a simple
yet effective approach of leveraging local probabilistic models for link prediction. Specifically, we use topological structure of the network to identify the central neighborhood set of two nodes, and then learn a local MRF model constrained on non-derivable frequent itemsets from this local neighborhood. We then infer the co-occurrence (link) probability from the resulting model and feed it as a feature into a supervised learning algorithm. We have shown that this co-occurrence feature is quite effective for link prediction on real data. When used in combination with other two types of features – topological and semantic features, we find that the resulting classification performance improves. As future work, we would like to examine and test on additional datasets from other domains. Also, we would like to investigate the use of temporal evolution information of these features in link prediction.
CHAPTER 5

LEARNING APPROXIMATE MARKOV RANDOM FIELDS ON LARGE TRANSACTIONAL DATA

We have presented our techniques of using local probabilistic models to glue local patterns together in the context of solving the link prediction problem. This is a local strategy since we only consider a local data region at one time. However, sometimes we want to be more ambitious – to incorporate more information and learn a probabilistic model over a large data region. In this chapter, we present our work on exploring a global strategy to glue frequent itemset patterns on transactional data.

5.1 Introduction

Pavlov et al. [56] propose a maximum entropy (ME) model based on frequent itemsets to tackle the query selectivity estimation problem on transactional data. The ME model specifies an MRF and is effective in estimating query selectivity. However, a key limitation of their approach is that it needs to learn a local model over query variables on the fly for every query. Due to the fact that inferring an ME model is an expensive iterative process, such a just-in-time model construction approach is not appropriate in settings where online estimation time is crucial. The alternative is to first learn a global model
offline. Subsequently, queries can be answered on the fly using standard probabilistic inference methods [37, 41, 43]. The advantages are a more accurate model (relies on more information from the data) and huge online performance gains. The critical challenge is that a global model may be prohibitive to compute from large high dimensional transactional data. To address this problem, we consider employing frequent itemsets to learn approximate global MRFs on large transactional data. Frequent itemsets capture important distributional information of the data. Hollmen et al. [61] propose to use frequent itemsets to learn mixture models of transactional data at a local scale. They only model a few number of most frequent items. Goldenberg et al. [62] propose an approach of using frequent itemsets to learn large Bayesian networks on transactional data. We conduct an empirical study on real datasets to show the efficiency and effectiveness of our model on solving the query selectivity estimation problem. In certain cases, it can outperform a state-of-the-art solution by up to three orders of magnitude in terms of online estimating time while delivering comparable estimations. The main contributions of this chapter are summarized below:

• We introduce a novel divide-and-conquer style approach based on graph partitioning to learning approximate MRFs on large transactional data. Learning such approximate models is necessary since learning exact MRFs is intractable for such large transactional data.

• We introduce a novel interaction importance and treewidth-based augmentation scheme to capture interdependencies across partitions.

• We conduct an empirical study on real datasets to show the efficiency and effectiveness of our approach.
5.2 Learning Approximate MRFs

Before discussing our proposed approach, let us consider an extreme case in which the whole graphical model consists of a set of disjoint non-correlated components. Then the joint distribution can be derived in a straightforward fashion according to Lemma 3.

Lemma 3 11 Given an undirected graphical model $G$ subdivided into disjoint components $D_1, D_2, \ldots, D_n$ (not necessarily connected components), and there is no edge across any two components, then the probability distribution associated with $G$ is given by: $p(X) = \prod_{i=1}^{n} p(X_{D_i})$

5.2.1 Clustering Variables Based on Graph Partitioning

The main idea of our proposed divide-and-conquer style approach comes directly from the above observation. Specifically, the variables are clustered into groups according to their correlation strengths. We call such a group a variable-cluster. Then a local MRF is inferred on each variable-cluster. In the end we aggregate all the local models to obtain a global model. From Lemma 3, we see that if we have a perfect partitioning of an MRF in which there is no correlations across partitions, the divide-and-conquer style approach produces the exact estimate of the full model. Even for an imperfect partitioning, if the correlations across partitions are not strong, we still expect a reasonable approximation of the full model. Correspondingly, the first problem we face is how to cluster the variables such that the correlations across partitions is minimized.

$k$-MinCut: The $k$-MinCut problem is defined as follows [124]: Given a graph $G = (V, E)$ with $|V| = n$, partition $V$ into $k$ subsets, $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$.

11This follows immediately from the global Markov property of the MRF.
\[ |V_i| = \frac{n}{k}, \] and \[ \cup_i V_i = V, \] and the number of edges of \( E \) whose incident vertices belong to different subsets is minimized. Given a partitioning \( P \), the number of edges whose incident vertices belong to different partitions is called the edge-cut of the partitioning. In the case of weighted graphs, we minimize the sum of weights of all edges across partitions.

The \( k \)-MinCut can serve our purpose of clustering variables. Each graph partition corresponds to a variable-cluster. Intuitively, we want to maximize correlations among variables within variable-clusters, and minimize correlations among variables across variable-clusters. To accomplish this we ensure that the weight of edges reflect the strength of correlations between variables. We have the collection of all frequent itemsets. In particular, 2-itemsets specify the connectedness structure of the model graph, and their associated supports indicate the strength of pairwise correlations between variables. We can use their supports as the edge weights directly. However, we also have higher-order statistics available, i.e., the larger itemsets. Our hypothesis is that taking into consideration all the itemsets will yield a better weighting scheme. To this end, we propose an accumulative weighting scheme as follows: for each itemset, we add its support to all related edges, whose two vertices are contained by the itemset. Intuitively, we tend to strengthen the graph regions that involve closely related itemsets in the hope that the edges within these regions will not be broken easily in the partitioning. Figure 5.1 illustrates the weighting scheme using a simple example. The collection of frequent itemsets and their supports are given in the figure.

An advantage of the \( k \)-MinCut partitioning scheme is that the resulting clustering is forced to be balanced. This is desirable for the sake of efficient model learning, since we will not encounter very large variable-clusters which can result in very complex local models. The \( k \)-MinCut partitioning scheme yields disjoint partitions. However, there exist
edges across partitions. In other words, different partitions are correlated to each other. So how do we account for the correlations across partitions?

5.2.2 Interaction Importance and Treewidth-Based Variable-Cluster Augmentation

The variable-clusters produced by the $k$-MinCut partitioning scheme are disjoint. However, there can be correlation information that is lost during the partitioning. To compensate for this loss, we propose an interaction importance-based variable-cluster augmenting scheme. The idea is that we allow each variable-cluster to grow outward. More specifically, it attracts and absorbs most important interactions (edges) incident to its vertices from outside to itself. As a result, some extra variables are pulled into the variable-cluster. We control the augmentation through the number of extra vertices pulled into the cluster (called growth factor). One can use the same growth factor for all variable-clusters to preserve their balance.

As an optimization, we account for the model complexity during the augmentation. We keep augmenting a partition until its complexity reaches a user-specified threshold.
More specifically, we keep track of the growth of the treewidth during the augmentation. 1-hop neighboring vertices are first considered for the augmentation, followed by 2-hop neighboring vertices and so on. Meanwhile, we still follow the interaction importance criteria. The resultant augmented partitions are likely to become unbalanced in terms of their size. The partitions with a small treewidth will grow more significantly than those with a large treewidth. However, these partitions are balanced in terms of their complexity. A benefit of this optimization is that more interactions across partitions will be accounted for in a computationally controllable manner, leading to a more accurate global model. After the augmentation, we obtain overlapped variable-clusters. Figure 5.2 presents a sketch of the augmented variable-clusters.

5.2.3 Approximate Global MRFs and A Greedy Inference Algorithm

For each augmented variable-cluster, we collect all of its related itemsets and use the iterative scaling algorithm to learn an exact local model. This is computationally feasible since the local model corresponding to each variable-cluster is much simpler than the original model. Two local models are correlated to each other if they share variables. The
Input: $F$, a collection of frequent itemsets; 
   $k$, number of partitions for MinCut partitioning; 
   $g$, growth factor; 
Output: $\mathcal{M}$, global MRF; 
1: Construct a weighted graph $G$ from $F$; 
   \{* $G$ specifies graphical structure of the exact MRF; *\} 
2: $k$-MinCut $G$; 
3: for all graph partition $G_i$ do 
4: \hspace{1em} $G'_i \leftarrow \text{augment}(G_i, g)$; 
5: \hspace{1em} Select itemsets $F_i$ related to $G'_i$ 
6: \hspace{1em} $M_i \leftarrow \text{LearnLocalMRF}(F_i)$; 
7: \hspace{1em} add $M_i$ to $\mathcal{M}$; 
8: end for 
9: return $\mathcal{M}$; 

Figure 5.3: Learning approximate global MRF algorithm 

aggregation of all the local models together, delivers a global model of the original trans-
actional data. We note that this global model is an approximation of the exact global MRF, 
since we lose dependency information by breaking edges in the exact graphical model. 
However, most strong correlations are compensated for during the variable-cluster aug-
mentation. As such, we believe that the proposed global model reasonably approximates 
the exact model. Figure 5.3 provides the formal algorithm for learning an approximate 
global MRF. 

