SHOPPING WITH NETWORKS: AN APPROACH TO MARKET BASKET ANALYSIS

A Thesis

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science and Engineering

by

Troy Raeder

Nitesh V. Chawla, Director

Graduate Program in Computer Science and Engineering

Notre Dame, Indiana

April 2009

SHOPPING WITH NETWORKS: AN APPROACH TO MARKET BASKET ANALYSIS

Abstract

by

Troy Raeder

The *market basket problem*, the search for meaningful associations in customer purchase data, is one of the oldest problems in data mining. The typical solution involves the mining and analysis of *association rules*, which take the form of statements such as "people who buy diapers are likely to buy beer." It is well-known, however, that typical transaction datasets can support hundreds or thousands of obvious association rules for each interesting rule, and filtering through the rules is a non-trivial task. One may use an interestingness measure to quantify the usefulness of various rules, but there is no single agreed-upon measure and different measures can result in very different rankings of association rules. In this thesis, we take a different approach to mining transaction data. By modeling the data as a *product network*, we discover more expressive communities (clusters) in the data, which can then be targeted for further analysis. We demonstrate that the network based approach can isolate influence among products without excessive ambiguous associations. We further consider a collaborative marketplace, where it may be beneficial for the market for stores to share their product networks. To that end, we propose a robust privacy preserving protocol that encourages stores to share their product network without compromising their individual information. We demonstrate the effectiveness of the product networks and the privacy preserving protocol on a real-world store data. Finally, we build upon our experience with product networks to propose a comprehensive analysis strategy by combining both traditional and network-based techniques.

CONTENTS

FIGURES iv

TABLES v

ACKNOWLEDGMENTS vi

CHAPTER 1: INTRODUCTION 1
- 1.1 Contributions 4

CHAPTER 2: DATA 6

CHAPTER 3: ASSOCIATION RULES 12

CHAPTER 4: NETWORK METHODS 19
- 4.1 Constructing a Network of Products 20
- 4.2 Community Detection 23
- 4.3 Detecting Communities of Products 26
- 4.4 Experimental Evaluation 32
- 4.5 The Performance of Other Community Detection Algorithms 37
- 4.6 Related Work 38
- 4.7 Conclusion 39

CHAPTER 5: MOVING TOWARD A COMPREHENSIVE ANALYSIS STRATEGY 40
- 5.1 Association Rules Networks 41
- 5.2 Center-Piece Subgraphs 45
  - 5.2.1 Objective Function Definition 46
  - 5.2.2 Application to Market Basket Analysis 47
  - 5.2.3 Center-Piece Subgraphs on Market Basket Data 48
- 5.3 A Strategy for Market Basket Analysis 52
- 5.4 Choosing the Minimum Support Parameter 53

CHAPTER 6: PRIVACY-PRESERVING SOLUTION . . . 56
6.1 Related Work . . . 57
6.2 Preliminaries of the Private Product Correlation (PPC) Protocol . . . 58
6.2.1 Homomorphic encryption . . . 58
6.2.2 Zero-knowledge proofs . . . 59
6.2.3 Privacy-preserving set operations . . . 59
6.3 Private Product Correlation Protocol . . . 60
6.3.1 Overview of the solution . . . 60
6.3.2 Protocol description . . . 61
6.3.3 Sub-protocols . . . 62
6.3.4 Complexity Analysis . . . 63
6.4 Security Analysis . . . 63
6.5 Security Caveats . . . 66
6.5.1 Number of Participants . . . 66
6.5.2 Collusion . . . 67
6.5.3 Lying Adversaries . . . 68
6.6 Efficiency Improvements . . . 70
6.6.1 Polynomial multiplication and evaluation . . . 70
6.6.2 Packing . . . 71
6.7 Performance . . . 72
6.7.1 Revealing Pairs with Zero Sales . . . 74
6.8 Incorporating Additional Analysis Techniques . . . 76
6.9 Conclusion . . . 77

CHAPTER 7: CONCLUSIONS AND FUTURE WORK . . . 78
7.1 Future Work . . . 80

BIBLIOGRAPHY . . . 81

FIGURES

2.1 Top 50 Association Rules 10

3.1 Number of associations discovered at varying levels of confidence and support. 14

4.1 Degree distribution for (a) the entire network and (b) the neighbors of a single product. 22

4.2 Example of a dendrogram. Taken from [46] 26

4.3 Distribution of utility scores for the 17 communities in our data. . . . 33

4.4 The first two communities in our data, ranked by the measure given in Equation 4.6. 33

4.5 Three more communities. 35

5.1 Two Association Rules Networks from the peripheral community of eggs. 41

5.2 Association Rules Network with z = Bagel. 42

5.3 Center-Piece Subgraphs with Chips as the query node. Edges are weighed by a) support and b) confidence. 48

5.4 A center-piece subgraph with eggs as the query node. 50

5.5 A center-piece subgraph with Diet Coke, Newspaper, and Peppermint Patties as query nodes, to explain the association rule. 51

5.6 Modularity of discovered communities as a function of minimum support. 54

TABLES

2.1 OUR TRANSACTION DATA, BROKEN DOWN BY THE NUMBER OF UNIQUE ITEMS IN A TRANSACTION 7

2.2 REVENUES FOR THE FULL SET OF TRANSACTIONS, BROKEN DOWN BY TRANSACTION SIZE 8

2.3 TOP TEN ITEMS IN THE STORE, BY BOTH SALES VOLUME AND REVENUE 8

3.1 HIGH, LOW, AND MEAN RANK AND STANDARD DEVIATION OF RANKS FOR THE TOP 10 RULES BY AVERAGE RANK AMONG THE 21 INTERESTINGNESS MEASURES IN [59] 17

6.1 TIMING RESULTS FOR PROTOCOL EXECUTION 74

6.2 TIMING RESULTS FOR PROTOCOL EXECUTION WHEN PARTICIPANTS DISCLOSE PAIRS WITH ZERO COUNT 75

ACKNOWLEDGMENTS

There are a number of people without whose dedicated assistance the completion of this thesis would not have been possible. I would like to thank my advisor, Dr. Nitesh Chawla, for his good-natured advice and direction and for having confidence in me even when I did not have confidence in myself. I would also like to thank Dr. Marina Blanton, with whom I collaborated on part of this work, for very patient, thoughtful, and thorough responses to a number of my questions, and Dr. Keith Frikken for assistance with the more difficult cryptographic aspects.

In addition, I am grateful to my fellow grad students, especially Karsten Steinhaeuser, Ryan Lichtenwalter, Dave Cieslak, and Darcy Davis for their general helpfulness and their willingness to assist in any and all aspects of my research.

Finally, I would like to thank the National Science Foundation and the Arthur J. Schmitt Foundation for providing portions of the funding that contributed to this work.

CHAPTER 1

INTRODUCTION

The collection and study of retail transaction data, known as *market basket analysis*, has become increasingly prevalent in the past several years. Many supermarkets, for example, issue *loyalty cards* [43]. While providing discounts to the customer, these cards allow the retailer to develop a better understanding of individuals' purchasing habits by associating customers with transactions. The uses of this information vary, but may include informing product placement decisions, designing personalized marketing campaigns, and determining the timing and extent of product promotions [1, 2, 17] among others.

Formally, the task of market basket analysis is to discover actionable knowledge in transaction databases. The problem can be understood as follows: A standard retail store sells a large set of products $\mathbf{P}$. Define a *transaction* $\mathbf{p} \subseteq \mathbf{P}$ as the set of products an individual customer buys in a single trip to the store. The store's *transaction database* $\mathbf{T} = \{\mathbf{p}\}$ is the set of all transactions the store has processed within a given time period.

The ubiquity of this type of data, and the broad application of its analysis leads to two distinct practical concerns. First, an effective analysis method should enable the retailer to draw clear, comprehensive conclusions from the data. Second, the data itself needs to reflect the shopping habits of the customer population at large. In other words, the data sample cannot be *biased* in any way because bias may impact the quality of the conclusions drawn from any analysis. [14, 34, 69]

One popular tool for market basket analysis in practice is the mining of *association rules* [2]. A set of association rules $R(\mathbf{T}, s, c)$ is defined by a transaction database $\mathbf{T}$, a *minimum support* parameter s and a *minimum confidence* parameter c. Define A and B as arbitrary sets of products. Further, define $\mathbf{A}$ (analogously $\mathbf{B}$) as the set of transactions containing every product in A. Formally, R is the set of all rules $A \rightarrow B$ such that:

- $\frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{T}|} \geq s$
- $\frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A}|} \geq c$.

Association rules have found successful application in many diverse contexts and a number of algorithms have been developed to discover them efficiently [2, 12, 35, 73], but they are not without limitations. The most prominent of these is sheer volume. Large transaction datasets tend to contain hundreds or thousands of rules at reasonable levels of support and confidence, and many of these may be redundant or obvious [41]. As a result, it is often difficult to isolate interesting relationships.

Two distinct classes of methods have evolved to address this problem. One class [31, 41, 70, 71] attempts to eliminate any rules that may be redundant, while the other [23, 44, 59] aims to elevate rules that are especially interesting (by sorting on an objective measure). Unfortunately, the concepts of both interestingness and redundancy are somewhat subjective. As a result, (which we show in Chapter 3) these methods are of limited use in practice.

Ultimately, existing literature on market basket analysis has failed to provide conclusive answers to many of the field's most pressing questions. There is little if any work, for example, on methods for appropriately selecting the support and confidence parameters for association rules. Further, there is no widely-accepted means of isolating representative or useful relationships in market basket datasets, and finally no existing work of which we are aware has attempted to offer any manner of procedural guidance for analyzing such data. In other words, no work has addressed the question *Given a new market basket dataset, what method or methods should I apply in order to obtain effective insights?*

This thesis attempts to address these concerns and improve the power and clarity of market basket analysis by modeling transactional data as a network. We show that by detecting *communities* of products in this network, we can discover strong and expressive relationships among products including relationships that are difficult to discover with traditional association rules.

We then build on our experience with product networks and with a number of different market basket analysis techniques to propose a novel procedure for mining unseen market basket datasets. The network representation of transaction data allows for the use of a number of different analysis techniques previously unavailable to the association rule community. As a result, this procedure is the first comprehensive market basket analysis framework ever proposed in the literature.

In order to ensure that a store's data provides accurate insight into the broader customer population, we propose an environment of information sharing. In isolation, any one store's data may suffer from *sample selection bias* for any number of reasons. Consider, for example, a store in a university setting that collects data only from individuals who pay on ID cards. Such a collection strategy may unintentionally lead to an overrepresentation of data from students and an underrepresentation of information about faculty, staff, and off-campus visitors.

One way to combat sample selection bias is to combine data from multiple sources with the idea that no two sources are likely to suffer the same bias. However, this seemingly simple solution is complicated by the fact that transaction data are highly proprietary. A great deal of information about strategy and business operations can be inferred from a list of transactions, and as a result few businesses are willing to share such information with competitors.

To avoid this problem, we develop a protocol for the secure sharing of product network information. Specifically, we develop a method that is secure against malicious adversaries, by which a set of n stores participate in a joint computation to determine c_{jk}, the number of times product j and product k have sold together in all stores combined. Each store chooses a set of products D_i about which it would like to learn, and the stores jointly compute the c_{jk} without leaking unintended information to any of the participants. That is, each store $\mathcal{P}_i$ only learns the total counts involving products in D_i, and no store learns any other store's inputs. We provide timing information from an actual implementation to verify that the techniques are practical for real-world use.

All of our developments and conclusions are verified on real transaction data from an on-campus convenience store at the University of Notre Dame.

1.1 Contributions

To summarize, the contributions of this thesis are:

1. A thorough analysis of the effectiveness of standard association rule methods on real-world data.
2. The development of a community detection framework which discovers complex relationships between products and a discussion of the implications of its application.
3. A privacy-preserving protocol for the secure exchange of network data among cooperating stores along with an analysis of the protocol's real-world performance.
4. A comprehensive set of recommendations for the analysis of unseen market basket datasets which incorporates both traditional and network-based analysis methods.

The remainder of this thesis is organized as follows: Chapter 2 introduces our data and discusses its characteristics in depth. Chapter 3 explores the performance of association rules on real-world data. Chapter 4 lays out our framework for market basket analysis and provides results. Chapter 6 outlines our privacy-preserving solution and proves its security in the malicious model. Chapter 5 provides our recommendations for incorporating both standard and network-based analysis techniques when analyzing market basket data. Finally, Chapter 7 concludes with final thoughts and directions for future work.

CHAPTER 2

DATA

Before introducing our methods, we briefly discuss the data we will use throughout this study. In doing so, we hope to familiarize the reader with the data that will form the basis of our results and analysis over the next several chapters.

We have collected purchase data, including items bought, purchase price, and method of payment, from the Huddle Mart, an on-campus convenience store at the University of Notre Dame. The data includes all purchases over a nearly two-year period from April 2005 to February 2007. For privacy reasons, there is no way to associate purchases with individual people, but we should still be able to derive a general picture of customers' purchase behavior.

Tables 2.1 and 2.2 break down the transactions in the store by the number of unique items each one contains. The former shows simply the number of transactions of each size and the latter indicates the total revenue generated by transactions of that size. The most striking aspect of the data is the distribution of transaction sizes: the majority of customer transactions contain only one unique item. While this is unusual of the supermarket data to which association rules are most commonly applied, small transactions are fairly typical in, for example, online retail data [76].

While single-item transactions are by far the most common, we see that they are not the most profitable. Comprising almost 59% of all transactions, they contribute only 35% of the store's revenue. By contrast large transactions, those with more

TABLE 2.1

OUR TRANSACTION DATA, BROKEN DOWN BY THE NUMBER OF UNIQUE ITEMS IN A TRANSACTION

Unique Items	Number of Transactions	Percentage of Total
1	1,023,191	58.9%
2	489,457	28.2%
3	150,068	8.6%
4	45,168	2.6%
5 or more	27,526	1.6%

than 4 items, account for almost 8% of revenues and less than 2% of checkouts. This is entirely expected, but we include it here to show that there is an advantage to understanding the interactions among products. First, those relationships that do exist are to some extent driving store revenue. Second, the store has great growth potential, as by making even a fraction of the single-item transactions larger, it could earn a great deal of money.

Finally, we provide an overview of the best-selling products and most popular product combinations in the store. Table 2.3 shows the top ten highest-revenue products in the store and the ten most popular products in terms of sales volume (the number of times sold). Note that there were two items, called "Misc. Non-Tax" and "Misc. Taxable" that we exclude from all our analyses. These designations do not represent one product in particular, but rather that the cashier entered the cost of the product by hand rather than on the scanner. Thus, any data tagged in this manner is unreliable. Misc Non-Tax would have been the second highest-revenue and second most-sold item had it been included.