Given the global model consisting of a set of local MRFs, how do we make inferences 
on this model efficiently? In the first case, where all query variables are subsumed by 
a single local MRF, we just need to calculate the marginal probability within the local 
model. In the second case, where query variables span multiple local models, we use 
a greedy decomposition scheme to compute. First, we select the local model that has
the largest intersection with the current query (i.e., covers most query variables). Then we select the next local model that covers most uncovered variables in the query. This covering process will be repeated until we cover all query variables. Simultaneously, all intersections between the above local models and the query are recorded. In the end, we derive an overlapped decomposition of the query. We notice that locally the dependency among small pieces in the decomposition often exhibits a tree-like structure, and we use Lemma 4 to compute the marginal probabilities.

**Lemma 4** Given an undirected graphical model $G$ subdivided into $n$ overlapped components, if there exists an enumeration of these $n$ components, i.e., $C_1, C_2, \ldots, C_n$, s.t., for any $2 \leq i \leq n$, the separating set, $s(C_i, \bigcup_{j=1}^{i-1} C_j) \subseteq (C_i \cap (\bigcup_{j=1}^{i-1} C_j))$, then the probability distribution associated with $G$ is given by: $p(X) = \frac{\prod_{i=1}^{n} p(X_{C_i})}{\prod_{i=2}^{n} p(X_{C_i} \cap (\bigcup_{j=1}^{i-1} X_{C_j}))}$

**Proof:** We follow the order $C_1, C_2, \ldots, C_n$ to deduce the full joint distribution as follows (repeatedly apply Lemma 5):

\[
p(X_{C_1} \cup X_{C_2}) = \frac{p(X_{C_2}) \cdot p(X_{C_1})}{p(X_{C_2} \cap X_{C_1})}
\]

\[
p(X_{C_1} \cup X_{C_2} \cup X_{C_3}) = p(X_{C_1} \cup X_{C_2}) \cdot \frac{p(X_{C_3})}{p(X_{C_3} \cap (X_{C_1} \cup X_{C_2}))}
\]

\[\vdots\]

\[
p(X_{C_1} \cup \ldots \cup X_{C_n}) = \frac{\prod_{i=1}^{n} p(X_{C_i})}{\prod_{i=2}^{n} p(X_{C_i} \cap (\bigcup_{j=1}^{i-1} X_{C_j}))}
\]
Lemma 5  Let $X_1, X_2, X_3$ be three disjoint sets of variables in an undirected graph $G$, such that $X = X_1 \cup X_2 \cup X_3$. Additionally, there is no edge across $X_1$ and $X_2$ (we only allow edges across $X_1$ and $X_3$, $X_2$ and $X_3$ (see Figure 5.4), i.e., the separating set for $X_1$ and $X_2$, $s(X_1, X_2) \subseteq X_3$, then the probability distribution associated with $G$ is given by:

$$p(X) = \frac{p(X_1, X_3) \cdot p(X_2, X_3)}{p(X_3)}$$

Proof:

$$p(X_1, X_2, X_3) = p(X_3) \cdot p(X_1|X_3) \cdot p(X_2|X_1, X_3)$$

($X_1$ and $X_2$ are independent given $X_3$)

$$= p(X_3) \cdot p(X_1|X_3) \cdot p(X_2|X_3)$$

$$= \frac{(p(X_3) \cdot p(X_1|X_3)) \cdot (p(X_2|X_3) \cdot p(X_3))}{p(X_3)}$$

$$= \frac{p(X_1, X_3) \cdot p(X_2, X_3)}{p(X_3)}$$

□

To use the above formula, we require that there is no cyclic dependency among components. The overall dependency among components has an exact tree-like structure. Essentially, Lemma 4 specifies a junction tree-like structure. Given any model and one of its such decomposition, we can use the above formula to make exact inferences. However, it is possible to have cyclic dependencies among the decomposed pieces. Therefore, the greedy inference scheme is a heuristic. Also, we note that our global model is not globally consistent in the sense that there can exist inconsistencies across the local models, i.e., for the same set of query variables, different local models can yield different inference results.
However, we expect that the global model is nearly consistent since two correlated local models support the same evidence (itemsets) regarding their shared variables.

5.3 Experimental Results

In this section, we examine the performance of our proposed model for the query selectivity estimation problem on real transactional datasets. The query selectivity estimation for transactional data can be defined as follows.

**Definition 7 (Query Selectivity Estimation).** Let $D$ be a transactional dataset $\{t_1, t_2, \ldots, t_n\}$ and $I$ be the set of all items in $D$. A transaction $t_i$ satisfies a conjunctive query $Q$ if and only if it has equal values as $Q$ on all items in $Q$. Estimating $Q$’s selectivity is to estimate $|D_Q| = |\{t_i| t_i \text{ satisfies } Q\}|$.

Let us look at a concrete example. Consider a conjunctive query $Q = A \cap \bar{B} \cap C$ that specifies $A$ and $C$ occur while $B$ does not occur. Clearly, transaction $\{A, C, D, E\}$ satisfies $Q$. However, transaction $\{A, B, C, E\}$ does not satisfy $Q$. Estimating $Q$’s selectivity is to estimate the number of transactions in the dataset that satisfy $Q$.

We compare our proposed model against Pavlov et al.’s approach [56] where a local MRF over query variables is learned on the fly for every query, referred to as the online local MRF approach (abbreviated as $OLM$).

5.3.1 Experimental Setup

All the experiments are conducted on a Pentium 4 2.66GHz machine with 1GB RAM running Linux 2.6.8. The MRF learning algorithm is implemented in C++. We use Apriori [8] to collect frequent itemsets and Metis [124] to obtain a $k$-MinCut of the exact graphical model.
Datasets: We use two publicly available datasets: the Microsoft Anonymous Web dataset\(^{12}\) with 32711 transactions and 294 items; the BMS-Webview1 dataset\(^{13}\) with 59602 transactions and 497 items.

Query Workloads: We consider the workloads consisting of conjunctive queries (following the same practice in [56]) of different sizes. We first specify the number of query variables \(n\), then we select \(n\) variables according to the probability of the variable taking a value of “1” and generate a value for each selected variable by its univariate probability distribution. As pointed out in [56], the variables are prone to take the zero values in sparse data, thus using purely random queries (variables are randomly chosen with randomly chosen values) would result in a preponderance of queries with zero count.

Performance Metrics: We consider the online time cost, the time taken to answer the queries using the model. We also consider the offline time cost, the time taken to learn the model. We quantify the accuracy of estimations using the average absolute relative error over all queries in the workload. The absolute relative error is defined as \(\frac{|\sigma - \hat{\sigma}|}{\sigma}\), where \(\sigma\) is the true selectivity and \(\hat{\sigma}\) is its estimation.

In the experiments, we vary \(k\), the number of clusters; \(g\), the number of vertices used to augment variable-clusters (the larger \(g\) is, the more overlapped the variable-clusters are, in the special case where \(g = 0\), the variable-clusters are disjoint); the treewidth threshold \(tw\) when the treewidth-based augmentation optimization is used and finally query size (4, 6, 8, 10, 12).

\(^{12}\)http://kdd.ics.uci.edu
\(^{13}\)fimi.cs.helsinki.fi
5.3.2 Results on the Microsoft Web Dataset

In this section, we report the experimental results on the Microsoft Web Dataset. We use the support threshold of 20 to collect the frequent itemsets, which results in 9901 frequent itemsets. According to the Maximum Cardinality Search (MCS)-ordering heuristic [40], the treewidth of the resulting MRF is 28, for which learning the exact model is considered intractable.

Figure 5.5 presents the estimation accuracy when $k$ is varied ($g = 5$) for queries of different sizes. As one can see, our approach produces very close or even better estimations compared with OLM. These results are not surprising since for OLM, we only use the local information to estimate the selectivity. However, for our model, we rely on the global information to make the estimation. An obvious trend that stands out is that as the query size increases, the quality of the estimations degrades. This is expected, since for larger sized queries, estimation errors grow for both approaches. Another observation is that the
Figure 5.6: Varying $k$ ($g = 5$): online time

Figure 5.7: Varying $k$ ($g = 5$): offline time
estimations are more accurate when we use fewer variable-clusters. This is because with fewer variable-clusters, the information loss due to the graph partitioning is smaller, thus we better capture the correlations between partitions. Figure 5.6 illustrates how the online times depend on $k$. It can be clearly seen the significant growth of the online times taken by OLM (note the Y-axis scale). The extreme online timing efficiency of our model can be clearly seen from the results. In most cases, it outperforms OLM by two to three orders of magnitude. Furthermore, we see that the smaller $k$ results in higher online estimating time. This is because the smaller $k$ results in more complex local models. In the extreme case where $k$ is 1, we revert to learning the exact global MRF, which has been shown to be computationally infeasible. Figure 5.7 presents the offline learning times of our model when varying $k$. An obvious trend is that as we increase $k$, overall the learning cost of our model decreases significantly. This is because the larger $k$ results in less complex local models.
Figure 5.9: Varying $g$ ($k = 20$): online time

Figure 5.10: Varying $g$ ($k = 20$): offline time
Figure 5.11: Varying \( tw (k = 25) \): estimation accuracy

Figure 5.8 presents the estimation accuracy when varying \( g \) (\( k \) is fixed as 20). As one can see, the error decreases steadily when increasing \( g \). When \( g \) is 0 (disjoint variable-clusters), the estimations are most inaccurate. In contrast, the estimations are much more accurate when \( g \) is 5. The results clearly show the effects of the interaction importance based variable-cluster augmenting scheme. Our model approximates the exact model better when more correlations across the local models are compensated for. Figure 5.9 presents the online times when varying \( g \). We see from the results that the model with the larger \( g \) takes more online time to answer the query. This is also expected, since the larger \( g \) results in more complex models (similar to the case of the smaller \( k \)). Figure 5.10 presents the offline learning times of our model when varying \( g \). An obvious trend is that as we increase \( g \), the time cost increases significantly, which is again expected.