It is clear from Table 2.3 that Bulk Candy is the most popular in the store, and

TABLE 2.2

REVENUES FOR THE FULL SET OF TRANSACTIONS, BROKEN DOWN BY TRANSACTION SIZE

Unique Items	Total Revenue	Percentage of Total
1	$1,824,356.47	35.6%
2	$1,693,376.71	33.1%
3	$831,887.21	16.2%
4	$365,213.35	7.1%
5 or more	$407,190.42	7.9%

TABLE 2.3

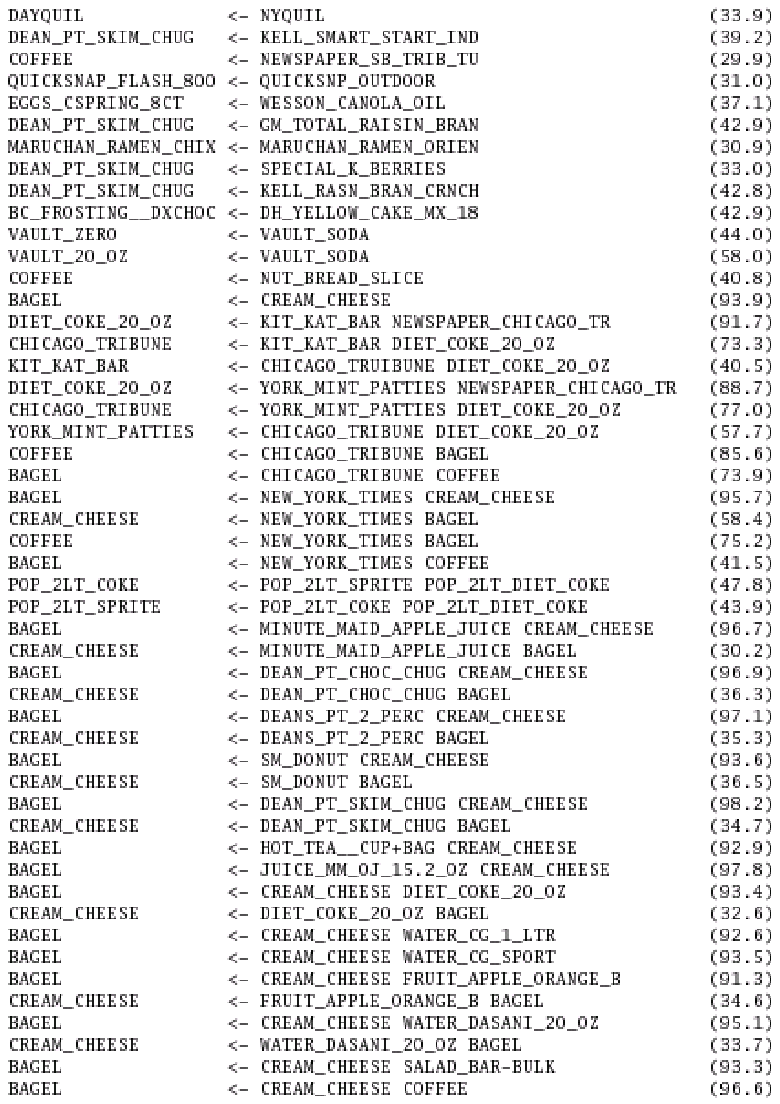
TOP TEN ITEMS IN THE STORE, BY BOTH SALES VOLUME AND REVENUE

Rank	Item	Revenue	Item	Volume
1	Bulk Candy	$231,222	Bulk Candy	124,103
2	Salad Bar-Bulk	$131,105	Quarter Dog	84,431
3	12 oz. Coffee	$82,143	12 oz. Coffee	68,030
4	16 oz. Coffee	$75,868	16 oz. Coffee	67,367
5	Ultimate Shake	$71,736	Bagel	57,314
6	20 oz. Water	$65,172	Salad Bar-Bulk	51,231
7	Bagel	$60,456	20 Oz. Water	49,724
8	1 L Water	$50,259	Fresh Fruit	37,442
9	1/2 Gal. Skim Milk	$47,156	20 oz. Coffee	35,059
10	20 Oz. Coffee	$46,048	Sport Water	33,160

indeed it is ranked highest by any reasonable measure we have tried. In general, there is a great deal of agreement between the two lists. The one high-profile difference is Quarter Dogs, which are hot dogs sold for 25 cents every day from midnight until closing. They are incredibly popular among students and long lines form in the store every day at midnight. Their relatively low cost keeps them off the high-revenue list. Other particularly notable products are Coffee and Water, which appear on both lists multiple times.

Finally, Figure 2.1 shows the top 50 association rules in our data (by confidence) at 0.01% support. In the picture, `B <- A` indicates that whenever A is bought, B is often bought. The numbers in parentheses show the confidence of a given rule. Whereas Table 2.3 shows popular products, Figure 2.1 shows popular combinations of products. Contrasting the two allows us to draw some quick conclusions. Bulk Candy and Quarter Dogs, which appear very prominently in the list of popular products, do not appear at all in the list of association rules. Our background knowledge suggests different reasons for this: Bulk Candy is a popular product without distinct ties to any other products. It is bought a lot, but not bought frequently enough with any one product for a true relationship to form. Quarter Dogs, on the other hand, are frequently bought with water and other drinks, but their limited availability (usually only two hours a day) means that the confidence of these relationships is too low to show up on our list.

Bagel and Coffee, on the other hand, are bought frequently with a number of other products. Thus, the list provides a cursory notion that these two products may be particularly influential in the store. In fact, Bagel appears in a majority (31) of the 50 rules, but many of these rules are *redundant* taking the form of {Bagel, X} $\rightarrow$ Cream Cheese or {Cream Cheese, X} $\rightarrow$ Bagel. The presence of such redundant rules and other issues will be discussed at greater length in Chapter 3.



```
DAYQUIL             <- NYQUIL                                   (33.9)
DEAN_PT_SKIM_CHUG   <- KELL_SMART_START_IND                     (39.2)
COFFEE              <- NEWSPAPER_SB_TRIB_TU                     (29.9)
QUICKSNAP_FLASH_800 <- QUICKSNP_OUTDOOR                         (31.0)
EGGS_CSPRING_8CT    <- WESSON_CANOLA_OIL                        (37.1)
DEAN_PT_SKIM_CHUG   <- GM_TOTAL_RAISIN_BRAN                     (42.9)
MARUCHAN_RAMEN_CHIX <- MARUCHAN_RAMEN_ORIEN                     (30.9)
DEAN_PT_SKIM_CHUG   <- SPECIAL_K_BERRIES                        (33.0)
DEAN_PT_SKIM_CHUG   <- KELL_RASN_BRAN_CRNCH                     (42.8)
BC_FROSTING__DXCHOC <- DH_YELLOW_CAKE_MX_18                     (42.9)
VAULT_ZERO          <- VAULT_SODA                               (44.0)
VAULT_20_OZ         <- VAULT_SODA                               (58.0)
COFFEE              <- NUT_BREAD_SLICE                          (40.8)
BAGEL               <- CREAM_CHEESE                             (93.9)
DIET_COKE_20_OZ     <- KIT_KAT_BAR NEWSPAPER_CHICAGO_TR         (91.7)
CHICAGO_TRIBUNE     <- KIT_KAT_BAR DIET_COKE_20_OZ              (73.3)
KIT_KAT_BAR         <- CHICAGO_TRUIBUNE DIET_COKE_20_OZ         (40.5)
DIET_COKE_20_OZ     <- YORK_MINT_PATTIES NEWSPAPER_CHICAGO_TR   (88.7)
CHICAGO_TRIBUNE     <- YORK_MINT_PATTIES DIET_COKE_20_OZ        (77.0)
YORK_MINT_PATTIES   <- CHICAGO_TRIBUNE DIET_COKE_20_OZ          (57.7)
COFFEE              <- CHICAGO_TRIBUNE BAGEL                    (85.6)
BAGEL               <- CHICAGO_TRIBUNE COFFEE                   (73.9)
BAGEL               <- NEW_YORK_TIMES CREAM_CHEESE              (95.7)
CREAM_CHEESE        <- NEW_YORK_TIMES BAGEL                     (58.4)
COFFEE              <- NEW_YORK_TIMES BAGEL                     (75.2)
BAGEL               <- NEW_YORK_TIMES COFFEE                    (41.5)
POP_2LT_COKE        <- POP_2LT_SPRITE POP_2LT_DIET_COKE         (47.8)
POP_2LT_SPRITE      <- POP_2LT_COKE POP_2LT_DIET_COKE           (43.9)
BAGEL               <- MINUTE_MAID_APPLE_JUICE CREAM_CHEESE     (96.7)
CREAM_CHEESE        <- MINUTE_MAID_APPLE_JUICE BAGEL            (30.2)
BAGEL               <- DEAN_PT_CHOC_CHUG CREAM_CHEESE           (96.9)
CREAM_CHEESE        <- DEAN_PT_CHOC_CHUG BAGEL                  (36.3)
BAGEL               <- DEANS_PT_2_PERC CREAM_CHEESE             (97.1)
CREAM_CHEESE        <- DEANS_PT_2_PERC BAGEL                    (35.3)
BAGEL               <- SM_DONUT CREAM_CHEESE                    (93.6)
CREAM_CHEESE        <- SM_DONUT BAGEL                           (36.5)
BAGEL               <- DEAN_PT_SKIM_CHUG CREAM_CHEESE           (98.2)
CREAM_CHEESE        <- DEAN_PT_SKIM_CHUG BAGEL                  (34.7)
BAGEL               <- HOT_TEA__CUP+BAG CREAM_CHEESE            (92.9)
BAGEL               <- JUICE_MM_OJ_15.2_OZ CREAM_CHEESE         (97.8)
BAGEL               <- CREAM_CHEESE DIET_COKE_20_OZ             (93.4)
CREAM_CHEESE        <- DIET_COKE_20_OZ BAGEL                    (32.6)
BAGEL               <- CREAM_CHEESE WATER_CG_1_LTR              (92.6)
BAGEL               <- CREAM_CHEESE WATER_CG_SPORT              (93.5)
BAGEL               <- CREAM_CHEESE FRUIT_APPLE_ORANGE_B        (91.3)
CREAM_CHEESE        <- FRUIT_APPLE_ORANGE_B BAGEL               (34.6)
BAGEL               <- CREAM_CHEESE WATER_DASANI_20_OZ          (95.1)
CREAM_CHEESE        <- WATER_DASANI_20_OZ BAGEL                 (33.7)
BAGEL               <- CREAM_CHEESE SALAD_BAR-BULK              (93.3)
BAGEL               <- CREAM_CHEESE COFFEE                      (96.6)
```

Figure 2.1. Top 50 Association Rules

The tables and figures presented in this chapter represent a fairly typical first attempt at any market basket analysis. Our discussion should familiarize the reader with some of the important products and product relationships that we will examine in the remainder of the paper. We have outlined some of the conclusions that can be drawn and some of the problems that arise from a basic analysis framework focusing on support and confidence. These issues, and the methods we propose to address them, are the focus of the next two chapters.

CHAPTER 3

ASSOCIATION RULES

The primary approach in computer science literature for analyzing market basket data is the discovery and interpretation of *association rules.* The association rules problem [2] is defined as follows:

Given a threshold s, called the *minimum support* and a threshold c, the *minimum confidence*, find all rules of the form $A \rightarrow B$, where A and B are sets of products, such that:

1. A and B appear together in at least $s\%$ of transactions.
2. B occurs in at least $c\%$ of the transactions in which A occurs.

Sets of products are typically called *itemsets*, itemsets of size k are called k-itemsets, and sets that meet the minimum support criterion are typically called *large* or *frequent* itemsets. An association rule is said to be *supported* in a transaction database if it meets both the minimum support and minimum confidence criteria.

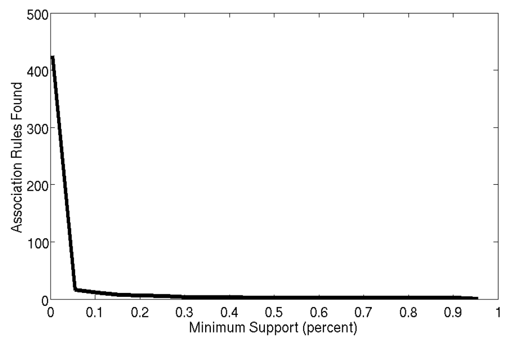
Algorithms for discovering association rules typically operate in two phases: *itemset generation* and *rule discovery.* The itemset generation phase enumerates all itemsets that are frequent at $s\%$ support, and the rule discovery phase produces from the frequent itemsets any rules with at least $c\%$ confidence.

One of the first efficient algorithms for discovering association rules in large databases was Apriori [2], which achieved substantial performance improvements over prior approaches by exploiting the *downward closure* property of frequent itemset generation, stated simply as: *If an itemset I is supported, then all subsets of I are supported.* It is easy to see that the downward closure property holds: if the set $A = \{a, b, c\}$ is not supported then no set $B = \{a, b, c, d\}$ can ever be supported, because the set B cannot possibly occur more frequently in the database than the set A.

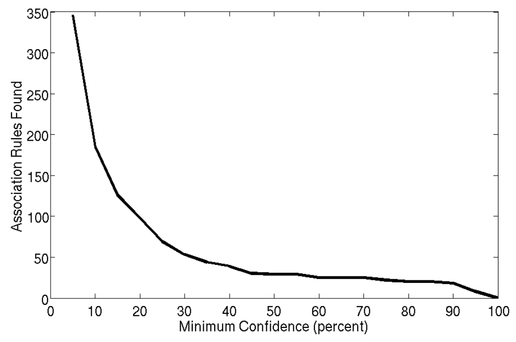
Following Apriori, a number of other candidate-generation algorithms such as Eclat [72], Dynamic Itemset Counting (DIC) [12], FP-Growth [32] improve on the performance of Apriori under certain circumstances by, for example, reducing the number of passes over the transaction database.

As association rules came into widespread use, researchers noticed that understanding the rules themselves was not a trivial matter. First, there is no obvious method for choosing appropriate support and confidence thresholds. If the thresholds are chosen too high, interesting associations may be missed. However, if they are chosen too low, the user may be inundated with thousands of weak rules that do not represent meaningful associations.

To illustrate the magnitude of this problem, and in particular the difficulty of isolating appropriate thresholds, we discovered association rules in our own data at varying levels of support and confidence. Figure 3.1(a) shows the number of association rules discovered at 10% confidence as support ranges from 0.005% to 1%. The number of rules is negligible above 0.1% support but increases very rapidly below 0.05%. Figure 3.1(b) shows a similar result, this time holding support steady at 0.01% and varying confidence from 5% to 100%. The increase appears substantially less drastic but this is largely due to a number of redundant multi-item associations with exceptionally high confidence. Note that from 10% to 5%, the number of rules more than doubles. Taken together, Figures 3.1(a) and 3.1(b) show that association



(a) Holding confidence at 10% and varying support.



(b) Holding support at 0.01% and varying confidence.

Figure 3.1. Number of associations discovered at varying levels of confidence and support.

rules can be incredibly sensitive to the choice of support and confidence parameters.

A second practical issue is that transaction databases often contain hundreds or thousands of association rules at reasonable levels of support and confidence [41], and many of those rules are either redundant or simply obvious [41].

A number of different techniques have been developed to address this issue. The first is the mining of *maximal* [31] or *closed* [70, 71] itemsets. An itemset I is closed if no superset of I has the same support as I and I is maximal at $s\%$ support if no superset of I has at least $s\%$ support. The effectiveness of these methods in practice depends on the composition of the data. If a dataset supports several rules

$A \rightarrow B, AC \rightarrow B, AD \rightarrow B, ...$ maximal itemset mining will prune the first of these rules but leave the others. If the first rule arises as a consequence of the others, then the pruning is useful. However, if the additional products C, D, etc. co-occur incidentally with the popular products A and B, then the remaining rules are the ones that are redundant. Furthermore, the number of pruned rules may be very small compared to the number of rules remaining.

As a practical example, our data supports 168 rules at 0.01% support and 10% confidence. Of these rules, 155 are maximal. Decreasing support to 0.005%, the numbers increase to 385 and 340 respectively. In both cases, all the itemsets are closed. Also, of the original 168 rules, 38 take the form {Cream Cheese, X} → Bagel or {Bagel, X} → Cream Cheese. Within these rules, all are closed and only two, (Bagel → Cream Cheese and Cream Cheese → Bagel) are not maximal. This result suggests that, in addition to pruning very few rules, maximal itemset mining, in our case, prunes incorrectly. Those 36 rules involving Bagel and Cream Cheese can be very effectively explained by the very strong relationship between Cream Cheese and Bagel.

These findings may seem to be in conflict with prior research on closed and maximal itemsets. For example, in [70], the author claims that the mining of closed itemsets can reduce the number of association rules found in a dataset by as much as a factor of 3,000. Those experiments, however, were conducted on generic machine learning datasets rather than market basket datasets. Furthermore, the results were obtained by mining association rules with multi-item consequents, which is rarely done in practice because it is known to produce redundant rules. We believe based on our results that maximal and closed itemset mining are of limited use for practical market basket analysis.

A second approach to combat the explosion of uninteresting rules is to calculate additional *interestingness* measures [59] on the rules. These measures can then be used to either rank the rules by importance (and present a sorted list to the user) or as an additional pruning criterion. How exactly interestingness is determined varies by measure, but many existing measures take the approach that interestingness is "deviation from independence." For example, one of the simpler such measures, the *lift* of a rule $A \rightarrow B$ is defined as:

$$L(A \rightarrow B) = \frac{P(AB)}{P(A)P(B)} \tag{3.1}$$

where P(X) is the proportion of transactions in which X occurs. Note that if purchases of A and B are perfectly independent, the lift $L(A \rightarrow B) = 1$. If A and B appear together more often than we would expect under independence, the lift is greater than 1, and otherwise it is less than one.