Figure 5.11, Figure 5.12 and Figure 5.13 present the estimation accuracy, the online estimating times and the offline learning times of our model when the treewidth-based
Figure 5.12: Varying $tw$ ($k = 25$): online time

Figure 5.13: Varying $tw$ ($k = 25$): offline time
augmentation optimization is used ($k$ is fixed as 25). As one can see, the optimization can further improve the estimation performance. For example, the average relative estimation errors are 0.29%, 0.97%, 2.01%, 3.66% and 4.81% on the workloads consisting of queries of size 4, 6, 8, 10 and 12, respectively. In contrast, the corresponding errors of OLM are 0.99%, 2.76%, 4.45%, 7.82% and 10.9%, respectively.

5.3.3 Results on the BMS-Webview1 Dataset

In this section, we report the experimental results on the BMS-Webview1 dataset. We use the support threshold of 50 to collect the frequent itemsets, which results in 8191 frequent itemsets. The treewidth of the resulting exact MRF is 44 according to the MCS heuristic, which also makes learning the exact model intractable. The results on varying $k$ and $g$ are similar to that on the Microsoft Web dataset and are thus omitted. We report the results when the treewidth-based variable-cluster augmentation scheme is used.
Figure 5.15: Varying $tw$ ($k = 25$): online time

Figure 5.16: Varying $tw$ ($k = 25$): offline time
Figure 5.14, Figure 5.15 and Figure 5.16 present the estimation accuracy, the online times and the offline learning times of our model when the treewidth-based augmentation optimization is used ($k$ is fixed as 70). As one can see, our model is able to achieve estimations close to OLM, when the treewidth threshold is 10. When we increase the threshold to 12, our model yields more accurate estimations. Furthermore, our model provides better online timing performance than OLM, though the difference is not as significant as that on the Microsoft Web data. The reason is that the BMS-Webview1 data contains much more items than the Microsoft Web data. As a result, the random queries generated are more likely to contain more uncorrelated items. As a result, we have to use more local MRFs to cover one query when we estimate its selectivity, slowing down the estimation. In contrast, learning an online local MRF becomes easier in this case. However, if we consider correlations between items when we generate random workloads, in other words, more correlated items are more likely to occur in the same query, we expect that our model will be significantly faster.

\section*{5.4 Related Work}

Pavlov et al. [56, 60] have done significant work on exploiting probabilistic models for query selectivity estimation on transactional data. They examine several models for this purpose. Besides the online local MRF approach, they have also examined the Chow-Liu tree model, the mixture Bernoulli model, the inclusion-exclusion AD-Tree model and the online Bayesian network model.

- The Chow-Liu tree model assumes that there are only pairwise dependencies between the variables. Furthermore, the dependency graph on the variables has a tree structure. To fit such a model, it suffices to know the pairwise joint probabilities of all the
variables (2-itemsets). The optimal tree model can be found by applying Kruskal’s algorithm [125] to find the minimum spanning tree of the full graph whose nodes are the variables and the edge weights are the mutual information values between the variables.

- The mixture Bernoulli model can be thought of as a generative model of the data, i.e., a procedure for generating the data. For this model, we assume that the data comes from $N_c$ different clusters. A data point is assumed to be produced by first selecting one of $C$ clusters, where the probability of selecting cluster $C_i$ equals $P(C_i)$, for $i = 1, 2, \ldots, N_c$, and satisfies $\sum_{i=1}^{N_c} P(C_i) = 1$. Once cluster $i$ is selected, a vector of variable values is sampled from the probability distribution associated with that cluster. We assume that in each cluster the variables are conditionally independent of each other give the cluster and follow the Bernoulli distribution. The model parameters can be fitted by using the Expectation-Maximization(EM) algorithm [126].

- In the inclusion-exclusion AD-Tree model, itemsets and their frequencies are stored in a data structure called an AD-Tree that supports an efficient implementation of the inclusion-exclusion principle in order to answer the query.

- In the online Bayesian network model, a local Bayesian network is learned on the fly for every query, similar to the online MRF model. They use a Bayesian score derived in [127] to evaluate the quality of a network structure containing decision graph. Then they use a greedy search algorithm to search the space of possible structures with the objective of maximization of the Bayesian score.

They show that among all of these approaches, the online local MRF approach yields most accurate selectivity estimations on sparse data.
Goldenberg et al. [62] propose the screen-based Bayes net structure search algorithm (SBNS), an approach of using frequent itemsets to learn large Bayesian networks. The main idea is to take a $k$-itemset as a $k$-way statistic, and exploit it fully for a local Bayesian structure. A complete search is carried out since this is affordable at this local scale. All itemsets are screened in a way that the redundant information gets pruned. Then they rely on a simple heuristic that iterates over potential edges once, efficiently exploiting locally collected statistics and structures to create a global Bayesian network. It is not necessary to make simplifying assumption such as restricting the number of possible parents and thus impacting the structure of the networks, since the local structural searches are very efficient. Furthermore, they augment the learned Bayesian networks with edges of high mutual information for variables that have not co-occurred in the data, since such dependencies are not captured by the frequent itemsets. They have shown that SBNS outperforms another scalable alternative – random hill-climbing in terms of the quality of the learned model.

Hollmen et al. [61] propose a mixture maximum entropy model for transactional data. The transactional data is first clustered using the EM clustering algorithm. Then for each data chunk, the frequent itemset patterns are identified and are used to build a probabilistic model based on the maximum entropy principle. They combine the models of all the data chunks together to deliver a global mixture model of the data. They show that this mixture model yields better quality in terms of explaining the observed data, than the original model without clustering the data. However, their modeling effort is at a local scale – they only model a few number of most frequent items (variables).
5.5 Conclusions

In this chapter, we have presented our efforts on employing a global strategy to glue local itemset patterns together for data analysis. To this end, we propose a new approach of leveraging frequent itemsets to learn approximate MRFs on large transactional data. Our proposed approach has been shown to be very effective and efficient in solving the query selectivity estimation problem on such data. In the future, we would like to exploit a belief propagation style approach to force the consistency of the model.
CHAPTER 6

MINING NON-REDUNDANT STRUCTURAL TREE PATTERNS
ON XML DATA

Mining frequent tree patterns is an important research problem with board applications in bioinformatics, digital library, e-commerce and so on. There exist efficient mining algorithms that can mine frequent tree patterns from a tree or graph dataset. Similar to the itemset setting, the number of mined tree patterns can be so large that using these patterns becomes not easy. In this chapter, we discuss our efforts on mining non-redundant tree patterns on large XML data and leveraging on these non-redundant patterns to model the XML data and solve the XML twig selectivity estimation problem.

6.1 Introduction

XML is gaining acceptance as a standard for data representation and exchange over the World Wide Web. However, for wide-spread deployment and use it is becoming increasingly clear that the design of an efficient high-level querying mechanism is necessary. Since XML documents may be represented as a rooted and labeled tree, this necessity has led to the development of tree-based (twig) querying mechanisms. Twig queries describe a complex traversal of the document graph and retrieve document elements through an intertwined (i.e., joint) evaluation of multiple path expressions.
Given the importance of twig queries as a basic selection mechanism in XML [95, 97, 98], efficient support for accurately estimating their selectivity is crucial for the optimization of complex queries. This is analogous to selectivity estimation in relational databases [3–6]. Accurate selectivity estimation is also desirable in interactive settings and for approximate queries. For instance, an end-user can interactively refine their query if they know it will return an overwhelmingly large result set. Similarly, the estimated value can be returned as an approximate answer to aggregate queries using the COUNT primitive.

The early work in this area has focused on determining the selectivity of path expressions (a special case of twig queries) [88–93]. The Lore system [88] adopts a Markov model-based approach for this purpose. The Markov table method [90] improves on the Lore system through the use of intelligent pruning and aggregation to reduce space requirements. Recently, Lim and Wang proposed XPathLearner [89], an on-line, tunable Markov table method which has been shown to be effective for path expression selectivity. A key limitation of these methods is that they do not adapt well to twig queries because they do not account for path correlations.

More recently, researchers have focused on selectivity estimation for twig queries [94–98]. Examples include Correlated Sub-Trees [95], XSketches [96, 98] and Tree-Sketches [97]. Among these it has been shown that TreeSketches is the most accurate and efficient method [97]. TreeSketches [97], a successor of XSketches, clusters the similar fragments of XML data together to generate its synopsis. The granularity of the clustering depends on the memory budget.

To estimate the selectivity of XML twig queries, the above approaches, as well as the approach presented in this chapter, define a summary data structure that houses important statistics about the data from which the selectivity may be estimated. Important issues at
hand include: the quality of estimation from the given summary; the time to construct the summary; and finally, the time to estimate the selectivity of queries from the summary. To address these issues we present a new approach to selectivity estimation. The key contributions of our approach are highlighted below.

First, we present a framework under which the selectivity of a query (represented as a rooted tree) can be estimated from its subtrees. We present and evaluate different strategies for decomposing the query into subtrees. These subtrees can then be used to arrive at a selectivity estimate. We present a theoretical basis for this approach and furthermore show that it subsumes the Markov model-based XML path selectivity estimation as a special case.

Second, to summarize an XML dataset we leverage the use of frequent tree mining. A dynamically-determined subset\(^\text{14}\) of all the discovered subtrees up to a certain size (number of nodes), coupled with associated occurrence statistics, forms the basis of our summary structure. More specifically, the dynamic subset we store is based on the notion of (non)-derivable patterns. We also rely on fast searching mechanisms to locate the subtrees of a given twig query within our summary structure.

Third, we conducted an extensive experimental study to examine the benefits of our approach and compare it against TreeSketches\(^\text{15}\). Empirical results show that our approach takes less time to construct the summary, and is usually much faster when computing the selectivity estimates. In our qualitative assessment we also find that our approach compares favorably with TreeSketches. We also offer a detailed explanation as to why the new approach (called \textit{TreeLattice}) outperforms TreeSketches [97] under certain conditions.

\(^{14}\)Due to storage costs, the complete lattice (all frequent patterns) cannot be held in memory, thus we only store a portion of it, which is dynamic and data dependent.

\(^{15}\)We are grateful to Neoklis Polyzotis for providing us with the TreeSketches executable and also for helping us tune the algorithm for a fair comparison.
The rest of the chapter is organized as follows. We formally define our problem and give an overview of TreeLattice in Section 2. In Section 3, we detail our proposed summary structure and twig decomposition-based XML twig selectivity estimation framework. We present experimental results in Section 4 and related work in Section 5. Finally we discuss the future work and conclude in Section 6.

6.2 Problem Definition and TreeLattice Overview

In the following section, we formally define the problem of estimating XML twig selectivity (Subsection 6.2.1). We follow with a discussion of the basic ideas and key challenges in our new approach, TreeLattice (Subsection 6.2.2).

6.2.1 Problem Definition

An XML document can be structurally modeled as a tree where each node is typically associated with a tag or a value. In practice, values are almost always associated with leaf nodes and tags with interior nodes. As with prior work by Polyzotis and Garofalakis [128], we do not model value elements.

A twig query $T_Q$ is defined as a node-labeled tree $T_Q(V_Q, E_Q)$, where each node $t_i \in V_Q$ is labeled with a path expression $P_i$. At an abstract level, each node $t_i$ corresponds to a subset of elements, while the path $P_i$ describes the structural relationship that must be satisfied between the elements in $t_i$ and the elements in its parent node. In particular, we only consider the parent/child relationship between different elements. Research on the more general ancestor/descendant relationship is underway. We next present the definition of a twig match as given by Chen et al. [95].
**Definition 8** A match of a twig query $T_Q = (V_Q, E_Q)$ in a node-labeled data tree $T = (V_T, E_T)$ is defined by a 1 − 1 mapping: $f : V_Q \mapsto V_T$ such that if $f(u) = v$ for $u \in V_Q$ and $v \in V_T$, then (i) $\text{Label}(u) = \text{Label}(v)$ and (ii) if $(u, u') \in E_Q$, then $(f(u), f(u')) \in E_T$.

The selectivity $\sigma(T_Q)$ of twig query $T_Q$ is defined as the number of matches of $T_Q$ in the data tree. Our objective is to accurately estimate the selectivity of an XML twig query $T_Q$ as efficiently as possible given constraints in space (summary storage) and time (summary construction and estimation time).

### 6.2.2 Basic Ideas and Key Challenges of TreeLattice

The first basic idea in TreeLattice comes from the observation that in many cases, the selectivity of a given twig query $\sigma(T_Q)$ can be reasonably estimated from the selectivity information of its sub-twig queries. For example, suppose twig $T_Q$ is the union of two sub-twigs $T_1$ and $T_2$, which differ by only one edge and share a common part $T$ (Figure 6.1a). We can expect $\sigma(T_1)$, $\sigma(T_2)$ and $\sigma(T)$ to provide good clues for estimating $\sigma(T_Q)$ in many real datasets. Furthermore, if the twig $T_Q$ is the union of a set of sub-twigs, the selectivity of all these sub-twigs can be used to estimate $\sigma(T_Q)$. Therefore, the first problem we face is *can we develop a reasonable selectivity estimate for a given twig query $T_Q$ by utilizing the selectivity of its sub-twigs?* This problem is answered in Subsection 6.3.1, where we construct such an estimator based on the *conditional independence assumption for growing a tree*. Note that in order to systematically estimate the selectivity of twig queries with this approach, we need to pre-compute a group of small twigs as the basis.

However, we can also expect that our assumption will likely be violated for some twig queries on a given XML dataset. To deal with this issue, we use another basic idea from
the observation that the selectivity information of different twigs can be of differing importance in terms of capturing the underlying twig distribution. For example, if the selectivity of $T_1$, $T_2$, and $T$ are available and $T_Q$ can be precisely estimated from them, then the selectivity of $T_Q$ should not be pre-computed. Here, we face another key challenge in TreeLattice: how can we select a group of twigs as the basis for selectivity estimation in order to minimize the estimation error? In particular, such selection needs to be performed under the budget of user-defined memory cost. Another problem closely related to this challenge is how can we decompose a large twig query into basic twigs and perform estimations if different decompositions exist? The solution to the latter actually helps us determine a solution for the former. In Subsection 6.3.2, we discuss the decomposition problem and in Subsection 6.3.3, we introduce our method to select basis for selectivity estimation.

Given the above discussion, we can see that our TreeLattice has three basic components: Basis Building, Twig Decomposition, Augmenting Estimation. The basis building is off-line and the other two components are computed at runtime while processing a query. When a new query arrives, we first decompose it into the small twigs in the basis and use the pre-computed selectivity of these basic twigs to infer the selectivity of the complex (larger) one.

### 6.3 An Estimation Framework based on Twig Decomposition

In this section, we will answer the three questions posed in the previous section. The twig decomposition-based selectivity estimation framework will be formulated during the course of this discussion.
Suppose we have two basic twigs $T_1$ and $T_2$, and they differ by only one edge (Figure 6.1(a)). If $T$ is common to both, then we can express $T_1$ as $T \cup \{e_1\}$ and $T_2$ as $T \cup \{e_2\}$, where $e_1$ and $e_2$ are two distinct edges. The edges are distinct in that they either attach to different nodes of $T$, or the two additional nodes $x$ and $y$ introduced by these two edges are different. The two twigs can be augmented together to generate a larger twig, denoted as $T_1 \cup T_2 = T \cup \{e_1\} \cup \{e_2\}$. Assuming the counts of $T_1$, $T_2$ and their common part $T$ are available and denoted as $\sigma(T_1)$, $\sigma(T_2)$ and $\sigma(T)$, respectively, we are interested in estimating the count of the augmented twig, $T_1 \cup T_2$, based on this information.

A complication arises when the occurrence of $T$ is coupled with one or more instances of edge $e_1$, as shown in Figure 6.1(b). Let $T_1^i$ denote the occurrence of $T$ with $i$ edges of type $e_1$. Then it is easy to see that the selectivity of $T_1$ is given by the decomposition formula\textsuperscript{16}:

$$\sigma(T_1) = \sigma(T_1^1) + 2 \times \sigma(T_1^2) + \cdots + n \times \sigma(T_1^n)$$

\textsuperscript{16}The coefficients in front of each term represents the number of choices one has to grow from $T$ to $T_1$. $n$ is the maximal number of $e_1$ edges under $T$. Similarly, $m$ is the maximal number of $e_2$ edges under $T$. 

**Figure 6.1**: (a) Augmented twigs $T_1 \cup T_2$; (b) Growing $T_1$ from $T$
and similarly the selectivity of $T_2$ is given by:

$$\sigma(T_2) = \sigma(T_2^1) + 2 \times \sigma(T_2^2) + \cdots m \times \sigma(T_2^m)$$

In order to derive our formula for estimating the augmented twig $T_1 \cup T_2$, we assume that the event of growing $T_1$ from $T$ is conditionally independent from the event of growing $T_2$ from $T$ (called the tree-growing independence assumption). More formally we have:

$$Pr(T_1^i \cup T_2^j|T) = Pr(T_1^i|T) \times Pr(T_2^j|T)$$

where:

$$Pr(T_1^i|T) = \sigma(T_1^i)/\sigma(T)$$

and:

$$Pr(T_2^i|T) = \sigma(T_2^i)/\sigma(T)$$

**Lemma 6** Given two non-trivial rooted and labeled twigs $T_1$ and $T_2$, which differ by only one edge, let $T$ be the common part between $T_1$ and $T_2$. Under the tree-growing independence assumption, the expected count of $T_1 \cup T_2$ is given by $\sigma(T_1) \times \sigma(T_2)/\sigma(T)$.