This notion of interestingness is intuitively reasonable, but there are over 20 such measures defined in the literature [9, 23, 59], and it has been shown that they tend to rank rules very differently [59]. Therefore, it is not obvious a priori which measure, if any, will elevate the desired rules to the top, or at what level of interestingness the useful rules will end.

To study this phenomenon in our own data, we found association rules at 0.01% support and 10% confidence and ranked them according to each measure given by [59]. Table 3.1 shows information about the top ten rules by average rank. Three of these rules are ranked best by at least one measure, and one ranks as badly as 134. Even the relationship between bagel and cream cheese, which is the strongest in the data (support almost 1%, confidence 93%) is ranked 128th by one measure. This variability implies that interestingness measures are useful mainly when experience or background knowledge is available to assist in the selection of an appropriate measure.

An alternative approach to searching through large sets of rules is to impose a

TABLE 3.1

HIGH, LOW, AND MEAN RANK AND STANDARD DEVIATION OF RANKS FOR THE TOP 10 RULES BY AVERAGE RANK AMONG THE 21 INTERESTINGNESS MEASURES IN [59]

Rule	High	Low	Mean	St. Dev.
1	1	128	18.07	33.79
2	3	65.5	21.85	19.61
3	6	71	24.95	15.77
4	2	96	28.05	25.67
5	8	96	28.85	22.90
6	1	129	31.37	36.32
7	1	133	32.02	42.90
8	16	70	33.40	13.41
9	3	69	34.37	43.42
10	2	134	34.47	44.55

pruning criterion that preserves only the strongest relationships in the data. Hyperclique Patterns [68] discover tightly-knit groups of items, potentially at a much lower level of support than is feasible with association rules. A hyperclique pattern P at support s and *h-confidence* c is a set of items $P = \{P_1, P_2, \ldots P_n\}$ such that for each association rule $P_i \rightarrow P_1 \ldots P_{i-1}, P_{i+1} \ldots P_n$, the support of the rule is at least s and the confidence of the rule is at least c. The advantage of hyperclique patterns is that they are able to discover relevant patterns without an explosion of the rule-space, as might be with using vanilla association rules. However, the criteria that define a hyperclique pattern are very strong in practice, and it is difficult to find hyperclique patterns of any substantial size in market basket data. For our data, there are no hyperclique patterns of size greater than two, even at support as low as 0.005%. Therefore hyperclique patterns, while effective at discovering certain strong relationships, are hardly a sufficient analysis technique on their own.

Association Rules Networks [15, 16, 51] reduce the ruleset by focusing solely on rules related to a single product. More specifically, given a set of association rules R and a target product z, the association rules network ARN(R, z) is the unique directed hypergraph G satisfying the following properties:

1. Each directed hyperedge in G corresponds to a rule in R with a single-item consequent.
2. There is a hyperedge corresponding to a rule whose consequent is the target product z.
3. The target product z is reachable from every vertex v in G.
4. No vertex $v \neq z$ is reachable from z.

Generally speaking, an ARN shows the extent to which rules "flow into" the target product. The resulting network can show both direct and indirect associations of the target product z. However, Association Rules Networks can be quite sensitive to the choice of target product, and there is no obvious proper choice. As a result, one must have some idea of the products he or she is interested in before association rules networks are applicable. We explore the integration of association rules networks into a broader strategy for market basket analysis in Chapter 5.

The above discussion suggests that no technique currently available in the literature sufficiently solves the problem of finding meaningful relationships in large transaction databases. This deficiency motivates our discussion of network methods for market basket analysis, which is the subject of the next chapter. We do not claim to definitively solve the market basket problem. However, we will show that as a first exploratory step, our techniques can discover expressive relationships from which we can draw direct conclusions about the nature of customer behavior in a store.

CHAPTER 4

NETWORK METHODS

For the remainder of this work, we will focus on *networks* of products and the implications of network representations of transactional data. In general, define a product network as a graph $G = (V, E)$, where vertices v represent products and an edge e connects two products that tend to be bought together.

Intuitively, the structured nature of networks should afford certain advantages over traditional association rules. First, networks can capture *indirect* relationships between products in addition to the direct relationships captured by association rules. Indirect relationships of the form $A \rightarrow B \rightarrow C$ may be of interest, for example, in pricing decisions, where we may conclude that lowering the price of A will increase sales of B and indirectly increase sales of C. Second, groups or *communities* of several closely related products should express meaningful relationships more concisely than association rules, reducing the need to search through large sets of irrelevant rules.

This chapter addresses *the process by which we propose to analyze networks of products.* In our presentation of results, we focus on the ability of product networks to concisely represent complex relationships, as well as their ability to support financial conclusions from the data as we feel these are the primary strengths of our approach compared to other published methods. In this chapter and all subsequent chapters we present results based not on the entire dataset (April 2005 to February

2007 transactions), but rather from the calendar year 2006. The reason for this is that the store retired certain products and added other products from year to year. The 2006 data was the largest consistent subset available to us.

4.1 Constructing a Network of Products

The first step when analyzing transactional data as a network is to construct the network. In doing so, we follow an approach similar to that of several other authors [33, 41, 50]: each node in the network represents a product, and an edge appears between any two products that have been bought together in a transaction. This method of construction, though intuitive, imposes a structure on product networks that is somewhat unlike that of other types of interaction networks.

It has been well-established, for example, that social networks often have *heavy-tailed* degree distributions, meaning that there are very few *hubs*, connected to hundreds or thousands of others while the vast majority of nodes have few neighbors. In product networks, there is an additional distribution of interest. Whereas interaction networks are typically unweighted, meaning that each link between nodes is either "present" or "absent" our product networks have distinct edge weights.

Therefore, in addition to analyzing the global degree distribution which sheds light on the *variety* of items with which different products are bought, we can explore the distribution of edge weights incident to any one product for insight into the *strengths* of the ties that a product has with its neighbors.

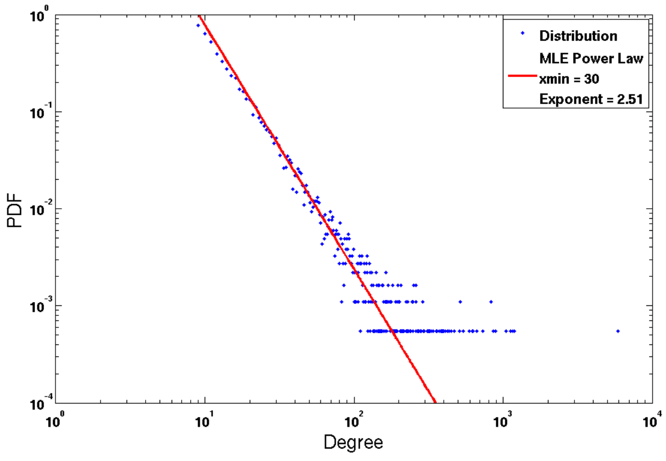
In our data, we find heavy-tailed behavior both locally and globally. Figure 4.1 shows both the degree distribution of the entire network and the distribution of the weights of edges adjacent to Bulk Candy. We see that both plots decay at least linearly on a log-log scale. Not only does the network have a heavy-tailed degree distribution, but individual products do as well. This result suggests that the average product is bought infrequently with the majority of its neighbors and frequently with only a few.

Figure 4.1 hints at the most difficult aspect of product networks in practice. They differ from more commonly-observed networks for one simple reason: the presence of an edge does not necessarily imply a confirmed relationship between products. Other networks, such as networks based on citations or phone calls, do not suffer from this problem to nearly the same degree.

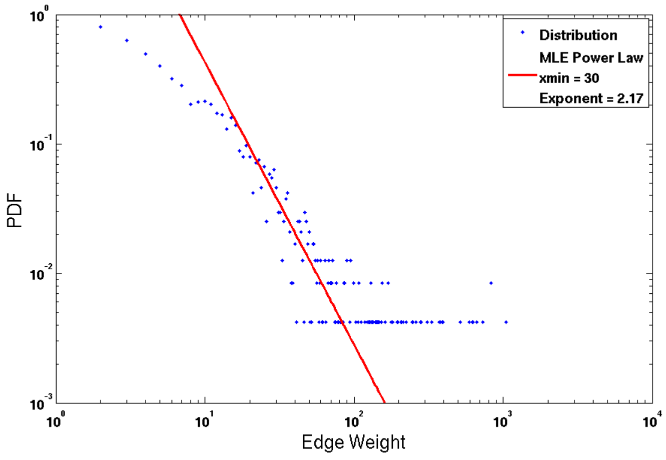
In citation networks, for example, two nodes linked together by an edge are necessarily related: if one paper cites another, there is a reason. A cell phone network will have a small number of incidental links, (wrong numbers, telemarketing, or random personal business), but most of the time, when one person calls another, it implies a connection between them. Product networks are different. Simply because a person buys paper towels and spaghetti sauce in the same transaction does not entail a common motivation for the two purchases. Worse, a person who buys several unrelated items in a single transaction will cause a clique to form between them, despite the absence of any true relationship.

As a result, product networks are very *dense*, with a large number of connections per node, but many of these edges are meaningless: representing spurious associations generated by chance. Our network contains 2,248 products and almost 250,000 edges between them. However, over 150,000 of these edges have a weight of one, meaning the two products were bought together only once in the entire year 2006, and over 235,000 have weight less than 10. These extremely low-weight edges are common but are unlikely to represent strong relationships.

In order to remove some of the noisy edges created by coincidental purchases and improve the quality of our subsequent analysis, we establish a minimum threshold σ, such that an edge exists between two products only if they have been bought together



(a)



(b)

Figure 4.1. Degree distribution for (a) the entire network and (b) the neighbors of a single product.

at least σ times. This is analogous to choosing a minimum support threshold for association rules. Note that, in the pruned network, the weight of an edge (p_1, p_2) remains the number of transactions in which p_1 and p_2 appear together.

Having described the construction of a product network and studied some of its properties, we now turn our attention to the analysis of the product space. Since the primary focus of market basket analysis is the discovery of relationships between products, we need to find groups of products whose structure or position within the network reveals useful information about the store itself.

Many real-world networks naturally contain *communities*: groups of nodes that are more strongly connected to each other than they are to the rest of the network. Often, these communities have an easily-interpretable significance. In a cell phone network [57], for example, communities may represent families or circles of friends. Conversely, in a network of web pages, [40], they may represent sites devoted to a common interest or theme. Community detection has been applied successfully in a numerous fields of science, ranging from social network analysis [57] to biology [4, 24] and molecular physics [42]. It seems logical to expect that communities of products, since they are mutually strongly-connected, would be of particular interest. Therefore, the remainder of the chapter will focus on the problem of *community detection* in product networks, and show how communities of products can be used to gain insight into the behavior of customers in a store.

4.2 Community Detection

Community detection is the process of finding strongly-connected groups of nodes (*communities*) in a network. How exactly to define a *strong* community is still something of an open question. It is generally agreed that communities should be tightly connected to each other (many in-community edges), and relatively weakly connected to the rest of the network (few out-of-community edges). In this vein, several mathematical notions of community have been proposed. Radicchi et. al [55] define communities in the "strong" sense as groups of nodes in which each node has more in-community connections than out-of-community edges. In other words, if we define k_i as the degree of node n_i, k_i^{in} as the number of in-community edges adjacent to n_i and k_i^{out} as the number of out-of-community edges adjacent to n_i, a set of nodes $N = \{n_1, \ldots, n_j\}$ is a community in the strong sense if the following relation holds:

$$k_i^{in} > k_i^{out} \; \forall i \in N \tag{4.1}$$

Similarly, a group of nodes represents a community in the "weak" sense if the entire *group* has more in-community edges than out-of-community edges. This condition is expressed mathematically as:

$$\sum_i k_i^{in} > \sum_i k_i^{out}. \tag{4.2}$$

Conversely, Newman [48] proposes to measure the strength of a community by a quantity known as *modularity*, which measures the extent to which the number of in-community edges within a set of nodes N exceeds the number which would be expected in a completely random network with the same degree distribution as the network in question. The modularity Q of a community is defined as:

$$Q = \frac{1}{2m}\left(\sum_{n_i \in N}\left(k_i^{in} - \sum_{n_j \in N}\frac{k_i k_j}{2m}\right)\right). \tag{4.3}$$

where m is the total number of edges in the network. The precise definition of community suggested by Equation 4.3 is that a set of nodes N is a *community* if the number of in-community edges within N is greater than the number expected by random chance. In other words, if $Q > 0$.

This notion is very intuitive. If a set of communities has a large fraction of its edges falling within communities (and therefore a relatively small fraction falling between communities), then that particular community decomposition probably represents a strong community structure. However, if the number of in-community edges is similar to what would be expected in a random graph, then no substantial community structure can be inferred regardless of the communities' density.

The most common class of community detection algorithms are *agglomerative heirarchical* algorithms [8, 46, 53, 55]. These algorithms proceed iteratively, beginning with each node in the network as its own community. Then, at each iteration, they merge two communities according to some objective function. The process terminates when the network has been joined into one community. An alternative class of algorithms (*divisive* algorithms [47]) start with the entire network as one community and repeatedly divide communities into two until each node is its own community. For either type of algorithm, the final result is a tree known as a *dendrogram* (Figure 4.2). Whenever communities are merged, this creates a new "level" in the dendrogram, and each level represents a distinct partition of the network into communities. Upon completion of the algorithm, one needs to choose the point at which to cut the dendrogram, i.e. the level of the dendrogram to choose as the final output of the algorithm.

Many algorithms [8, 47, 46] choose to use the *maximum-modularity* partition as the final partition, where the modularity of a partition is defined as the sum of the modularities of the individual communities as defined in Equation 4.3.

Regardless of the definition employed or the community-detection strategy used, the result of the community detection process is similar: Given a graph G, the algorithm will produce a set of disjoint subgraphs $\{G_1, \ldots, G_n\}$ such that the vertices of the individual G_i are tightly connected according to some measure. The application

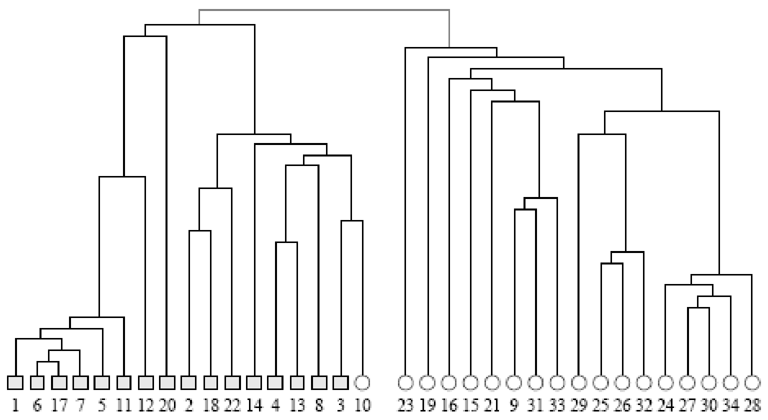


Figure 4.2. Example of a dendrogram. Taken from [46]

.

to market basket analysis is clear: isolating tight communities within the network of products will allow us to identify strong relationships among the products and therefore meaningful correlations in customer purchase behavior. Furthermore, because communities can be arbitrarily large, they should be able to represent these relationships more expressively and with less redundancy than ordinary association rules.

4.3 Detecting Communities of Products

In order to evaluate the effectiveness of community detection for isolating relationships between products, we built a product network in the manner described in Section 4.1, setting minimum support $\sigma = 0.01\%$. We present communities discovered with the algorithm of Blondel et al. [8], which is one of the more scalable algorithms developed to date.

The algorithm itself is a modularity optimization heuristic that operates in two phases. The first phase individually considers all nodes i in the network. For each neighbor j of i, it calculates the difference in modularity ΔQ associated with removing i from its current community and placing it in the same community as j. It then assigns i to the community for which ΔQ is maximized, understanding that the ΔQ associated with leaving i in its current community is zero.

The first phase is repeated as many times as necessary until no further improvement of the modularity is possible. The second phase of the algorithm contracts each community into a single vertex, where the weight of the edge between community i and community j is the sum of the weights of all edges between vertices in community i and vertices in community j. Within-community edges, then, become self-edges for the vertex i.

At this point, the computation returns to the first phase, optimizing modularity on the contracted network. The two phases are repeated until no further improvement in the modularity is possible, or until the network has been merged into one giant community. The algorithm is hierarchical and, because it is a modularity-optimization algorithm, the maximum-modularity partition is chosen as the final partition of the network.

Though we consider only one algorithm here, our studies have shown that the choice of algorithm makes only minor differences in the communities discovered. We will discuss the effect of algorithm choice further in Section 4.5.

Overall, there were 17 communities discovered in the pruned network, ranging in size from two products to over 70. In order to expedite and formalize our analysis, we sought to define a measure of interestingness for communities. Specifically, we wish to answer the question: *given a set of communities in a product network, which are most useful to a human analyst?*

Intuitively, the *utility* of a community can be determined by two opposing forces: *information* and *information density.* A useful community will be large enough to provide a substantial insight into customer behavior, but small enough to be human-interpretable. Finally and most importantly, a useful community should contain strong associations, so that any community that is highly regarded by our measure represents a significant relationship among its constituent products.

One natural candidate for an interestingness measure of communities is the modularity as defined in Equation 4.3. However, one property of this modularity measure makes it largely unsuitable for the task: modularity is nothing more than the sum of a series of contributions from individual nodes. As such, the communities with the greatest modularity in a given partition tend to be very large. In our data the two highest-modularity communities are of size 39 and 77 respectively. Communities of this size tend to be too large to expose the true relationships among products.

We propose a different quantitative measure which overcomes this limitation. Let $G_i = (V_i, E_i)$ be the subgraph formed by the vertices within community i. We wish to define suitable functions $I(G_i)$ and $D(G_i)$ to quantify the concepts of information and information density described above, so that we may present the utility $U(G_i)$ of a community as a suitable combination of information and information density.

Intuitively, the information within a community is conveyed by its edges. Strong edges indicate that two items are strongly related, while weak or non-existent edges indicate insignificant relationships. Thus, given a function $F(e)$ which quantifies the strength of an edge, one natural measurement for the information present in a community is simply the sum of the strengths of its constituent edges. In other words:

$$I(G_i) = \sum_{e \in E_i} F(e). \tag{4.4}$$

In seeking to define the notion of information density, we note that several potential definitions of density follow immediately from Equation 4.4. One could, for example, define density as the *average strength* of an edge in E_i: $D(G_i) = \frac{I(G_i)}{|E_i|}$.

However, such a measure would completely disregard nonexistent edges (in our scenario, pairs of products that are never bought together), and as such may lend very high scores to very sparse communities. Similarly, one might define the density as the information per *potential edge* in E_i: $D(G_i) = \binom{|V_i|}{2}$. This definition is attractive because it provides a normalized measure. If $F(e)$ is bounded, say in the range $(0, 1)$, $D(G_i)$ would be similarly bounded. However, the number of potential edges within a community is quadratic in $|V|$ and it is unreasonable to assume, even for strong communities, that all these edges will be strong. As a result, this measure tends to rate two-item communities orders of magnitude higher than reasonably dense communities of even six or seven products.

Thus we propose a compromise, defining the information density $D(G_i)$ as the amount of information per *node* of G_i:

$$D(G_i) = \frac{I(G_i)}{|V_i|}. \tag{4.5}$$

Finally, we define the overall utility of community i as the harmonic mean of the above-defined quantities:

$$U(G_i) = \frac{2I(G_i)D(G_i)}{I(G_i) + D(G_i)}. \tag{4.6}$$

The one question that remains, though, is: what is the most suitable function $F(e)$ to represent the strength of an edge? We propose to define the strength of the edge between two products p_1 and p_2 as the *confidence* (see Chapter 3) of the association rule $p_1 \rightarrow p_2$. In other words, $F(p_1 - - - p_2) = P(p_1|p_2)$. Since we treat edges as undirected, we evaluate both $F(p_1 - - - p_2)$ and $F(p_2 - - - p_1)$ for all edges. The general information equation given in Equation 4.4 then becomes:

$$I(G_i) = \sum_{(p_1,p_2)\in E_i} P(p_1|p_2) \tag{4.7}$$

and equations 4.5 and 4.6 are defined accordingly. We could have chosen, in lieu of confidence, a number of measures for the strength of an edge. The choice of confidence is convenient for two reasons. First, it is bounded. An unbounded measure, which can take values up to infinity, may assign an unreasonably high value to a community based on a single relationship. Second, it is *null invariant* [59], meaning that its measure of the relationship between A and B depends only on transactions containing A or B. To see why null invariance is important, consider a concrete example.

Assume two products A and B only sell for one month of the year. During this month, A appears in 1,000 transactions, B appears in 500 transactions, and the two appear together in 250 transactions. The confidence of $A \rightarrow B$ is 25% and the confidence of $B \rightarrow A$ is 50%. If the store sees 100,000 transactions a month, then the support of the rule $A \rightarrow B$ is 0.25% during the time that the products actually sell. However, since the support of a rule depends on the number of transactions in the transaction database, and not just transactions containing A and B, this support decreases as more data is considered. In three months of data, the support has fallen below 0.01% even though the additional data says nothing about the actual strength of the relationship between A and B. A null-invariant measure such as confidence, by contrast, preserves the strength of the relationship even in the presence of irrelevant transactions. While the above example is admittedly extreme, this concern is quite significant in practice. Breakfast products that sell heavily together in the morning hours, for example, should not be penalized simply because they are sold out by noon.

By substituting the definitions of $I(G_i)$ and $D(G_i)$ into Equation 4.6 and simplifying, we find that:

$$\begin{aligned} U(G_i) &= 2\frac{I(G_i)}{|V_i|+1} \quad \mathit{or} \\ U(G_i) &= 2D(G_i)\frac{|V_i|}{|V_i|+1}. \end{aligned} \tag{4.8}$$

Clearly then, in the limit of large $|V_i|$, our utility measure $U(G_i)$ approaches a constant multiple of our density measure $D(G_i)$. To see how our utility measure captures the intuition laid out earlier, note that if any two communities have equal density, the larger of the two communities will have the higher utility. This is exactly what we want: our primary concern is that "useful" communities should be tightly-connected, but given two communities of equal or roughly-equal strength, we prefer the community that is larger, since it captures a relationship among the greatest number of products.

The formulation in Equation 4.8 converges very rapidly to the density of the community, since a two-item community with density d already has 2/3 the utility of an infinitely-large community with the same density. If desired, however, the formula can be tuned to penalize communities of arbitrary size. Defining $U(G_i) = D(G_i)\frac{|V_i|}{|V_i|+k}$, for example, assigns half as much utility to a community of size k as an infinitely-large community of equal density.

It is worth noting that, because the computation in Equation 4.7 depends on the actual number of edges present in the community, our utility measure depends somewhat on the method of graph construction. In other words, if we allow an edge between any two products that are bought together, the computation will be different than if we restrict edges to products bought together at least 100 times. The end result of this is that our utility measure is *not* comparable across different network constructions. We do not consider this to be a significant issue because it is designed to help a human analyst assess one set of communities.

While our utility measure is designed for product networks, we believe that the tradeoff between size and density is very general and that, in principle, Equation 4.6 could be applied to other domains. In an email network, for example, if one defines *information* as the frequency of email correspondence between members of the community over some time period, an analog of Equation 4.6 follows naturally.

4.4 Experimental Evaluation

We evaluated each of the 17 communities discovered earlier using the utility measure defined in Equation 4.6 and the results appear in Figure 4.3. The calculated utilities range from very near zero to slightly over 1. We see that a large number of communities have very low utility, with five communities falling in the first bin (below 0.14). At the other end of the spectrum, there are two communities rated substantially higher than the others (1.01 and 0.92 respectively). We find that highly-rated communities are generally very well-connected and have a clear purpose.

Figure 4.4(a) shows the most highest-rated community ($U(G) = 1.01$), a community of chips and salsa. The community is very densely connected, and it carries a very clear message: that people often buy chips and salsa together, and yet is small enough for a human to easily interpret. The community is nearly bipartite, with chips connecting only to salsa and salsa connecting only to chips. The one exception is a single edge between `FL_SALSA_CON_QUE` and `FL_SALSA_MILD`. From a marketing standpoint, the community indicates that chips and salsa are *complementary* products, while the different types of chips (and respectively salsa) are *substitutes* for one another. The salsa con queso, though, is an exception, because it is distinctly different from the other types of salsa available.

Figure 4.4(b) shows the second-ranked community: a collection of eggs and

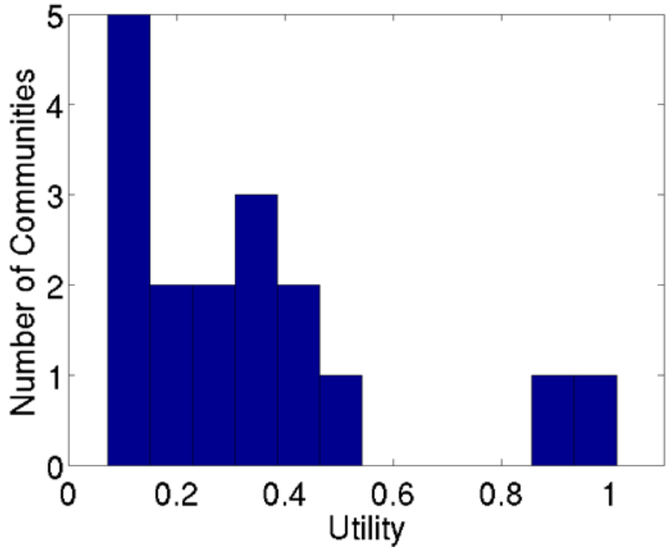
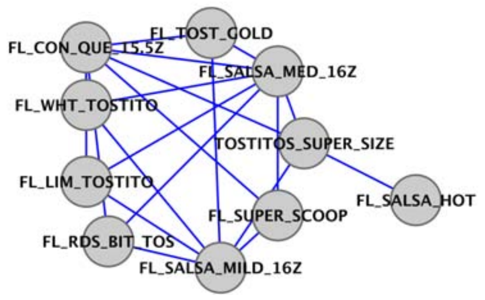
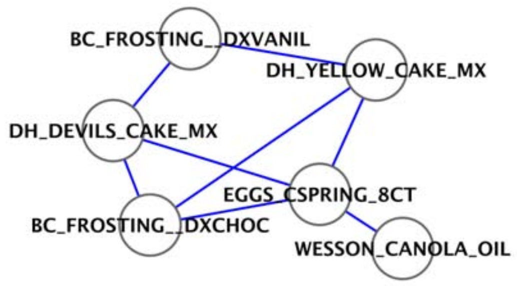


Figure 4.3. Distribution of utility scores for the 17 communities in our data.



(a) Chips and salsa.



(b) Eggs and baking products.

Figure 4.4. The first two communities in our data, ranked by the measure given in Equation 4.6.

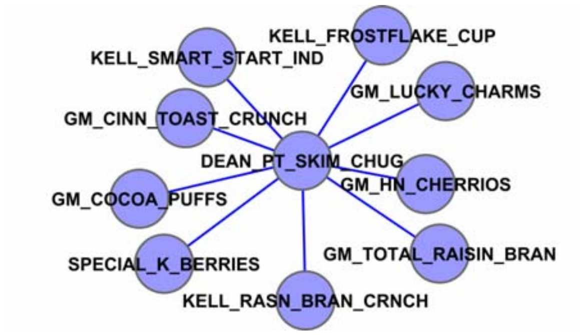
baking products. The structure of the community, with eggs as a hub in the center and the baking items on the periphery, seems to imply that when people buy eggs in our store, they buy them for baking. Further investigation supports this initial hypothesis.

There were 541 distinct products bought with eggs at our store in the calendar year 2006, and in 18.5% of the cases, eggs were bought alone. However, at least one item among the five neighbors appears in over 39% of all transactions containing eggs, which is especially significant because most of the transactions in our store are small. As a case study, we further quantify the impact of this particular community.

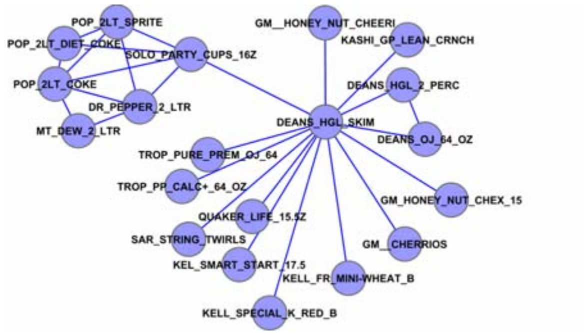
Intuitively, cake mix is the most likely "causal" item in the group (it is unlikely, for example, that people buy frosting because they have a craving for eggs). Therefore, we calculate expected additional sales from each sale of cake mix as: $E(Sales) = P(Eggs|CakeMix) * Price(Eggs) + P(Frosting|CakeMix) * Price(Frosting)$ and find that the store can expect to generate \$2.30 in additional sales from each cake mix sold. Therefore, the store stands to profit from any promotion that increases the sales of cake mix at a cost of less than \$2.30 per transaction. Since cake mix itself costs \$2.69, the expected additional revenue is 85.5% of the item's purchase price. This analysis is admittedly simple, but it demonstrates that communities can help identify profitable promotions in a store.

The third-and-fourth-ranked communities, shown in Figures 4.5(a) and 4.5(b) are communities of cereal and milk. The first of these shows a small container of milk as a hub surrounded by a series of cereals. The second is composed of two nearly-disconnected subgraphs: a hub-and-spoke arrangement of larger milks and cereals and a clique of sodas. The disparate structures are each connected, by one edge, to a single product: plastic beverage cups.

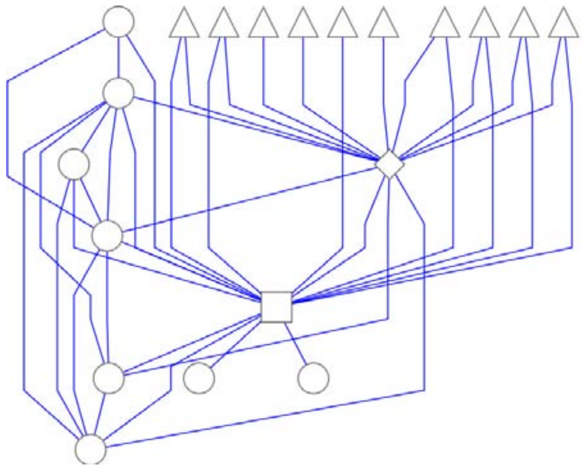
These communities support several conclusions, in addition to the notion that



(a) A community of milk and cereal



(b) A community of milk, cereal, and soda. The soda connects to the rest of the community with only one link.



(c) A community of fruit (diamond), salad (square), and yogurt (triangle).

Figure 4.5. Three more communities.

people buy cereal and milk together. First, we see that there are separate relationships between cereal and milk at two levels: smaller sizes of milk correlate with smaller sizes of cereal, while larger milks relate to larger cereals. Second, the strong mutual correlation among sodas suggests that they are often purchased several at a time, potentially for groups of people, while the disconnection among cereals indicates that people buy them largely for personal use.

The final community of interest is shown in Figure 4.5(c). A community of fruit, salad, yogurt, and soft drinks, it is much less dense than the others and therefore, at number eight, is ranked much less favorably. However, it still contains useful insights. Figure 4.5(c) shows the single fruit product (diamond) connected to nine different yogurt products (triangles). The associations between fruit and any of the individual yogurt products are not strong: none is ranked better than 78th in a list of 168 rules by any of the interestingness measures in [59], but in combination the association is quite powerful.

If all the different varieties of yogurt are combined, they become the most popular product purchased with fruit, and we find that 10% of all fruit sales (by dollar value) come in transactions that contain yogurt, and that 9.5% of all yogurt transactions contains some form of fruit. By contrast, if all varieties of coffee are combined, coffee (the runner-up) occurs in only 8% of fruit transactions, despite the fact that it is bought five times more frequently than yogurt overall. The fruit and yogurt association is a significant relationship whose significance is hidden by the number of yogurt products available.