**Proof:** Given the tree-growing independence assumption, we can treat the count of $T_1 \cup T_2$ as a random variable. The expected value of this random variable, $E(\sigma(T_1 \cup T_2))$, is as follows:

$$(\text{From the decomposition formula})$$

$$E(\sigma(T_1 \cup T_2)) = \sum_{i=1}^{n} \sum_{j=1}^{m} E(\sigma(T_1^i \cup T_2^j))$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} (i \times j \times Pr(T_1^i \cup T_2^j|T) \times \sigma(T))$$
(By the conditional independence assumption)

\[= \sum_{i=1}^{n} \sum_{j=1}^{m} i \times j \times Pr(T_i | T) \times Pr(T_j | T) \times \sigma(T)\]

\[= \sigma(T) \times \sum_{i=1}^{n} i \times Pr(T_i | T) \times (\sum_{j=1}^{m} j \times Pr(T_j | T))\]

\[= \sigma(T) \times \sum_{i=1}^{n} i \times Pr(T_i | T) \times (\sum_{j=1}^{m} j \times \frac{\sigma(T_j)}{\sigma(T)})\]

\[= \sigma(T) \times \sum_{i=1}^{n} i \times Pr(T_i | T) \times \frac{1}{\sigma(T)} \times (\sum_{j=1}^{m} j \times \sigma(T_j))\]

(The decomposition of count of \(T_2\), \(\sigma(T_2)\))

\[= \sigma(T) \times \sum_{i=1}^{n} i \times Pr(T_i | T) \times \frac{\sigma(T_2)}{\sigma(T)}\]

(The decomposition of count of \(T_1\), \(\sigma(T_1)\))

\[= \sigma(T) \times \frac{\sigma(T_1)}{\sigma(T)} \times \frac{\sigma(T_2)}{\sigma(T)}\]

\[= \sigma(T_1) \times \frac{\sigma(T_2)}{\sigma(T)}\]

□

In our approach, we will use the expected count of \(T_1 \cup T_2\) as the estimate of the true count of \(T_1 \cup T_2\), denoted as \(\hat{\sigma}(T_1 \cup T_2) = \sigma(T_1) \times \sigma(T_2) / \sigma(T)\).

An important lemma that follows from Lemma 6 is stated next.

**Lemma 7** Given two subtrees \(T_1\) and \(T_2\) that share a common subtree \(T\), where

\[|T| = \text{min}(|T_1|, |T_2|) - 1\]

then \(\sigma(T_1 \cup T_2)\) can be estimated as follows:

\[\sigma(T_1 \cup T_2) = \frac{\sigma(T_1) \times \sigma(T_2)}{\sigma(T)}\]
Proof: From the statement of the lemma we have \( T_1 \cap T_2 = T \). Without loss of generality we can assume the following hold: \( |T_1| = k; |T_2| = l; l < k; \) and can therefore deduce \(|T| = l - 1\).

Now, we pick a tree, \( T'_2 \), such that: i) \( T'_2 \) contains \( T_2 \); ii) \( T_1 \cup T'_2 = T_1 \cup T_2 \); and iii) \( |T_1 \cap T'_2| \), denoted as \(|T'| = k - 1\). We would like to note here that condition i) can be derived from the other two conditions but we explicitly state it here for the sake of expository simplicity. One can find such a tree by essentially starting with the intersection component \((T)\) of \( T_1 \), and expanding it to all but one element of \( T_1 \) and then union it with \( T_2 \) to get \( T'_2 \).

Now from Lemma 6 and the way we pick \( T'_2 \) we have:

\[
\sigma(T_1 \cup T_2) = \sigma(T_1 \cup T'_2) = \frac{\sigma(T_1) \times \sigma(T'_2)}{\sigma(T)} = \frac{\sigma(T_1) \times \sigma(T'_2)}{\sigma(T_1 \cap T'_2)} \tag{1}
\]

Now, we pick a tree, similar to the above instance, \( T''_2 \), such that: i) \( T''_2 \) contains \( T_2 \); ii) \( T'_2 = T' \cup T''_2 \); and iii) \( |T' \cap T''_2| \), denoted as \(|T''| = k - 2\). From Lemma 6 we know have:

\[
\sigma(T'_2) = \frac{\sigma(T'_2) \times \sigma(T)}{\sigma(T' \cap T''_2)} \tag{2}
\]

Substituting this (2) in (1) and canceling terms we have:

\[
\sigma(T_1 \cup T_2) = \frac{\sigma(T_1) \times \sigma(T''_2)}{\sigma(T' \cap T''_2)}
\]

Expanding \( T' \) we have:

\[
\sigma(T_1 \cup T_2) = \frac{\sigma(T_1) \times \sigma(T''_2)}{\sigma(T_1 \cap T''_2 \cap T'_2)}
\]

Since we know that \( T''_2 \) is a strict subset of \( T'_2 \) we can reduce this to:

\[
\sigma(T_1 \cup T_2) = \frac{\sigma(T_1) \times \sigma(T''_2)}{\sigma(T_1 \cap T''_2)}
\]

The rest of the proof relies on repeating this process till we get:
σ(T₁ ∪ T₂) = \frac{σ(T₁)×σ(T₂)}{σ(T₁ ∩ T₂)}

6.3.2 Twig Decomposition

In this section, we discuss how to decompose a large twig query into basic twigs and also how to estimate its selectivity.

Recursive Decomposition Scheme

This decomposition is obtained directly from Lemma 6. Since each tree has at least two leaf nodes (if the root node has degree 1, it can also be considered a leaf node for our purposes), we can always obtain two subtrees of the original tree by removing one leaf node or the other. These subtrees are labeled T₁ and T₂, respectively. If the size of T is k, then the size of T₁ and T₂ will be (k − 1). Suppose the common part between T₁ and T₂ is T₃, then we can apply the above formula to estimate the selectivity of T, given the selectivity of T₁, T₂ and T₃.

This decomposition scheme ensures that the overlap between T₁ and T₂ is maximal and thus ensures that the correlation of occurrence is well captured. If T₁ and T₂ are too large to fit in the lattice summary, then we execute the above decomposition process recursively, until we reach the brim of the lattice summary. We present an example of this recursive decomposition in Figure 6.2a. Here a twig of size 7 is decomposed into a set of sub-twigs of size 4. The bold nodes are chosen to be eliminated at each step in the recursion. Figure 6.3 presents the formal algorithm of the estimator.

Voting Scheme Extension: We note that a twig may have more than two leaves. In this case the choice of leaf nodes for decomposition may result in different estimates. Correspondingly, we can have multiple estimations at each recursive step. As an optimization,
Procedure: Estimate \((T, L)\)

Input: \(T\), an XML twig;
\(L\), the lattice summary;

Output: \(\hat{\sigma}\), selectivity estimation for \(T\);

1: \textbf{if} \(T \in L\) \textbf{then}
2: \hspace{1em} \textbf{return} the associated count;
3: \textbf{else}
4: \hspace{1em} select a pair of \(T\)’s nodes \((v_1, v_2)\) having degree of 1;
5: \hspace{1em} remove \(v_1\) from \(T\) to get \(T_1\),
6: \hspace{1em} remove \(v_2\) from \(T\) to get \(T_2\);
7: \hspace{1em} evaluate \(T_3 = T_1 \cap T_2\);
8: \hspace{1em} \textbf{return} \(\hat{\sigma} = \frac{\text{Estimate}(T_1, L) \times \text{Estimate}(T_2, L)}{\text{Estimate}(T_3, L)}\)
9: \textbf{end if}

Figure 6.3: Algorithm for recursive decomposition estimator
we record all the estimations at a given level and average them to obtain a resulting estimate to be used in the next step. Intuitively, we expect to avoid skewed estimates resulting from poor initial choices and that this optimization will prevent the propagation of errors during the course of the decomposition. Different voting schemes can be applied here. We will demonstrate the effect of this optimization in Section 6.4.

**Fast Fixed-sized Decomposition Scheme**

Assuming we can keep the information of all subtrees no larger than $k$ in the lattice summary, we can decompose a large query $T$ in the following way: We use small fixed-sized subtrees to progressively cover $T$. First, we sort all nodes in the twig in pre-order fashion. Then we choose a $k$-subtree of $T$ to cover the first $k$ nodes. Let the covered portion of $T$ be denoted as $T_c$. At each following step we cover a new node $v$ using $T_{new}$, where all the nodes of $T_{new}$ is a subset of $T_c$ except $v$. Correspondingly, we update $T_c$ as the union of the previous $T_c$ and $T_{new}$. Thus, $T_c$ will progressively grow until it covers all the nodes in $T$. Also, it holds that the part common between $T_c$ and $T_{new}$ is a $(k - 1)$-subtree. Clearly, $T$ can be covered by exactly $(\text{size}(T) - k + 1) k$-subtrees. The correlation between two subtree patterns is captured by their common part. In Figure 6.2b, we present an example of this decomposition. Newly covered nodes are highlighted at each step. Figure 6.4 presents the formal algorithm of the fixed-sized decomposition scheme.

The correctness of the above algorithm is formally stated as Lemma 8. Furthermore, Lemma 9 describes the corresponding selectivity estimator using such a decomposition scheme.
Procedure: FSD($T$, $k$)
Input: $T$, an XML twig of size $n$; $k$, a fixed size;
Output: $D$, a set of $k$-subtrees satisfying the condition in Theorem 2;

1: Order all nodes of $T$ according to pre-order:
i.e., $v_1, v_2, v_3, \ldots, v_n$;
2: Choose the subtree $t_1$ consisting of the first $k$ nodes
   from the node list and label them as covered;
   \{ * $t_1$ must be a valid subtree * \}
3: Initialize $T_c$ by $t_1$, add $t_1$ to $D$;

4: for all remaining uncovered node $v_i$ do
5:   select a subtree $t_i$ containing $v_i$ as the rightmost node,
      all other nodes are from $T_c$;
6:   add $v_i$ to $T_c$, label $v_i$ as covered and add $t_i$ to $D$;
7: end for
8: return $D$;

Figure 6.4: Fixed-sized decomposition algorithm

Lemma 8 Given a rooted ordered labeled tree $T$ of size $n$, it can be covered by $n - k + 1$
of its subtrees of size $k$ ($n > k$), i.e., $T_1, T_2, \ldots, T_i, \ldots, T_{n-k+1}$, such that $T_i \cap (\bigcup_{j=1}^{i-1} T_j)$ is
a $(k - 1)$-subtree.