The largest community discovered in the data contains over 70 products. Composed of many of the store's most popular items, it is too large and dense to be easily interpreted. This fact, in conjunction with the communities mentioned above, suggests that community detection can play a useful supplementary role in market basket analysis. The highly-ranked communities discussed above provide a good deal of insight into the purchases of items as diverse as fruit, cereal, and frosting, but communities reveal very little with regard to the dense "core" of the network: popular products such as coffee, bagels, and water.

Therefore, we propose that community detection be used as a first exploratory step in the analysis process, where it will illuminate the relationships among important, but more peripheral, products. Then, the subsequent association rules analysis can focus more intently on products whose role is not clear within the community decomposition, perhaps with techniques like Association Rules Networks [15, 16, 51].

4.5 The Performance of Other Community Detection Algorithms

To study the impact that algorithm choice has on the types of communities discovered in the data, we found communities in the same network using the Walktrap algorithm of Pons and Latapy [53] and Newman's eigenvector modularity algorithm [47][1]. Even though WalkTrap is not necessarily a modularity maximization algorithm, we chose to present the maximum-modularity partition to avoid arbitrarily choosing one.

The strong communities of eggs, chips, and soda were entirely identical across all three algorithms. In all, differences between the algorithms were minor. Strong observations, if they are not preserved identically, are preserved well enough that the same conclusions can be drawn. The fruit-salad-yogurt community of Figure 4.5(c), for example, is missing three yogurts under WalkTrap, two of which connect to fruit. The remaining seven yogurts, however, are sufficient to make the relationship clear.

Where the differences are more pronounced is in the discovery of smaller communities. The Blondel algorithm failed to distinguish a five-item community of ice

[1]Our implementation did not use a full Kernighan-Lin optimization, but a faster greedy one. The communities discovered are still high-quality.

cream and sundaes, placing it instead in with the largest, most general community. There were 12 items that WalkTrap chose not to place into a community, or rather, to place into single-item communities. Newman's algorithm, by contrast, joined six separate two-item communities (discovered by the other algorithms) into a single community. In general, then, we find that the stronger communities in which we are naturally interested have been preserved across different algorithms and that differences arise primarily when smaller communities or weaker associations are considered.

4.6 Related Work

There exists a limited quantity of prior work that is either tangentially or directly related to the topics of this chapter. Association rules networks, introduced in Chapter 3 combine the notion of association rules with a network representation. However, as we discussed earlier, they are more explanatory than exploratory. Because each ARN is constructed with respect to a single target product, a prospective user must already have a set of interesting products in mind before association rules networks can be applied. Our work, by contrast, is completely exploratory and can discover relationships among a multitude of products without prior knowledge on the part of the user.

Clauset et al. [18] apply community detection to Amazon.com transaction data, but their treatment of the data is very basic. They do not explain any of the communities found or address any practical issues, but merely state that the communities "make sense." Hao et al. [33] develop an application that uses networks to visualize association rules from e-commerce transaction data. Specifically, the application does a force-directed layout of the products in a network, and is capable of performing k-means clustering on the resulting visualization. Our approach is more general, in that community detection algorithms do not require users to specify the number of communities to find. Also, k-means can be sensitive to the initial locations of the cluster centers, which imposes an additional parameter on the process.

4.7 Conclusion

This chapter studies the application of a social network model and community detection algorithms to the problem of *market basket analysis*: the search for meaningful relationships in purchase data. We posit that ties arising from random chance are an especially grave problem in product networks and suggest a thresholding approach for eliminating redundant ties. We find communities in our data that express complex relationships among products. These relationships illuminate the products' role in the store, but may also be potentially profitable. Additionally, we develop a novel measure of the utility of a community of products, and show that it favors communities that are large enough to be useful, but small enough to be interpretable. Finally, Our method of network construction requires only one parameter: a minimum support, and therefore is no more dependent upon parameters than traditional association rules.

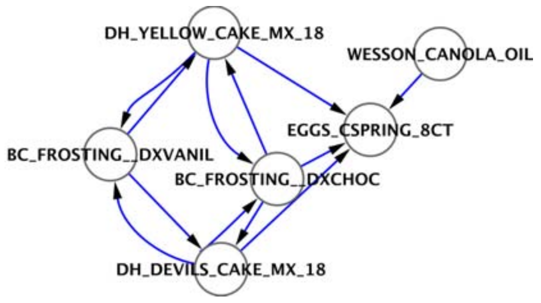
Because community detection seems to be most effective at determining relationships between products outside the core of the network, we recommend the use of community detection as a first exploratory step in conjunction with association rule analysis methods.

CHAPTER 5

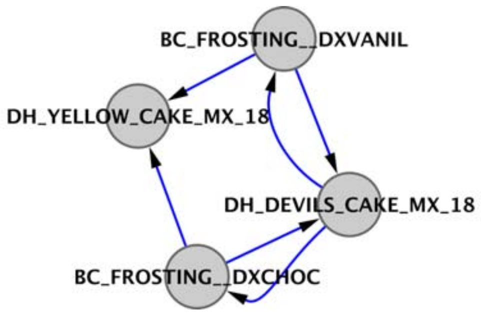
MOVING TOWARD A COMPREHENSIVE ANALYSIS STRATEGY

A great deal of literature has been published on the subject of market basket analysis and survey papers about algorithms [35, 73], interestingness measures [44, 59], and visualization techniques ([7], section 2) abound. In spite of all this effort, however, the community has made no substantive attempt to answer the following basic question: *Given a fresh, unseen market basket dataset what method or set of methods should be employed to obtain quick, actionable results?* There are several possible reasons for this. The first is a dearth of widely-available transaction data, which we alluded to in the introduction. The second is a general lack of diversity in analysis techniques: closed or maximal itemset mining, for example, is not different enough from traditional association rules that the two can complement each other such that one is strong where the other is weak. Finally, most studies that do consider real data are only conducted within a single domain (i.e. supermarkets or online retailers), and so the ability to draw overarching conclusions is limited.

Since we too are confined to a single dataset, we cannot address the third concern, but this chapter addresses the first and the second. In doing so, we call upon not only the techniques developed here in Chapter 4, but also a series of methods developed by other authors. To our knowledge, these methods (*Association Rules Networks* [15, 16, 51] and *Center-Piece Subgraphs* [60]) have never been applied to market basket data, but in the course of our work we have found that they complement



(a) Association Rules Network with z = Eggs



(b) Association Rules Network with z = Cake Mix

Figure 5.1. Two Association Rules Networks from the peripheral community of eggs.

community detection nicely.

The rest of the chapter is organized as follows: Section 5.1 explores practical concerns regarding the use of Association Rules Networks (introduced in Chapter 4), Section 5.2 introduces the Center-Piece Subgraph problem and studies its application in the domain of product networks, Section 5.3 ties together the discussion of this chapter and the prior chapter in order to propose a unified strategy for mining market basket data, and Section 5.4 briefly discusses strategies for parameter selection.

5.1 Association Rules Networks

Recall from Chapter 4 that an *Assocation Rules Network* $ARN(R, z)$ is a directed hypergraph representation of the ruleset R that mops out the direct and indirect associations of the target product z. The concerns we must address when applying Association Rules Networks are 1) How do we choose an appropriate ruleset R? and 2) How do we choose an appropriate item z? The first question essentially boils down to the appropriate choice of support and confidence parameters, and we do not address it here. With regard to the second question, it is natural first to ask: is the

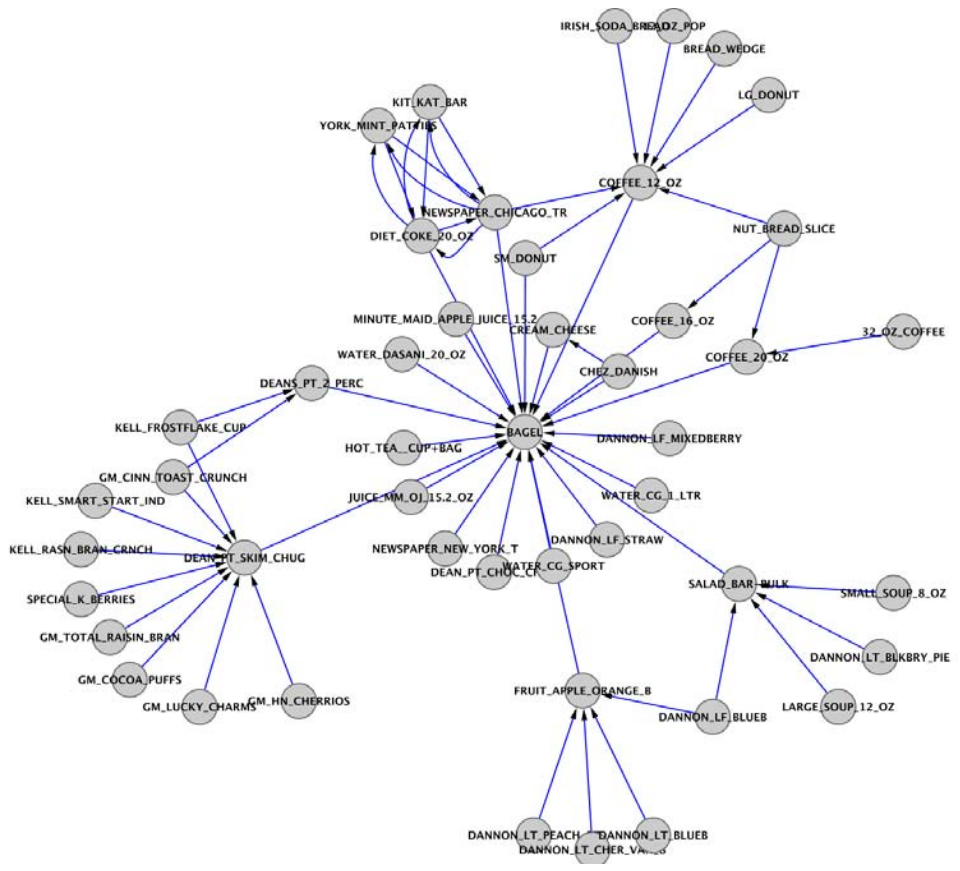


Figure 5.2. Association Rules Network with z = Bagel.

choice of z important? Figure 5.1 shows two different Association Rules Networks. In Figure 5.1(a), eggs are used as the target product, and in Figure 5.1(b), we use cake mix. Even though the two products chosen are related, we see that the resulting networks are quite different. While Figure 5.1(a) shows a relationship between oil, eggs, cake mix and frosting, similar to what was found with community detection, Figure 5.1(b) contains only cake mix and frosting.

While Figure 5.1 makes it clear that the target product z cannot be chosen arbitrarily, it does not shed any light on the process for making an appropriate choice. Figure 5.2 shows a separate Association Rules Network flowing into Bagel. Recall from Chapter 2 that Bagel is one of the most popular items in the store and appears in more association rules than any other. This network is large and expressive, and includes two of the relationships, Fruit-Yogurt and Cereal-Milk that we found with communities earlier (although not to the same detail). It provides an effective visualization of the relationships between some of the more central products in the store.

Many of the items that appear in the network, such as newspapers and donuts, are items that we would intuitively expect to sell well in the mornings. A cursory glance at the network suggests that Coffee may drive food sales during the morning hours and Bagels may drive drink sales. Coffee does not connect to any other drinks, whereas Bagel connects to drinks almost exclusively. Additionally, the network provides insight into the key relationships other core products: milk (with cereal), salad (with soup), and fruit (with salad and yogurt).

To understand why this Bagel network is so much more informative than the Cake Mix network described above, recall the list of association rules given in Chapter 2. There were 31 rules containing Bagel in the 50 highest-confidence association rules, and there are even more in the ruleset from which this network is constructed. The great diversity among Bagel's "neighbors" in the product network allows its ARN to span different segments of the product space.

Thus, it appears that an effective choice for z, when constructing an Association Rules Network from transaction data, is to choose the item that appears in the most rules in the underlying ruleset R. One might consider instead the most popular product in the store, or the item which has been bought with the greatest number of other products. In our data, however, these strategies are less effective. Bulk Candy, which is both the most frequently-sold and bought with the most items, has only two products in its Association Rules Network, and 20-ounce water, another popular product, has none.

The reason for this, of course, is that association rules involving Bulk Candy and Water, which are bought with a stunningly wide variety of items, do not meet the minimum confidence criterion that we have used throughout the paper. We contend, however, that relationships which do not meet the minimum confidence criterion may still be interesting. There are several potential causes of low confidence, but the most relevant in the case of water is *substitution*. There are many different types of water available in the store, and this variety erodes the confidence of certain relationships.

To illustrate the effect of substitution on rule confidence, assume n different products $F_1, \ldots, F_n$ are all substitutes for each other, meaning that they serve roughly the same function F. Furthermore, assume a product $\mathcal{P}$ correlates with items of the function F, such that the confidence of the association rule $F \rightarrow \mathcal{P}$ is c or

$$\frac{P(F\mathcal{P})}{P(\mathcal{P})} = c. \tag{5.1}$$

If the products $F_1, \ldots, F_n$ are all bought equally with $\mathcal{P}$, then for any F_i, the confidence of the rule $F_i \rightarrow \mathcal{P}$ is given by

$$\frac{\frac{P(F\mathcal{P})}{n}}{P(\mathcal{P})} = \frac{c}{n}. \tag{5.2}$$

Thus, the substitution erodes the confidence of the association $F_i \rightarrow \mathcal{P}$ even though the overarching association $F \rightarrow \mathcal{P}$ may be sufficiently interesting. It is also trivially true that substitution erodes the support of any relationship.

This parameter sensitivity is a problem inherent to every technique we have covered thus far. Association rules, ARNs, and the community detection framework we have defined will all systematically fail to find relationships that fall outside the specified support and confidence thresholds for any reason (substitution or otherwise). To address this issue, we turn to Center-Piece Subgraphs.

5.2 Center-Piece Subgraphs

Center-Piece Subgraphs (CePS) [60], like Association Rules Networks, describe the neighborhood of a node or set of nodes, but they differ considerably in how they define this neighborhood. The *Center-Piece Subgraph* $C_p(G, b, Q, k)$ is a subgraph $\mathcal{H}$ of the graph G that contains all *query* nodes in the set Q, contains at most b other nodes, and maximizes an objective function $g(\mathcal{H})$. The parameter k is called a soft_AND coefficient. In simple terms, k is the number of query nodes to which a node must be strongly related in order to be considered a candidate for the subgraph.

In other words, association rules networks define the "neighborhood" of the target product z as the set of set of products that are either direct or indirect causes of z within the ruleset R. A center-piece subgraph, by contrast, defines the neighborhood of the query nodes Q as the set of b products that are most closely related to the products in the set Q according to the objective function $g()$.

The benefit of center-piece subgraphs in the context of market basket analysis is that they allow us to trade scope for granularity. While community detection can find relationships in the product network with virtually no guidance, it requires a reasonable support threshold in order to isolate useful relationships. Similarly, association rules networks require the specification of a ruleset which is, by definition, constrained by a minimum support and confidence. Thus, in both cases, the *number* of products about which one can learn useful information is significantly constrained.

Center-Piece subgraphs provide the opportunity to consider *all* products in an analysis, because the budget parameter b constrains the size of the sets that can be discovered. The cost of this added power is a tremendous decrease in scope. Whereas communities can discover relationships anywhere in the network, and an ARN may extend several levels out from the target product (recall the Bagel ARN of Figure 5.2), a center-piece subgraph is constrained to the set Q of query nodes and at most b other related products. As a result, the set of query nodes Q must be carefully defined in order for the resulting subgraph to be meaningful.

The remainder of the chapter discusses the objective function $g(\mathcal{H})$ maximized by CePS and outlines practical concerns regarding its application to market basket data. We conclude that, for the reasons stated above, center-piece subgraphs are primarily useful for either verification of hypotheses suggested by other techniques or for explaining unexpected results arrived at by other methods. For both of these applications, the set of query nodes Q will be very well-defined.

5.2.1 Objective Function Definition

Define a *Random Walk with Restart* (RWR) [61] on the graph G starting from a node $n \in V(G)$ as follows: At time t, a randomly-walking particle existing at node $n_t \in V(G)$ ($n_0 = n$) randomly transmits itself to one of the neighbors of n_t with a probability proportional to the weight of its edge with n_t. At any time, the particle has a fixed probability c of returning to node n.

From the normalized matrix of edge weights $\mathbf{W}$, one can calculate the probability $p_{i,j}^{(t)}$ that a randomly-walking particle starting at node i stands at j after exactly t steps. The limit as $t \to \infty$ of the $p_{i,j}^{(t)}$ is known as the *steady-state probability* that a particle starting at i will exist at node j. The vector of steady-state probabilities originating from node i, $\mathbf{p}_i$ can be calculated by the equation [61]:

$$\mathbf{p}_i = c\mathbf{W}\mathbf{p}_i + (1-c)\mathbf{e}_i. \tag{5.3}$$

where $\mathbf{e}_i$ is the prior probability of a particle being at node i, i.e. the probability of starting from node i. The matrix $\mathbf{W}$ is *normalized* in the sense that it is a transition matrix: i.e. $\mathbf{W}_{i,j}$ represents the probability that the randomly-walking particle will transition from i to j independent of the possibility of restart.

The RWR problem is very general and has been applied in a number of contexts. For example PageRank [10] now incorporates the notion of restart in its random-walk determination of page relevance to prevent assigning outrageous scores to dense communities of web pages. The CePS problem incorporates RWR into its goodness function as follows:

Define $r(i, j) = p_{i,j}$ to be the steady-state probability that a RWR starting at i exists at j. Further, define $r(Q, j, k)$ to be the steady-state probability that at least k RWRs originating from nodes in the query set Q simultaneously meet at node j. For "hard AND" queries, which are the type of query we will be most interested in, we can define the probability $r(Q, j)$ that random walkers from all query nodes meet at j as:

$$r(Q, j) = \prod_{i \in Q} r(i, j). \tag{5.4}$$

The objective function $g(\mathcal{H})$ for a subgraph $\mathcal{H}$ follows as:

$$g(\mathcal{H}) = \sum_{v \in V(\mathcal{H})} r(Q, v) \tag{5.5}$$

Ref. [60] provides a fast algorithm for extracting subgraphs with high $g(\mathcal{H})$, and our experience shows that it scales to networks with thousands of nodes. In the next section we explore practical concerns regarding the application of CePS to market basket analysis and present results from our data.

5.2.2 Application to Market Basket Analysis

Each technique we have discussed to this point has been limited by the need to specify a minimum support (and possibly minimum confidence) with which to discover relationships. As a result, strong relationships with low levels of support and substitution relationships with artificially low confidence are undiscovered.

Because center-piece subgraphs are constrained in size by the budget parameter b, it is unnecessary to further constrain them with minimum support and confidence

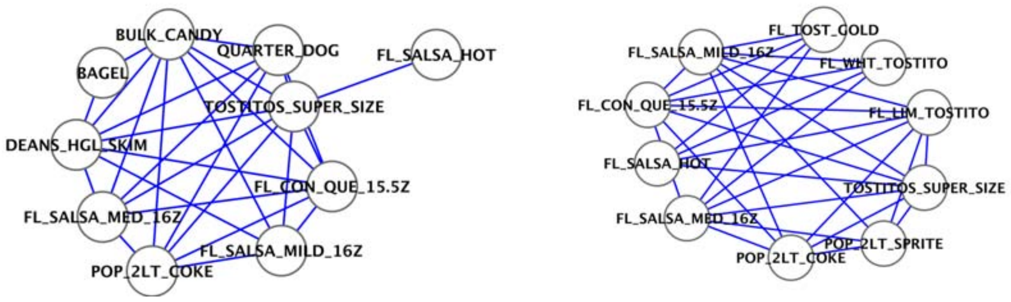


Figure 5.3. Center-Piece Subgraphs with Chips as the query node. Edges are weighed by a) support and b) confidence.

parameters. As a result, they are the only technique we have discussed which is capable of discovering relationships between any and all products that make up the product space. The following sections will show that this property makes center-piece subgraphs invaluable for the exploration of results obtained through other means. Specifically, they are effective for either verifying hypotheses suggested by other techniques or explaining relationships that do not, on the surface make sense.

5.2.3 Center-Piece Subgraphs on Market Basket Data

Figure 5.3(a) shows a center-piece subgraph constructed from the full 2006 product network using a type of tortilla chips (TOSTITOS_SUPER_SIZE) as the query node and a budget $b = 10$. The network contains other chips and salsa, as our prior experience would lead us to expect, but also contains some items (Bulk Candy and Bagel) that are marginally related at best. We explored this phenomenon by constructing subgraphs of gradually increasing size in order to determine which items the algorithm considered more "important" with respect to the tortilla chips. In doing so, we found that Bulk Candy was added as the 6th member of the subgraph, before some of the salsas to which the chips have a stronger connection.

The reason for this is that Bulk Candy, as a popular product, is bought with a tremendously large array of other products (recall the degree distribution of Chapter 4). To see why this causes problems for CePS, imagine a seldom-sold product p_j, appearing in 5 transactions, with which Bulk Candy is bought once. By standard normalization, the transition probability *from* p_j *to* Bulk Candy is at least 1/5, meaning that any random particle that reaches j is highly likely to reach Bulk Candy. Combining this effect over hundreds of less popular products results in a very substantial steady-state probability for popular products.

To reduce the influence of such products, we weighted the edges by confidence instead of by absolute support. That is, the edge A — B is weighted with $\min(P(A|B), P(B|A))$. There are two distinct advantages to using confidence in this instance. First, it forces all edge weights onto a uniform scale between zero and one. Second, it lessens the impact of coincidental purchases with popular products. In the example of the previous paragraph, the weight of the edge between p_j and Bulk Candy is now $\approx \frac{1}{60{,}000}$ and after normalization it is likely that the transition probability from p_j to Bulk Candy is much lower.

Figure 5.3(b) shows the impact of weighting edges by confidence. Now, instead of extraneous products like Bagel and Bulk Candy, we see sodas and other types of chips, which much more closely matches our intuition and corroborates the results found with other techniques.

Figure 5.4 shows a center-piece subgraph with Eggs (EGGS_CSPRING_8CT) as the lone query node and a budget of 10. When we examined the community of eggs and cake mix in Chapter 4 we concluded that when customers bought eggs in our store, they bought them for baking. The subgraph in Figure 5.4 further corroborates this notion: it includes four additional products (brownie mix, butter, margarine, and chocolate chips) and all of them are baking products.

To this point, we have used CePS simply to explore the neighborhood of individ-

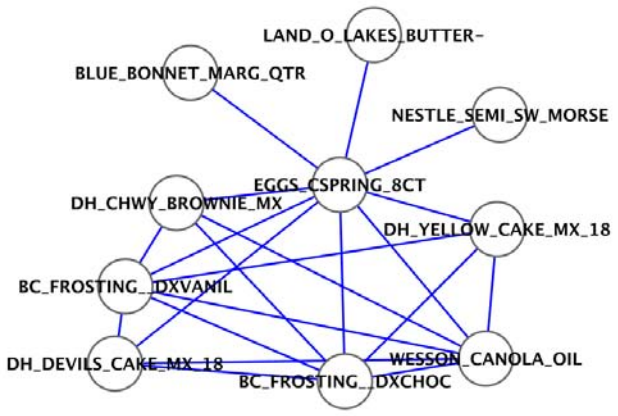


Figure 5.4. A center-piece subgraph with eggs as the query node.

ual items, similar to the way in which we might apply Association Rules Networks. As we mentioned before, however, the CePS framework is actually much more general, and can handle any number of query nodes. The following discussion explores the ability of CePS to explain a single association rule.

Figure 5.5 shows a ten-node center-piece subgraph for one of the less intuitive (and more interesting) rules in the dataset: `Diet Coke, York Mint Patties` → `Chicago Tribune`. Specifically, it is a center-piece subgraph with those three items as query nodes and a budget of 10. The three items in question seem to be entirely unrelated, and yet the rule is ranked highly by a number of interestingness measures. Ideally, the center-piece subgraph would illuminate the relationship between the products and explain the association rule.

Looking at the network, we see something interesting. In addition to patties and Kit Kat, which appear in the Association Rules Network of Figure 5.2, we also see three more types of candy: Hershey's, Mounds and Chuckles. This observation implies that there is some sort of relationship between Chicago Tribune, Diet Coke, and candy. As it turns out, the newspapers in our store are located at the front of

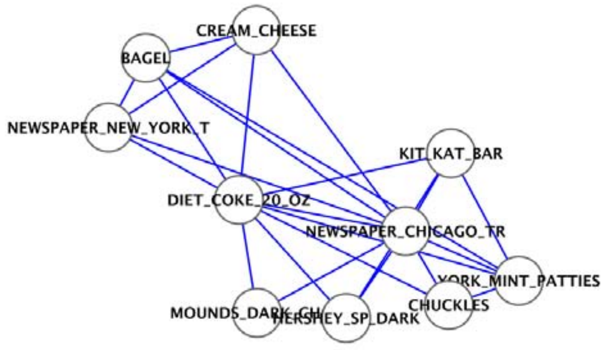


Figure 5.5. A center-piece subgraph with Diet Coke, Newspaper, and Peppermint Patties as query nodes, to explain the association rule.

the store, next to the rack where those candies are sold.

Figure 5.5 shows that, because center-piece subgraphs can consider the entire product network without requiring excessive computation time or providing overwhelming output, they are very effective for *exploration* or *validation* of relationships provided by other methods. As such, they complement nicely the other techniques outlined in this chapter.

Center-Piece Subgraphs require substantially more parameters than any of the other techniques we have discussed. All of our experiments were conducted on small networks ($b \approx 10$), with “hard AND”, meaning that k is equal to the number of query nodes. Though we did not conduct any detailed studies of the parameter selection process, informally we found that the choice of k and b makes little difference in the quality of the subgraph discovered. By choosing b to be large, we observed that popular products such as Bulk Candy and Bagel came to be included in the subgraph. Altering k had no discernible effect for the types of queries we tried.

5.3 A Strategy for Market Basket Analysis

The research we present here has allowed us to make and corroborate a number of significant observations about market basket analysis of real-world data. We re-state the chief observations here, citing the work of others where appropriate.

1. Deriving interesting, actionable knowledge from association rules is difficult because rulesets are often muddied by a preponderance of obvious or redundant rules [41].

2. One can choose to mine *maximal* or *closed* itemsets instead, but these techniques fail to prune away many redundant rules.

3. Similarly, one may choose to rank rules by an *interestingness* measure, but there are many such measures to choose from and they may rank rules inconsistently [59]. As such, it may be difficult to choose an appropriate measure in the absence of prior knowledge.

4. Detecting *communities* of products within the network formed by customer purchases can alleviate redundancy by discovering larger, more expressive relationships among groups of products. However, community detection is less effective within the dense *core* of the network and requires a minimum support threshold, which imparts parameter sensitivity.

5. *Association Rules Networks* are more effective at exploring the core of the network, provided that the chosen target product appears in a large number of association rules. Under other circumstances, they are highly sensitive to the choice of target product and certain networks, even for very popular products, are small and uninformative.

6. *Center-Piece Subgraphs* are useful for explaining or validating relationships discovered by other methods because they do not require a support or confidence threshold to be effective. They are less useful for general analysis because they are necessarily limited in scope.

This list of observations naturally suggests a unified strategy for the analysis of unseen market basket data. First, select a minimum support threshold. On the basis of this threshold, construct a product network and discover communities. The structure of the *interesting* communities in the network (as defined by Equation 4.6) provides a quick overview of any especially strong relationships within the data. The discovered relationships are generally more complex and expressive than those discovered with association rules.

Next, the analyst should decide on a minimum confidence threshold and discover association rules. Choosing a popular product, such as the product that appears in the most rules, as the distinguished item, construct an Association Rules Network. This network will provide a roadmap of some of the important relationships within the core of the network and may illuminate some associations that were not clear in the list of communities.

The set of communities and the Association Rules Network, along with the actual list of association rules if desired, will provide a degree of insight into customer behavior in the store. As a final step, one can apply Center-Piece Subgraphs to analyze carefully selected subsections of the entire (unpruned) network. These subgraphs can serve to corroborate or debunk hypotheses about customer behavior or explain unexpected results in the data. Our experiments have suggested that Center-Piece Subgraphs are most effective if the edges of the network are weighted by confidence rather than support.

5.4 Choosing the Minimum Support Parameter

Since the first step in our proposed procedure requires the user to choose a minimum support parameter, we attempt to provide some guidance into this choice. We are aware of no prior work from which to draw, but one can imagine several reasonable options. For example, one might select an arbitrarily high threshold and iteratively reduce it until the number of rules becomes unmanageable. Alternatively, one may attempt to find a certain number (some hundreds or thousands) of rules, or a certain number of rules that score highly based on his or her favorite interestingness measure.

All of these are valid choices and to evaluate them critically is beyond the scope of this thesis. However, if community detection is the target then existing community

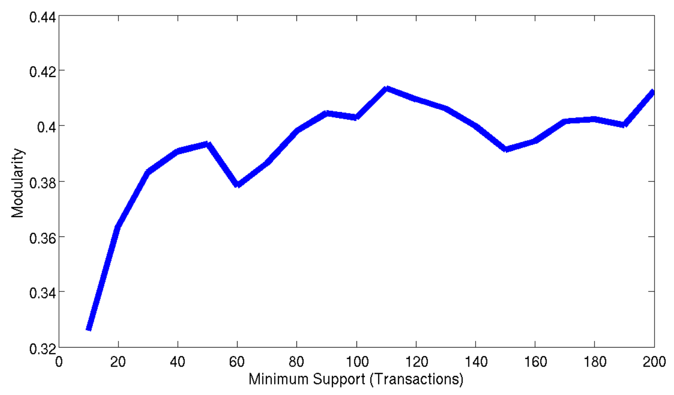


Figure 5.6. Modularity of discovered communities as a function of minimum support.

detection research affords us another option. In Chapter 4, we briefly alluded to the fact that community detection algorithms find poor communities at low levels of minimum support. This fact can be used, in principle, to choose a minimum support threshold.

Modularity (Equation 4.3) provides us with a measure of the quality of a community structure. It follows, then, that discovering communities at a given support threshold with a modularity-maximization algorithm (e.g. [8, 46, 47]) will provide an estimate of the quality of the communities available at that threshold. This suggests the following procedure:

1. Beginning with a very low support threshold (possibly one transaction), discover communities using a modularity-maximization algorithm.
2. Iteratively increase the threshold until the modularity of the discovered community structure begins to plateau or decrease.
3. If there are several thresholds with very similar modularities, it is probably best to pick the lowest one, because it preserves information about the greatest number of products.

Figure 5.6 shows the modularity of the communities discovered by our implementation of Newman's eigenvector modularity algorithm [47] as a function of the minimum support threshold. We chose this algorithm in particular because it is one of the more effective at finding high-modularity decompositions. The graph shows a local maximum at a minimum support of 50 transactions (0.008%) and a global maximum at 110 (0.017%). This suggests that a minimum support threshold of 50 transactions may have been superior to our fairly arbitrary choice of 0.01%. Further evaluation of this method of support tuning will make interesting future work.

CHAPTER 6

PRIVACY-PRESERVING SOLUTION

Having developed an initial framework for mining market basket data as a network, we turn our attention to a related question: How can we *share* this network information securely? The sharing of network data among stores has several potential benefits. First, it should reduce sample selection bias. The data collected by any one store may be inherently biased due to either limitations on the method of collection method or by customer choice. If only data from loyalty card holders is collected, for example, you miss out on people who forget the card, prefer not to use one, or leave it at home.

Second, the ability to produce aggregate network data may improve the availability of market basket datasets for scientific study. A network of products contains significantly less information about a store than does a transaction database, and aggregating the information of multiple stores should further reduce the sensitivity of the data. If "real-world" product networks are widely available for scientific study, then both retailers and the wider research community will benefit.

We present a solution to the above problem, which we call the Private Product Correlation (PPC) protocol. PPC allows a number of participating stores to share their network information securely, such that no malicious adversary can learn anything about a single store's data.

Section 6.1 outlines related prior work in the field of privacy-preserving data mining and draws distinctions with our own work. Section 6.2 provides background on the various cryptographic techniques used. Section 6.3 formally states the protocol and analyzes its complexity. Section 6.4 proves security in the malicious model, Section 6.5 addresses limitations of the basic protocol and provides solutions for overcoming them, Section 6.6 suggests implementation techniques to improve performance, and Section 6.7 presents a timing analysis on real-world data.