Lemma 9 Assume we have a twig query $T$ decomposed into $k$-subtrees, i.e., $T_1$, $T_2$, \ldots,
$T_i$, \ldots, $T_{n-k+1}$, and, $C_{i-1} = T_i \cap (\bigcup_{j=1}^{i-1} T_j)$, $2 < i \leq n - k + 1$. Then the selectivity of $T$
may be estimated as follows:

$$\hat{\sigma}(T) = \prod_{i=1}^{n-k+1} \sigma(T_i) \prod_{j=1}^{n-k} \sigma(C_j)$$

The advantage of this scheme is that it is very simple and the decomposition is very fast.
In reality however, the lattice summary does not necessarily store all patterns up to some
size. Thus, the above decomposition can not be applied directly. To overcome this prob-
lem, we devise a hybrid version of this scheme and the recursive decomposition scheme
with voting. The hybrid scheme works as follows: For a large twig, we first decompose it into fixed-sized sub-twigs and then use the recursive decomposition scheme with voting to estimate the selectivity of all of these sub-twigs. Finally, we use Lemma 9 to obtain the estimation for the original query. The advantage of this scheme is that it is much faster than the recursive decomposition scheme with voting. Additionally, it utilizes the summary information more effectively through voting, compared to the recursive decomposition scheme without voting. We call this hybrid version the fast fixed-sized decomposition scheme and refer it as fast decomposition in Section 6.4.

### 6.3.3 Building Basis Statistics

The summary records the occurrence statistics of basic twigs. There exists redundancy in the summary that can be pruned to reduce its size. With this in mind, we formally define the notion of a $\delta$-derivable pattern.

**Definition 9** A twig pattern is $\delta$-derivable if and only if its true selectivity is within an error tolerance of $\delta$ to its expected selectivity (according to TreeLattice).

By Definition 9, 0-derivable ($\delta$-derivable with $\delta = 0$) patterns have the exact true selectivity as their expected selectivity. It is therefore safe to prune away the 0-derivable patterns from the lattice summary without sacrificing the quality of the estimations. This observation is formally stated as Lemma 10. As a result, we have more space to store more non-derivable patterns in the lattice summary.

**Lemma 10** The estimation given by TreeLattice with a lattice summary $L$ is exactly the same as that when 0-derivable patterns are removed from $L$.

**Proof:** The proof is trivial and is omitted. □
Procedure: TreeLattice-Build \((D, S, \delta)\)

Input: XML document \(D\); space budget \(S\); error tolerance \(\delta\);

Output: TreeLattice summary \(L\) of size \(\leq S\);

1: Obtain all 1-subtree and 2-subtree patterns in \(D\) and their counts;
   Use them to initialize \(L\);
2: \(k = 3\);
3: \(\textbf{while} \; k < \text{MAX\_LEVEL} \; \textbf{do}\)
4: \(\textbf{end while}\)
5: \(\textbf{if} \; e > \delta \; \textbf{then}\)
6: \(\textbf{end if}\)
7: \(\textbf{if} \; \text{size}(L) \geq S \; \textbf{then}\)
8: \(\textbf{end if}\)
9: \(\textbf{return} \; L\);

Figure 6.5: Algorithm TreeLattice-Build

The above idea can be generalized by varying \(\delta\), thereby controlling the trade-off between accuracy and memory utilization. We build the basis statistics in a bottom-up fashion. We collect the selectivity information of small twigs first, followed by the larger twigs. Essentially, we give more priority to smaller twigs, since they are more basic building blocks. Furthermore, at each level, we give priority to the more frequent twigs, as they are more important in capturing the overall twig distribution. Note that we only keep the information of non-derivable patterns in the lattice summary. Figure 6.5 presents the formal algorithm of building the basis statistics.
6.4 Experimental Results

In this section, we examine the performance of our proposed approach for XML twig selectivity estimation on synthetic and real-life datasets. We compare our approach with TreeSketches, a state-of-the-art scheme [97].

6.4.1 Experimental Setup

All the experiments were conducted on a Pentium 4 2.66GHz machine with 1GB RAM running Linux 2.6.8. Below we detail the datasets, workloads and error metric considered in our evaluation.

Datasets: We use four publicly available datasets in our experiments: Nasa, a real-life dataset converted from legacy flat-file format into XML and made available to the public; PSD (Protein Sequence Database), a real-life dataset of integrated collection of functionally annotated protein sequences; XMark, a synthetic dataset that models transactions in an online auction site and IMDB, a real-life dataset from the Internet Movie Database Project. We would like to note that for the PSD dataset, both algorithms take a long time to process, so we present results on a sample. The main characteristics of the datasets are summarized in Table 6.1.

Query Workloads: In our experiments, we consider three different kinds of workloads: random, frequent-twig and negative-query. Regardless of workload, the first step is to enumerate all possible queries for a given dataset. This set of queries is further partitioned, where each partition corresponds to twig queries of a certain size. For the random workload, we sample a fixed amount from each partition under a uniform random distribution to yield a total of 1000 queries. This level-wise partitioning and sampling also enables
us to evaluate the performance of our strategies, in particular their error propagation, in a controlled manner.

For the frequent-twig workload, we pick the most frequent 1000 twig queries as the workload. An alternative strategy would be to sample twigs as a function of the frequency of occurrence (a stratified sampling model). However, we observed little difference in the performance of these two frequency-based strategies and thus limit our discussion to the frequent-twig workload. Twig queries in the frequent-twig workload will have large selectivity rates, as expected.

We also generate and evaluate various negative-query workloads (workloads exclusively consisting of queries with zero selectivity). To generate these workloads, we followed the initial step of enumerating all possible queries. For each twig we then replaced node labels in accordance with their frequency of occurrence. More frequent labels are used for replacement more often so there is a greater chance for erroneous predictions (since sub-twigs are more likely to occur frequently). We then filter those queries whose selectivity is above 0. Once again we limit the workloads to be of size 1000. Experimental results show that TreeSketches is always accurate (100% of the time), and that TreeLattice is almost always as accurate (99% of the time), and returns the correct answer (zero). There is little difference between these two strategies for negative workloads, so we do not consider this workload further.

**Error Metric:** We quantify the accuracy of estimations using the average absolute relative error over all queries in the workload. The absolute relative error is defined as $|\sigma - \hat{\sigma}| / \max(s, \sigma)$, where the sanity bound $s$ is used to avoid the artificially high percentages of low selectivity queries. Following common practice [97, 98], we set $s$ to be the 10-percentile of true query counts. We use a lower bound of 10 if $s$ should fall below that value.
6.4.2 Accuracy of Estimators

Here we examine the accuracy of the estimators on our workloads. For both TreeLattice and TreeSketches, we limit the summary size to 50KB. Figures 6.6a-d show the average selectivity estimation error on various frequent-twig workloads for all four datasets.

An obvious trend that stands out is that as the size of the twig query increases, the quality of the estimation decreases. This is not surprising, since the estimation errors grow for larger-sized queries for both strategies. Specifically for TreeLattice, the smaller sized queries are closer to the lattice boundary (exact information maintained in the summary) and thus subject to less estimation error. In contrast, for larger queries, depending on the number of decomposition and estimation steps, the error will accumulate, finally affecting the quality of the estimations. On the Nasa dataset, for example, the recursive decomposition estimator yields very accurate estimations on frequent-twig workloads of size 5 and 6, with error 0.0% and 4.2%, respectively. In contrast, on the frequent-twig workloads of size 8 and 9, the error increases to 11.6% and 17.8%, respectively. The effect of error accumulation can be clearly seen from the results. The other two estimators have a similar trend when working on various workloads for all datasets.

We would like to note that the voting scheme refines the estimations effectively by mitigating the error propagation. The recursive decomposition estimator with voting usually yields the most accurate estimations. Additionally, one should note that the estimations returned by the fast decomposition estimator is very similar to that returned by recursive decomposition estimator with voting.

When comparing the two strategies, it can be observed that TreeLattice significantly outperforms TreeSketches for both the Nasa and PSD workload on all query sizes. On the XMark dataset, TreeLattice is near perfect and TreeSketches is marginally worse (note the
Y-axis scale). On the IMDB workload, TreeSketches outperforms TreeLattice significantly on larger query sizes. On smaller query sizes, the difference is not as significant. Note that on Nasa, PSD and XMark, in most cases, even the weakest estimation strategy in TreeLattice, recursive decomposition estimator without voting, does better than TreeSketches.

Figures 6.7a-d show the average selectivity estimation error on the random workloads for all four datasets. The trends are very similar to the ones observed for the frequent-twig workloads. Two differences are that TreeLattice is closer in performance to TreeSketches on the IMDB dataset and on the XMark dataset TreeSketches performs poorly (the errors are well above 100% in some cases).

To examine a possible outlier effect, we plotted the cumulative distribution function of the errors. Figures 6.8a-d present the results for frequent-twig workloads. The results are consistent with Figure 6.6, showing that TreeLattice outperforms TreeSketches consistently on all datasets except IMDB. The results on random workloads are similar and are omitted.