6.1 Related Work

There is a large number of publications on privacy-preserving data mining protocols, with privacy-preserving solutions for association rules on horizontally partitioned data being the closest to the problem we study. Among publications on this topic, [38] gave the first protocol followed by a large body of other works [3, 27, 29, 56, 75, 52, 58, 65, 37, 66, 54, 62, 74, 77]. Our work differs from the closest of these publications (that use secure multi-party techniques in constructing the solution) in two important respects: First, we are addressing a different problem, which gives more flexibility to the participants. In particular, we permit the participants to decide the items about which they would like to learn instead of supplying all parties with a set of frequent items computed above a rigid threshold.

Second, we provide stronger security guarantees than in prior published works. In particular, we do not place assumptions that the participants will faithfully follow the prescribed computation required of the the semi-honest model, which prior publications assume. The justification for accepting a semi-honest model is that each participant needs to behave honestly in order for the output to be meaningful. That is to say, a store that fails to follow the protocol will be hurting itself as much as it will be hurting its competitors. However, to demonstrate that the semi-honest model is trivially insufficient, we note that normally in such multi-party protocols only one participant obtains the final answer and is expected to forward it to all other participants. If this designated party simply forwards a wrong value to all other participants, such behavior will not be detected and cannot be defended against in previous publications. In our protocol, on the other hand, relevant information is simultaneously disclosed to the participants who are supposed to learn the result using threshold decryption, which eliminates this major difficulty.

Further, it cannot be assumed that any solution in the semi-honest model can automatically be compiled into a solution secure in the malicious model. Having a protocol secure in the stronger model, however, gives the ability to use it as a protocol in the semi-honest model. That is, normally first a solution in the semi-honest model is developed and then it is enhanced with additional techniques to prevent unauthorized behavior. Then if the only malicious behavior we want to prevent in our protocol is announcement of an incorrect result by a party who recovers the output, this will be done by using threshold encryption, but other expensive zero-knowledge techniques can be left out.

6.2 Preliminaries of the Private Product Correlation (PPC) Protocol

6.2.1 Homomorphic encryption

Our solution utilizes semantically secure homomorphic encryption that allows computation on encrypted data without knowledge of the corresponding plaintext. In particular, we use public-key additively homomorphic encryption such as Paillier [49]. Suppose there is a public-private key pair (pk, sk); we denote encryption of message m as $E_{pk}(m)$ and decryption of ciphertext c as $D_{sk}(c)$. The additive property gives us $D_{sk}(E_{pk}(m_1) \cdot E_{pk}(m_2)) = m_1 + m_2$ and $D_{sk}(E_{pk}(m)^c) = m \cdot c$. It is also possible, given a ciphertext $c = E_{pk}(m)$, to compute a re-encryption of m such that it is not feasible to tell whether the two ciphertexts correspond to the same message or not; this is done by multiplying the ciphertext by an encryption of zero.

Our protocols use a threshold version of homomorphic encryption. Threshold Paillier encryption was developed by [19, 22, 25]. In an (n, k)-threshold encryption scheme, the decryption key is distributed among n parties and the participation of k of them ($k \leq n$) is required to decrypt a ciphertext.

6.2.2 Zero-knowledge proofs

Our protocols rely on zero-knowledge proofs of knowledge from prior literature. In particular, we use:

Proof of plaintext multiplication: Given ciphertexts $c_1 = E_{pk}(a)$, $c_2 = E_{pk}(b)$, and $c_3 = E_{pk}(c)$, the prover proves that c corresponds to multiplication of a and b, i.e., $c = a \cdot b$. Cramer et al. [19] give a zero-knowledge protocol for this proof using Paillier homomorphic encryption, which is the type of encryption used in this work.

6.2.3 Privacy-preserving set operations

Prior literature [26, 39, 28] contains results that permit privacy-preserving operations on sets (or multi-sets). A set $S = \{s_1, s_2, \ldots, s_\ell\}$ is represented as the polynomial $f_S(x) = (x - s_1)(x - s_2) \cdots (x - s_\ell)$. This representation has the property that $f_S(s) = 0$ if and only if $s \in S$.

Privacy-preserving operations on sets use encrypted representations of sets. Given a polynomial $f(x) = a_\ell x^\ell + a_{\ell-1} x^{\ell-1} + \ldots + a_1 x + a_0$, its encryption is formed as encryption of each coefficient a_i: $E(f) = (E_{pk}(a_\ell), \ldots, E_{pk}(a_0))$. This representation can be used to perform many operations on sets in privacy-preserving manner. One such an operation used in our solution is *polynomial evaluation*, which given $E(f)$, y, and public parameters allows one to compute $E(f(y))$. This is done by computing the product $\prod_{i=0}^{\ell} E_{pk}(a_i)^{y^i}$. We also utilize the *set union* protocol of [28], which is the fastest protocol for computing the union of two sets.

6.3 Private Product Correlation Protocol

In our solution we assume that there are n participants (i.e., stores) $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Each participant $\mathcal{P}_i$ sells a set of products we call L_i, where we further define $\ell_i = |L_i|$. We assume that a unique naming convention is used, and different participants will use the same name for a particular product.

6.3.1 Overview of the solution

A natural solution to the product correlation problem in a non-private setting proceeds as follows: each participant counts the number of instances in which two products were sold together at its store, across all pairs of products the participant offers. Given these counts from all participants, the aggregate counts are computed for each pair of products the participants collectively carry. Each participant then saves the aggregate counts corresponding to the products it is interested in.

The same logic could be used in constructing a privacy-preserving protocol for product correlation: each participant computes the counts privately, all of them then engage in a variant of set union protocol preserving (and summing) the counts during the protocol, and finally each participant performs a set intersection on the result to recover the counts for the products of interest.

Existing techniques, however, do not allow this functionality to be implemented in the above form. While privacy-preserving protocols for both set union and set intersection exist, they cannot be combined without leaking intermediate results. Furthermore, the way sets are represented in these protocols does not permit additional information (such as a count) to be stored with an element of the set. These limitations led us to design alternative mechanisms for achieving the above task. In our protocol, every participant initially commits to the set of products about which it would like to learn, without revealing this set to others. The participants agree on the set of products they are going to use in the clear (a naming convention for each of the products they sell). Each participant then computes its counts for all pairs of products and broadcasts encrypted counts to others. The participants jointly add the counts. Each participant is then able to recover (with the help of others) information about the products listed in the commitment at the beginning of the protocol (without others learning anything about the products or their counts).

6.3.2 Protocol description

The participants agree on a (n, n)-threshold homomorphic encryption scheme and generate a public-private key pair for it. Let $E_{pk}(\cdot)$ denote such encryption with the public key.

PPC Protocol:

1. Each participant commits to the list of products about which it would like to learn. Let $D_i = \{d_1, \ldots, d_{m_i}\}$ represent such a set for $\mathcal{P}_i$[1]. $\mathcal{P}_i$ commits to this set by committing to the polynomial $Q_i = (x - d_1) \cdots (x - d_{m_i}) = q_{m_i} x^{m_i} + q_{m_i - 1} x^{m_i - 1} + \cdots + q_1 x + q_0$ for constants $q_{m_i}, \ldots, q_0$. More specifically, at the beginning of the protocol $\mathcal{P}_i$ posts $E_{pk}(q_{m_i}), \ldots, E_{pk}(q_0)$ along with a zero-knowledge proof that q_{m_i} is non-zero as described in sub-protocol NZProof below.

2. Each participant $\mathcal{P}_i$ prepares a list of its products L_i. The participants engage in a privacy-preserving set union protocol to determine the set of products all of them sell. Let $L = \{p_1, \ldots, p_\ell\}$ denote the outcome of the protocol.

3. For each pair of products $p_j, p_k \in L$, $\mathcal{P}_i$ computes $E_{pk}(c^i_{jk})$, where c^i_{jk} is its count for the number of times products p_j and p_k were sold together for $\mathcal{P}_i$. Note that if at least one of p_j and p_k is not in L_i, c^i_{jk} will be 0. $\mathcal{P}_i$ broadcasts the values $E_{pk}(c^i_{jk})$ for each $0 \le j, k \le \ell$ $(j \ne k)$.

4. Each $\mathcal{P}_i$ locally computes the encryption of the sum of all counts for each product pair p_j, p_k as $E_{pk}(c_{jk}) = \prod_{i=1}^{n} E_{pk}(c^i_{jk}))$. $\mathcal{P}_i$ then rearranges the values to form tuples $(p_j, E_{pk}(c_{j1}), \ldots, E_{pk}(c_{j\ell}))$ for each $p_j \in L$.

[1]This set can be inflated with fake items to hide the actual size of the set of items in which the participant is interested.

5. Now each $\mathcal{P}_i$ obtains the decryption of counts of the products to which $\mathcal{P}_i$ committed in step 1 (i.e., all counts c_{jk} such that $p_j \in D_i$, $1 \leq k \leq \ell$, and $k \neq j$). To accomplish this, we perform the following steps in parallel for each $\mathcal{P}_i$:
   (a) One party posts $E_{pk}(Q_i(p_1))$, $E(Q_i(p_2))$, $\ldots$, $E_{pk}(Q_i(p_\ell))$. Note that $E_{pk}(Q_i(p_j)) = E_{pk}(q_{m_i})^{p_j^{m_i}} \cdot E_{pk}(q_{m_1-1})^{p_j^{m_i-1}} \cdots E_{pk}(q_0)$, and since this is a deterministic process, everyone can verify the result of the computation. Also, note that $Q_i(p_j)$ will be 0 iff p_j was in D_i.
   (b) For each value $E_{pk}(Q_i(p_j))$, $j = 1, \ldots, \ell$, the participants randomize the underlying plaintext by engaging in a NZRM (non-zero random multiplication) protocol described below (each participant executes the NZRM protocol in order). In this protocol each participant multiplies each plaintext by a random non-zero value and proves correctness of the computation. We denote the result of the computation by $E_{pk}(b_j^i)$. Note that b_j^i is 0 if $p_j \in D_i$ and a random value otherwise.
   (c) For each $0 \leq j, k \leq \ell$ ($j \neq k$), one party computes the values $E_{pk}(b_j^i) \cdot E_{pk}(c_{jk}) = E_{pk}(b_j^i + c_{jk})$. Note that the encrypted value is c_{jk} if $p_j \in D_i$ and is a random value otherwise.
   (d) The parties engage in a joint decryption protocol to reveal the values $b_j^i + c_{jk}$ to the $\mathcal{P}_i$ for all $0 \leq j, k \leq \ell$.

6.3.3 Sub-protocols

NZProof Protocol: A user has a ciphertext c and would like to prove that the corresponding plaintext a (where $c = E_{pk}(a)$) is non-zero.

1. The prover chooses a random value b and posts $E_{pk}(b)$ and $E_{pk}(ab)$.
2. The prover proves in zero knowledge that the decryption of $E_{pk}(ab)$ corresponds to the multiplication of the decryptions of $E_{pk}(a)$ and $E_{pk}(b)$.
3. The prover decrypts $E_{pk}(ab)$ and posts ab (this value is jointly decrypted in case of threshold encryption). If ab is non-zero, a is also nonzero.

NZRM Protocol: Given a ciphertext $c = E_{pk}(a)$, a participant multiplies the underlying plaintext by a random non-zero value b, outputs $c' = E_{pk}(ab)$ and proves correctness of the computation.

1. The participant chooses a random value b and posts $E_{pk}(b)$ and $c' = E_{pk}(ab)$.
2. The participant proves in zero knowledge that the decryption of $E_{pk}(ab)$ corresponds to the multiplication of the decryptions of $E_{pk}(a)$ and $E_{pk}(b)$.
3. The participant also proves that $E_{pk}(b)$ encrypts a non-zero value using the NZProof protocol.

6.3.4 Complexity Analysis

To simplify the analysis, let m be the upper bound on the number of items to which a participant commits (i.e., $m = \max_i\{m_i\}$). Then the total work and communication for each participant $\mathcal{P}_i$ in step 1 of the protocol is $O(m+n)$, which amounts to $O(mn+n^2)$ communication across all parties. In step 2, the overhead associated with the semi-honest (malicious) version of the set union protocol is bounded by $O(n\ell)$ ($O(n\ell^2+n^2\ell)$, resp.) computation and communication per person and therefore $O(n^2\ell)$ ($O(n^2\ell^2+n^3\ell)$, resp.) overall communication. In step 3, each participant's work and communication is $O(\ell^2)$ resulting in $O(n\ell^2)$ overall communication. Step 4 involves $O(\ell^2)$ cheaper operations (modular multiplications) per participant and no communication.

To determine the output for a single participant $\mathcal{P}_i$ in step 5, the overhead is as follows: step 5.a requires $O(m\ell)$ operations and the same amount of overall communication. Step 5.b requires $O(\ell)$ work and communication per participant, resulting in the total of $O(n\ell)$ communication. Step 5.c involves $O(\ell^2)$ computation and overall communication. Finally, step 5.d involves $O(\ell^2)$ threshold decryptions, which amounts to $O(\ell^2)$ work and communication per participant resulting in $O(n\ell^2)$ total communication. Since this part is executed for each participant, the total communication of step 5 over all participants is $O(n^2\ell^2)$.

Thus, if the protocol is built to resist malicious adversaries, the work per participant is $O(n\ell^2+n^2\ell)$ and the overall communication is $O(n^2\ell^2+n^3\ell)$.

6.4 Security Analysis

In this paper we argue security using the standard simulation argument. To show security of a protocol in secure multiparty computation, an ideal security model is assumed. In the ideal model, participants privately submit their inputs to a fullytrusted third party (TTP) that performs the desired computation and forwards the output to respective recipients. Then to argue security of our protocol, we need show that for any adversary in our protocol, there exists a polynomial-time adversary in the ideal model that achieves the "same" effect as the real adversary, i.e., the real adversary cannot learn more information about the participants' data than in the ideal model. Thus, the protocol is as secure as the ideal protocol. The adversary in this framework assumed to be taking part in the protocol and possibly has the ability to control several users. We allow the adversary to arbitrarily deviate from the protocol, i.e., assume an active adversary. As our protocol makes the total set of products known to the participants at an intermediate stage, we begin by describing the ideal model that our protocol is equivalent to:

1. Each participant, $\mathcal{P}_i$, reveals its D_i and its L_i to the TTP.
2. The TTP reveals L (the union of the L_i sets) to all participants along with the size of D_i and L_i (for all i) to all participants.
3. $\mathcal{P}_i$ submits c^i_{jk} for each pair of items p_j and p_k to the TTP.
4. The TTP reveals to $\mathcal{P}_i$ c_{jk} for all $p_j \in D_i$.

Since our protocol makes the list of products known, it potentially can allow dishonest participants to change their count values based on the result of the set union they obtain. It also reveals an upper bound on the size of the D_i and L_i sets (as mentioned earlier, sets can be padded with fake values). These values/capabilities do not appear to be useful to an adversary.

We first argue that a malicious adversary's actions in the protocol can be mimicked by a malicious adversary in the TTP model, assuming that the TTP adversary can terminate the TTP at any time. If the adversary stops the protocol at any time, then this is the same as stopping the TTP at the corresponding stage, and so in what follows we assume that the adversary does not terminate the protocol.

In Step 1, the real adversary submits an encrypted polynomial. Since the leading coefficient is non-zero, we know that this is not the zero polynomial. Therefore, this polynomial will be zero in at most m_i locations and so this represents a set with at most m_i items, which is exactly what happens in the ideal model (i.e., the adversary in the ideal model has to do). In Step 2, the adversary must submit a set of items to the set union protocol. Assuming the protocol uses a maliciously secure set union protocol, this is exactly what the adversary submits to the TTP in the ideal model. After receiving the results from Step 2, the real adversary submits a list of count values, but whatever values he submits can also be submitted to the TTP in the ideal model. Steps 4, 5a, and 5c of the protocol are deterministic, so the adversary's actions cannot deviate from the prescribed protocol. Assuming the existence of a secure protocol for NZRM in the malicious model, the adversary cannot deviate from this protocol in Step 5b. Similarly, assuming the existence of a secure decryption protocol, the adversary cannot deviate in Step 5d.