In conclusion, these results demonstrate that TreeLattice is effective in summarizing the distribution of the underlying twigs. Furthermore, we show that TreeLattice is effective in processing both frequent and infrequent twig queries. When the query size is increased, the quality of the estimation is reduced (due to the error propagation). Specifically among the strategies evaluated, the recursive decomposition with voting estimator usually yields the best estimations. Finally, the fast decomposition estimator yields close estimations to the recursive decomposition estimator with voting.

### 6.4.3 Impact of Varying Summary Size

In this experiment, we measure the estimation error while varying the summary size. We use a frequent-twig workload containing frequent 8-twig queries. Figures 6.9a-d show
the average selectivity estimation error when varying the summary size for Nasa, PSD, XMark and IMDB, respectively. As expected, we observe that an increase in the size of the summary yields more accurate estimations. As before, TreeLattice works extremely well for Nasa, PSD and XMark. An important point here is that the estimation error for these datasets is well below 10% when we use at least a 40KB summary. For the IMDB dataset, TreeSketches is better than TreeLattice.

6.4.4 Implications on Estimation Time

In this experiment, we compare TreeLattice against TreeSketches in terms of selectivity-estimation time. Figures 6.10a-d present the response times of the different approaches on frequent-twig workloads for Nasa, IMDB, PSD and XMark, respectively. The results on random workloads are similar and are omitted. As seen in the figures, in most cases, all TreeLattice estimators are much more efficient than TreeSketches. Specifically, TreeLattice runs extremely fast when processing relatively small twig queries. As we increase the query size of the workload, the recursive decomposition estimator with voting becomes much slower. The degradation of response time becomes more significant as we increase the size of the twig queries. This is not surprising, since the number of all possible decompositions increases exponentially with the number of recursion levels. The recursive decomposition without voting is fastest. However in terms of accuracy, this strategy is the weakest among the three. The overall performance of the fast decomposition estimator is clearly the best since it is close to recursive decomposition estimator in terms of response time, and it is close to the recursive decomposition estimator with voting in terms of estimation quality.
6.4.5 Impact of $\delta$-derivable Pruning

The pruning strategy we describe earlier allows us to replace $\delta$-derivable patterns with non-derivable patterns in the lattice summary. Here we examine the potential benefits of this strategy on the IMDB dataset. We use the frequent-twig workload for this experiment. Figure 6.11 presents the estimation quality at the different $\delta$-levels. All summary sizes are fixed at 50KB. As can be seen in the figure, when we increase $\delta$, the estimations become more accurate for large twig queries. This comes at a small sacrifice in the estimation accuracy for small twig queries. If the estimation error for the small twig queries is tolerable, we can continue to increase $\delta$ for the benefit of improved estimations for large twig queries. If we consider a single large workload consisting of a uniform number of different-sized queries (4 to 9), $\delta = 20\%$ gives the lowest average error (17.9%).

6.4.6 Comparison of Summary Construction Times

In this experiment, we evaluate the cost of constructing the summary. In TreeSketches, this is a very expensive operation as it involves a bottom-up clustering of similar substructures in the XML data tree. In contrast, our approach relies on fast off-the-shelf efficient tree-mining algorithms to build the summary. Table 6.3 presents the time required by both approaches to construct a 50KB summary on each of the four datasets. The advantage of our approach over TreeSketches is quite telling—with an improvement of about one order of magnitude.

6.4.7 Result Summary and Rationale

The experimental results have shown that TreeLattice is very effective and efficient in estimating selectivity of the XML twig queries. In most cases, TreeLattice outperforms
TreeSketches in terms of both accuracy and response time. In addition, pruning \( \delta \)-derivable patterns can further refine the selectivity estimation for large queries. We also notice that TreeLattice is outperformed by TreeSketches on IMDB, though it still yields reasonable estimations. Here we attempt to explain the rationale behind these results.

We know that TreeLattice is based on the conditional independence assumption of twig growing. If real XML data satisfy this assumption well, then TreeLattice will perform well. On the other hand, if the assumption does not hold, it will not. From the experimental results, it would appear that Nasa, PSD and XMark satisfy the assumption well and IMDB does not. Our expectation is supported by Table 6.4, which lists the number of patterns satisfying the assumption at different lattice levels on the four datasets. From the table, we see for the Nasa, PSD and XMark datasets, the ratio of 0-derivable patterns to total patterns is quite high, meaning they satisfy the assumption well. In contrast, the ratio on IMDB is much lower, which is reflected in the results.

Now let us take a closer look at TreeSketches. The TreeSketches synopsis is constructed by a bottom-up clustering of the similar substructures in the XML data tree. In it, an edge \((x, y)\) with weight \(\alpha\) represents that on average, each node in set \(x\) has \(\alpha\) children in set \(y\). Assume we have \(n\) nodes in set \(x\), and the nodes have \(\alpha_1, \ldots, \alpha_n\) children in set \(y\), respectively. If there are many similar substructures in the XML data tree found by bottom-up clustering, then TreeSketches should work very well (e.g., IMDB). On a detailed examination of IMDB, we find this to be the case. However, if this does not hold, then one is forced to cluster substructures that are not very similar in order to compress the XML data tree. This results in a large variance of \(\alpha_i\) which leads to larger errors that propagate rapidly. We believe this explains the poor performance of TreeSketches on the other three datasets.
Figure 6.6: Average estimation error on frequent-twig workload: (a) Nasa (b) PSD (c) XMark (d) IMDB

Figure 6.7: Average estimation error on random workload: (a) Nasa (b) PSD (c) XMark (d) IMDB

Figure 6.8: Average estimation error distribution on frequent-twig workload: (a) Nasa (b) PSD (c) XMark (d) IMDB
Figure 6.9: Average estimation error when varying summary size: (a) Nasa (b) PSD (c) XMark (d) IMDB

Figure 6.10: Average response time on frequent-twig workload: (a) Nasa (b) PSD (c) XMark (d) IMDB

Figure 6.11: Average selectivity estimation error when varying delta (IMDB)
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Elements</th>
<th>File Size(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nasa</td>
<td>476646</td>
<td>24</td>
</tr>
<tr>
<td>PSD</td>
<td>335193</td>
<td>12</td>
</tr>
<tr>
<td>XMark</td>
<td>167864</td>
<td>12</td>
</tr>
<tr>
<td>IMDB</td>
<td>155898</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 6.1: Dataset characteristics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nasa</th>
<th>PSD</th>
<th>XMark</th>
<th>IMDB</th>
</tr>
</thead>
</table>
| Query Size
| Frequent | Random | Frequent | Random | Frequent | Random | Frequent | Random |
| 4       | 5377  | 4073 | 6601  | 3321  | 2774   | 1722   | 3519    | 784    |
| 5       | 8282  | 3742 | 11563 | 2827  | 5995   | 2058   | 40703   | 985    |
| 6       | 21334 | 3978 | 28160 | 2398  | 6347   | 3251   | 11815   | 1982   |
| 7       | 58920 | 4004 | 68877 | 2383  | 169993 | 6641   | 17193   | 2937   |
| 8       | 29558 | 2855 | 129892| 2920  | 288944 | 10394  | 29962   | 3559   |
| 9       | 18814 | 2608 | 148993| 2464  | 281808 | 5748   | 37963   | 5825   |

Table 6.2: Workload characteristics (average no. of binding tuples)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TreeLattice</th>
<th>TreeSketches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nasa</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>PSD</td>
<td>21</td>
<td>102</td>
</tr>
<tr>
<td>XMark</td>
<td>15</td>
<td>78</td>
</tr>
<tr>
<td>IMDB</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 6.3: Summary construction time (in minutes)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nasa</th>
<th>PSD</th>
<th>XMark</th>
<th>IMDB</th>
</tr>
</thead>
</table>
| Lattice Level
| # total | # 0-derivable | # total | # 0-derivable | # total | # 0-derivable | # total | # 0-derivable |
| 3       | 213   | 174 | 282   | 201   | 365   | 302   | 877   | 156   |
| 4       | 668   | 434 | 1284  | 1016  | 1283  | 1138  | 9839  | 3625  |
| 5       | 2296  | 1866| 6728  | 5778  | 4378  | 3948  |  -    |  -    |
| 6       | 8274  | 7768| 34976 | 31580 | 14492 | 13251 |  -    |  -    |
| 7       | 30492 | 29232|  -   |  -   | 46628 | 43373 |  -    |  -    |

Table 6.4: Number of total and 0-derivable patterns on four datasets
6.5 Related Work

Chen et al. [95] were among the first to study the problem of estimating twig counts. They propose the Correlated Sub-path Tree (CST) method for estimating the selectivity of XML twig queries. A CST is a suffix tree-based data structure used to store all the paths up to certain length. To estimate the selectivity of a given twig query, this approach needs to decompose a twig into a set of paths stored in the CST. Note that even though both the CST and our TreeLattice approach depend on decomposing a large twig into basic twigs, they are quite different in several respects. First, our approach utilizes the subtrees instead of paths as the summary of an XML document. Our results have shown that these subtrees capture the structure of an XML document very effectively. In contrast, in order to perform selectivity estimation, CST has to store additional information, called set hashing signature, in order to capture the correlation among paths. Our approach is essentially a generalization of the Markov model-based approach for XML path selectivity estimation. When dealing with XML path queries, TreeLattice yields the same selectivity estimation as the Markov model-based approaches, which have been shown to be more effective than the CST-based approach [90].

XSketches [128] exploits localized graph stability in a graph-synopsis model to approximate path and branching distribution in an XML data graph. Its successor integrates support for value constraints as well, by using a multidimensional synopsis to capture value correlations [96]. They augment the XSketches model with new distribution information [98] to estimate the selectivity of XML twig queries and show that XSketches performs better than CST, yielding estimates with significantly lower estimation error.

TreeSketches [97], a successor of XSketches, clusters similar fragments of XML data together to generate its synopsis. The granularity of the clustering depends on the memory
budget. Also, it outperforms its predecessors in terms of both accuracy and construction time. We note that the scope of TreeSketches is much broader than that of TreeLattice, since they are able to handle more general twigs (containing // operator).

A particular case of the twig query is the XML path query. The wide use of XML path queries has motivated many researches on estimating their selectivity. The Lore system [88] is one of the earliest works in this direction. It stores statistics of all distinct paths up to length $m$, with $m$ being a tunable parameter. Selectivity of paths longer than $m$ are estimated assuming the Markov property. Aboulnaga et al. [90], extends the idea used by Lore system in their Markov table method. It consists of a set of pruning and aggregation techniques on the statistics used in the Lore system and therefore offers an improvement by reducing the space requirements. Aboulnaga et al. [90], also propose a tree-based method known as the path tree, for estimating the selectivity of XML paths without data values. A path tree is a summarized form of the XML data tree. Compared with the Markov table method, this approach is inferior in terms of estimation accuracy for real datasets [90].

XPathLearner [89], is an on-line, self-tuning, Markov table-based approach used to estimate the selectivity of XML paths. The statistics of the data are collected in an on-line fashion, thus it is workload-aware. By design, our approach is also incremental in nature and can maintain summaries on-line, though we do not evaluate this aspect here. Our method is a generalization of these Markov model-based approaches for more complex twig queries. Recently, Wang et al. [92] propose the use of Bloom Histograms to estimate XML path selectivity. It is the first approach that gives a theoretical bound on the estimation error. However, it does not handle twig queries.
6.6 Conclusions

In this chapter, we have described our efforts on mining non-redundant tree patterns on large XML data. These patterns are leveraged to model the XML data and to solve the XML twig selectivity estimation problem. Our proposed TreeLattice approach is shown to be comparable or better than TreeSketches in terms of estimation accuracy. Moreover, our technique is significantly faster both in terms of summary construction and in terms of selectivity estimation. Furthermore, we have provided theoretical foundations for the estimation process and have shown that TreeLattice subsumes the successful Markov model-based XML path selectivity estimation approach as a special case.

In the future, we will study the following issues: First, we would like to extend TreeLattice to handle more complex twig queries with recursion predicates (/ operator). In this case, we are allowed to grow the twig in a more relaxed fashion. We conjecture that the conditional independence assumption of tree growing will still hold even for this case. Second, an error bound associated with the estimation would be very useful and we have made some initial progress towards this end. Third, we would like to adapt TreeLattice in a manner similar to XPathLearner, where information learned from an on-line workload can dynamically guide what is to be maintained in the summary structure.
CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this thesis, we have proposed a framework of leveraging local patterns and probabilistic models to model, query and analyze large-scale structured and semi-structured data. The framework consists of three main steps:

**Step 1:** Mining non-redundant local patterns from data;

**Step 2:** Gluing these local patterns together by employing probabilistic models (e.g., Markov random fields (MRF), Bayesian networks);

**Step 3:** Reasoning (making inference) over the data for solving various data analysis tasks.

We have studied a series of important problems in this framework: mining non-redundant frequent itemset patterns on large transactional data; employing local probabilistic models to glue these non-redundant itemset patterns; employing global probabilistic models to glue these non-redundant itemset patterns; and mining non-redundant tree patterns on large semi-structured XML data. Our research has introduced new techniques and algorithms to solve these problems, and also demonstrated their efficiency through detailed experimental evaluation. We have shown that an aggregation of local patterns mined from structured and semi-structured data can provide key insights of the data and can be used to model the data.
Furthermore, the probabilistic models can be used to glue these local patterns together to model the data. Finally, the learned models can be effectively employed for various data analysis tasks.

- In Chapter 3, we presented an approach of identifying non-redundant itemset patterns from a large collection of frequent itemsets on transactional data. A frequent itemset pattern is non-redundant if its occurrence count cannot be inferred or estimated from lower-order statistics. The basic idea is to leverage undirected graphical models. We use statistics of smaller itemset patterns (lower-order statistics) first to learn an MRF, and then use the learned model to infer the occurrence statistics of larger itemset patterns (higher-order statistics) thereafter. If the estimations are accurate enough, we bypass the corresponding patterns. Otherwise, we use the extra information from them to augment the model. All the itemsets selected by the final model afford a concise representation of the original collection of itemset patterns, with each one encoding non-redundant information for modeling the underlying data distribution. Experimental results show that the technique can effectively eliminate redundancies from a large collection of itemset patterns. Interestingly, we note that these selected itemsets can be viewed as generalized non-derivable itemsets.

- In Chapter 4, we described an approach of employing local probabilistic models to glue non-redundant itemset patterns together to solve the link prediction task in social network analysis. The new technique effectively combines topology analysis on network structure data and frequency analysis on network event log data. For a given pair of nodes, the local common neighborhood of these two nodes in the network is first identified governed by the topology of the network. Subsequently, relevant local patterns are collected from the underlying event log data and used to learn a
local probabilistic model over this local common neighborhood data region. Finally, we use the model to induce a co-occurrence probability feature of the two nodes and use it to aid link prediction. We demonstrate that this co-occurrence probability feature can be computed in a scalable manner and is highly discriminatory for link prediction when compared with state-of-the-art topological and semantic features on several real world datasets. Moreover, we demonstrate that the co-occurrence probability feature can be effectively combined with other features (semantic and topological features). One can then employ standard supervised learning algorithms to make link predictions.

- In Chapter 5, we explored employing global modeling strategies to glue non-redundant itemset patterns on large transactional data. We propose an approach to learning approximate Markov random fields based on non-redundant frequent itemset patterns. The modeling approach first employs graph partitioning to cluster item variables into balanced disjoint partitions. Then we use important interactions across partitions to augment these partitions to account for interdependencies across them. An exact local MRF is learned for each partition and all the local MRFs are then aggregated together to deliver an appropriate global model of the data. A greedy inference mechanism has been developed for this model. Experimental results on real datasets show the efficacy of this model for solving the problem of approximate querying of the data.

- In Chapter 6, we presented our efforts to model semi-structured XML data. The above idea of identifying non-redundant itemsets is generalized to deal with more complex structural tree patterns. We propose an approach to identifying non-redundant
tree patterns from a large collection of structural tree patterns. To this end, we pro-
pose a conditional independence assumption for tree growth. If a tree pattern satis-
fies this assumption well, we consider it redundant. Otherwise, we consider it non-
redundant. We show that our approach can effectively eliminate redundancies from
a large collection of structural tree patterns. Furthermore, we propose techniques of
employing these non-redundant tree patterns as summary statistics for the XML data
to handle XML twig selectivity queries effectively.

Many interesting problems have been raised in our research. Although we have made
significant progress towards tackling them, some of them still remain open. In the future,
we would like to address the following research problems.

• As we noted in Chapter 3, the non-redundant itemset pattern selection problem can
be treated as a feature selection problem. $L_1$ regularization-based feature selection
techniques are increasingly popular nowadays. LASSO [105] is perhaps the most
famous feature selection technique for linear regression. We are specifically inter-
ested in using $L_1$ regularization techniques in our study. It has been known that $L_1$
regularization optimizes a joint objective that trades off fit to data with the sum of
the absolute values of the parameters, and tends to prefer sparse solutions. From
the non-redundant itemset selection’s perspective, this presents a natural tradeoff be-
tween the number of selected patterns and their ability in modeling the underlying
data distribution.

• We would like to explore the problem of selecting non-redundant graph patterns from
a large set of graph patterns on large graph data. Dealing with tree patterns is a spe-
cial case of this problem. Our proposed tree growth conditional independence model
in Chapter 6 is a meaningful attempt towards this direction. We believe that this effort can be vigorously expanded. The motivation is clear – A concise representation of a large collection of graph patterns will improve their interpretability and end-usability. There has been an interesting work by Ueda et al. [129], where they propose a probabilistic model for labeled ordered trees. More specifically, a node in the tree depends on its immediately elder sibling as well as its parent. Their model considers the dependencies between ordered siblings as well as parent-child relationships in a tree, which has been shown to be enough for capturing various types of patterns in labeled ordered trees [129]. We plan to further investigate this work, and extend it for the task of selecting non-redundant tree or graph patterns.

- Data is evolving all the time. We want to fully exploit the temporal information of the data in our framework. This leads to a series of interesting research problems as follows.

  (1) How to mine non-redundant local patterns for such evolving data?

  (2) How to build and maintain models for such evolving data efficiently, and can we do this in an incremental manner?

  (3) Can we harness the temporal behaviors of these patterns to better tackle various data analysis tasks?

Furthermore, many times data can have rich semantic information. For example, person nodes in a friendship social network are usually associated with certain affiliation or demographic information. This information should be accounted for in the analysis.
We have presented local and global modeling strategies for gluing local patterns. Therefore, a natural question to ask is what the optimal modeling strategy is for a specific data analysis task. How much information about local patterns does one need to solve the task? Is a local modeling strategy good enough already, or one has to employ more information and build more expensive larger models?