What is left to be shown is that the adversary does not learn additional information from the protocol. We do this by utilizing the composition theorem of [13], according to which, if parts of a protocol (sub-protocols) can be shown to be secure, then they can be combined to achieve overall secure protocol. we replace the protocols for set union, NZRM, and decryption by calls to TTP protocols. That is, if the protocol is secure when replacing the protocols with these TTP calls, then the protocol that uses secure versions of these protocols will also be secure. To do this we show that everything the adversary sees in the protocol can be simulated by a polynomial-time algorithm that knew only the adversary's inputs and outputs. We analyze each step of the protocol in more detail:

- Step 1: When given the size of each party's set, the simulator can create the correct number of encryptions and a zero-knowledge proof of a non-zero value. Since the encryption scheme is semantically-secure, encryptions of random values are indistinguishable from the encryptions in the real protocol.

- Step 2: This has been replaced with a call to a TTP, furthermore the output of the TTP is a proper subset of the adversary's output.
- Step 3: Again the values are simply encryptions with a semantically-secure encryption scheme, which are straightforward to simulate in an indistinguishable manner.
- Step 4, 5a, and 5c: There is no new information in these steps.
- Step 5b: Assuming the NZRM is done with a TTP, the adversary sees only a new list of semantically-secure encryptions.
- Step 5d: In this step the adversary learns the values. For the values corresponding to adversary outputs the simulator reveals the outputs. For the values that are not revealed to the adversary (i.e., items not in D_i) the simulator just uses a randomly chosen value. This is indistinguishable from the actual value revealed in the protocol because $Q_i(p_j) \cdot R$ is a random value (where R is a random value unknown to the adversary – this corresponds to the value from the NZRM protocol).

To summarize, assuming the protocol uses secure building blocks, it is secure in the malicious model.

6.5 Security Caveats

While the protocol described above is secure in the malicious model, a standard strong security model in secure multiparty computation, it is not without vulnerabilities. The malicious security model guarantees that participants will be forced to perform the prescribed protocol steps correctly and are unable to learn any information besides their output during protocol execution. It does not, however, address the ability of adversaries to cleverly engineer their input such that they learn extra information from the output they receive or deceive other participants into accepting incorrect results. This section discusses these vulnerabilities and, where possible, addresses them.

6.5.1 Number of Participants

In the protocol described above, privacy of individual inputs rests on the fact that such inputs are hidden among data from other participants and no one can determine the inputs exactly. This means that when the number of participants is small, weaker privacy guarantees can be provided. In particular, in the case of two participants, inputs of the other party can be determined exactly from the inputs and outputs alone. The benefits of the protocol are, however, the most pronounced with a larger number of participants, who individually cannot make accurate predictions. Thus, combining data from only a couple of small stores will likely still not result in achieving high precision of data that large retailers such as WalMart have. The participants, however, must be aware of the possibility of leaking partial information when the protocol is executed with a small number of participants and can opt out from participation. This problem is not unique to our approach, and several related protocols suffer from it as well [38, 63].

If a small number of stores still would like to engage in collaborative computation, the function being computed on their transaction data must be modified to reduce information leakage. From prior literature, Verykios et al. [64], for example, present algorithms for hiding a specific set of association rules in a list of transactions. A modified version of this approach could be applied to our protocol to provide a partial solution. As the case of two participants is not central to this work, detailed treatment of this case is out of scope of this discussion.

6.5.2 Collusion

The protocol described above provides the highest possible guarantee in presence of collusion[2]. That is, in order to learn private information of participant $\mathcal{P}_i$, the remaining $n-1$ parties must collude. Specifically, the colluding parties can carry out the protocol a second time without P_i. (More generally, any one participant can

[2]Theoretically, if the function being cooperatively computed is one-way, the inputs of each participant are highly non-predictable, and the input domain is too large for exhaustive search, $n-1$ colluding parties will have difficulties recovering the inputs of the remaining party. Such assumptions are, however, unrealistic for functions computed in practice in the privacy-preserving setting.

learn private inputs of another participant by engaging in the protocol with $n - 2$ other participants in groups of any size.) The participants thus must be aware that this possibility is unavoidable, and if any single participant believes that there is a chance that the remaining participants might conspire against it, it should refuse to participate.

6.5.3 Lying Adversaries

Finally, and perhaps most troubling, a single dishonest participant can deceive all other participants while retaining the correct output for itself. The procedure for this is as follows: without loss of generality assume that instead of submitting the proper count c^i_{jk}, $\mathcal{P}_i$ submits $c^i_{jk} + 100$. In this case, at the end of the protocol, all participants will receive the erroneous count $c_{jk} + 100$. $\mathcal{P}_i$ is aware of its lie, and can correct the error. All other participants, meanwhile, are unaware of the deception and receive incorrect output.

While this is a significant limitation, no privacy-preserving computation can guarantee the legitimacy of its inputs, only that the computation on those inputs is correct. Privacy-preserving association rules protocols suffer from a similar flaw in the sense that any participant can make any rule appear supported or unsupported by providing incorrect input. In these protocols, however, a malicious user cannot "undo" its lie simply because the protocols do not reveal the global support and confidence of rules, but merely whether they meet the minimum criteria.

In the protocol of [38], for example, the end result of the protocol is the so-called *excess support* ES_I of an itemset I, defined as:

$$ES_I = \sum_i X.sup_i - s * |DB_i| \tag{6.1}$$

where $X.sup_i$ is participant i's support for the rule, s is the global support threshold, and $|DB_i|$ is the number of transactions in participant i's transaction database. The excess support is the number of transactions by which the aggregate global support exceeds the support threshold. If this count were revealed, an unscrupulous participant $\mathcal{P}$ can exploit the protocol in the following manner: For a given rule with support count $X.sup_i$, instead contribute $X.sup_i + 100$ to the sum. Upon learning the global excess support, $\mathcal{P}$ would be able to subtract his lie and determine if, and to what extent the rule was supported.

If the participants so desire, our protocol can be made more resilient to such misbehavior by adapting it to this same framework. The participants agree on a minimum support threshold s and wish to determine, for each pair of products (p_j, p_k), whether the count c_{jk} exceeds the minimum support threshold. We now outline a procedure for this computation. Let ES_{jk} denote the global excess support of the relationship between p_j and p_k, ES^i_{jk} denote the local excess support for participant $\mathcal{P}_i$, and N is the public Paillier modulus.

1. The participants calculate the encryption of the global excess support $E_{pk}(ES_{jk})$ from the encrypted local excess supports $E_{pk}(ES^i_{jk})$ using the homomorphic property of Paillier encryption: $E_{pk}(ES_{jk}) = \prod_i E_{pk}(ES^i_{jk})$.
2. The first $n-1$ participants randomize the underlying plaintext using the following procedure:
   (a) Participant $\mathcal{P}_1$ chooses a random number $r_1 \in \mathbb{Z}_N$ and computes $C_1 = E_{pk}(ES_{jk} + r_1) = E_{pk}(ES_{jk}) * E_{pk}(r_1)$ and makes the result available to all participants.
   (b) Each participant $\mathcal{P}_i$ in $(\mathcal{P}_2, \ldots, \mathcal{P}_{n-1})$ chooses a random $r_i \in \mathbb{Z}_N$ and computes $C_i = E_{pk}(ES_{jk} + \sum_{t=1}^{i} r_t) = C_{i-1} * E_{pk}(r_i)$.
3. The participants jointly decrypt the result and reveal $R = (ES_{jk} + \sum_{i=1}^{n-1} r_i)$ to P_n.
4. The participants securely evaluate a circuit which accepts n values $(r_1, \ldots, r_{n-1}, R)$. The circuit outputs 1 if $(R - \sum r_i) \mod N < \frac{N}{2}$ and 0 otherwise.
5. If the circuit outputs 1, the participants know the association is supported. Otherwise, it is not.

The protocol works by computing the securely computing global excess support as the sum of local excess support values. The given association is supported if the result of the computation is "positive" (i.e. $< \frac{N}{2}$) and not supported otherwise. Techniques for secure boolean circuit evaluation are well-known, and software for performing this computation recently became available. [6] It is worth noting that the above procedure is secure only in the semi-honest model, since participants could change their value of r_i between steps 2 and 4 above. The protocol can be made resilient to malicious behavior by utilizing additional techniques such as zero-knowledge proofs of knowledge for Paillier encryption scheme (proof of knowledge of plaintext and proof that two plaintexts are equal) [22, 5], committed oblivious transfer and multi-party computation on committed inputs (see, e.g., [20, 21, 30, 36] and others). We leave the protocol details as a direction for future work.

6.6 Efficiency Improvements

6.6.1 Polynomial multiplication and evaluation

As described in [26], polynomial evaluation can be performed more efficiently by applying Horner's rule. Recall from section 6.2.3 that, given $E(f) = (E_{pk}(a_\ell), \ldots, E_{pk}(a_0))$ and y, computing $E(f(y))$ amounts to calculating $\prod_{i=0}^{\ell} E_{pk}(a_i)^{y^i}$. More efficient computation can be performed by evaluating it from "the inside out" as $E(f(y)) = ((\cdots(E_{pk}(a_\ell)^y E_{pk}(a_{\ell-1}))^y \cdots)^y E_{pk}(a_1))^y E_{pk}(a_0)$. Considering that the polynomial is always evaluated on small values (compared to the size of the encryption modulus), this results in significant performance improvement.

The set union protocol we utilize [28] also uses polynomial multiplication, which for large polynomials becomes inefficient. Given an encrypted polynomial $E(f_1) = (E_{pk}(a_{\ell_1}), \ldots, E_{pk}(a_0))$ and another polynomial $f_2(x) = b_{\ell_2}x^{\ell_2} + \cdots + b_0$, the multiplication consists of computing (encrypted) coefficients c_i of their product.

That is, for $i = 0, \ldots, \ell_1 + \ell_2$, $E_{pk}(c_i) = \prod_{j=\max\{0,i-\ell_2\}}^{\min\{i,\ell_1\}} E_{pk}(a_j)^{b_{i-j}}$. This operation can be performed faster by using multi-base exponentiation (see, e.g., [45] for more detail), where instead of computing exponentiations $g_1^{x_1}, \ldots, g_k^{x_k}$ separately, exponentiation is performed simultaneously as $g_1^{x_1} \cdots g_k^{x_k}$ for a fixed (small) value of k. This can speed up computation by several times.

6.6.2 Packing

Assume that the (total) c_{ij} values are no larger than 2^M. It is likely that such a bound can be found, where $M \ll \rho$ and ρ is the number of bits in the modulus of the homomorphic encryption scheme. In Step 3 of the PPC protocol, the number of encryptions that a participant posts is ℓ. We, however, can reduce this number by storing $s = \lfloor \frac{\rho}{M} \rfloor$ values in a single encryption. This means that this new way of encoding counts would require only be $\lceil \frac{\ell}{s} \rceil$ encryptions.

To see how this compression is done, suppose we want to place the following M-bit values $x_1, ..., x_s$ into a single encryption. We could do this by computing $E_{pk}(\sum_{i=1}^{s} 2^{M(i-1)} x_i)$. Note that as long as individual results do not get larger than M bits, we can add to such compressed encryptions (to obtain the value representing the pairwise values) and we can multiply them by constants. In the protocol, addition is the only operation performed on the counts. Such compressed encryptions of counts can also easily be used in Step 5 of the protocol if only the counts that correspond to the same product are combined together (e.g., c_{jk} and $c_{jk'}$ can be included in the same ciphertext for any k, k'). Doing this compression does not reduce the asymptotic communication of the protocol (as this has no effect on the set union), but it does improve the performance of Steps 3–5 in the protocol.

6.7 Performance

One persistent concern with privacy-preserving data mining protocols is that they tend to be very slow in practice. We now present a detailed discussion of the performance of our protocol and the optimizations that we employed to make it tractable for our data. This is, to our knowledge, the first significant study of the real-world performance of any privacy-preserving data mining algorithm.

We implemented the protocol in C using the GNU Multiple Precision (GMP)[3] library for large-precision integers. We use (n, n)-threshold Paillier encryption. and our set union protocol is the protocol of [28], with one optimization. The participants decide on a number, B, of buckets, and each participant divides its products among the B buckets according to a hash function. The participants then run the protocol B times and combine the results of the runs. If the participants split the products uniformly among $B = \frac{\ell}{log(\ell)}$ buckets, the buckets will contain $log(\ell)$ products on average, reducing the time complexity of the set union from $O(n\ell^2)$ to $O(n\frac{\ell}{log(\ell)} * log(\ell)^2) = O(n\ell * log(\ell))$. In general, the participants do not know the number of the products in the union beforehand, so we approximated ℓ with $n * \ell_i$ where ℓ_i is the number of products sold by $\mathcal{P}_i$. We find that this optimization significantly reduces the running time of the set union phase.

Each experiment ran on a dedicated 2.2 GHz AMD Opteron core processor and was guaranteed at least 2 GB of RAM. In conducting the experiments, we tried to account for the parallelism afforded by the protocol. In other words, even though our experiments were run on single machines the numbers that we present are an estimate of the amount of time it would take if each store were doing its own processing in parallel wherever the protocol allows for it. In situations where the computation must be serialized (such as portions of the set union protocol), we time

[3] `http://www.gmplib.org`

the computation of all n participants.

The following list describes our time estimates in greater detail:

1. We do not include key generation, the construction of unencrypted polynomials for the set union protocol, or the construction of the commitment polynomials. These are considered pre-processing steps done outside the context of PPC.

2. Within the set union protocol, we report $\frac{1}{n}$ of the total time for the randomization step (step 2). The reasoning behind this is that, though each participant must randomize each tuple, they need not wait for each other (i.e the first participant could randomize one set while the next participant does a disjoint set, and then they could trade off).

3. Within the set union protocol, we only report the time for one participant to decrypt (step 4). In this case, all decryptions can be done completely in parallel because each participant gets simultaneous access to the full list of items to be decrypted. The time taken to combine the encrypted values is also, necessarily, included.

4. We report the time for one participant to compute the tuples $(p_i, p_j, E(c^i_{jk}))$ in step 3 of the PPC protocol. Again, this is a step that can be done completely in parallel.

5. In step 5d of the PPC protocol, we time only one participant's decryption effort. The combination step is also timed.

Table 6.1 shows timing results for several different choices of the parameters n and ℓ. All of the experiments were done with a 1024-bit key in keeping with modern standards for public-key encryption. It provides performance information as a function of the number of participants n, the size of the set union ℓ, and size of each store. We constructed several stores from our single dataset by assigning products at random to each store from a pre-determined list of the top-selling products. Specifically, if each store was to have ℓ_i products, it chose those products at random from the top $4\ell_i$ products in our data. The size of the set union depends on the overlap between the selections of the different stores and is, therefore, random.

The time taken to complete the protocol scales most significantly with the size of the set union. Increases in the number of participants are partially offset by the in-

TABLE 6.1

TIMING RESULTS FOR PROTOCOL EXECUTION

Participants	Products Per Participant	Products in Union	Time (min:sec)
3	100	239	105:25
3	200	461	386:14
3	400	942	1,637:09
5	100	312	293:33
5	200	601	1,097:21
5	400	1231	4,794:07
10	100	375	939:17
10	200	744	1,883:04
10	400	1502	7,191:15

crease in the amount of available parallel computation afforded by the participation of multiple entities.

6.7.1 Revealing Pairs with Zero Sales

In our data, there are many pairs of products that have never been purchased together. The reasons for this are varied, but the two main causes are either that the products are not available at the same time (such as Halloween and Christmas seasonal products) or that they are simply both unpopular, and as a result have never appeared in combination. In some cases, participants may be willing to leak that products are never sold together, because the information is unlikely to be useful to an adversary. In this case, the users will see several performance benefits:

- When forming tuples $(p_j, p_k, E(c^i_{jk}))$, participants can use the value 1 instead of a proper expensive encryption of zero.
- When computing the final count c_{jk} a participant can ignore the contribution of any "trivial" counts with $E(c^i_{jk}) = 1$

TABLE 6.2

TIMING RESULTS FOR PROTOCOL EXECUTION WHEN PARTICIPANTS DISCLOSE PAIRS WITH ZERO COUNT

Participants	Products Per Participant	Products in Union	Time (min:sec)	Speedup
3	100	239	14:21	7.35
3	200	461	45:40	8.46
3	400	942	98:42	16.69
5	100	312	61:57	4.74
5	200	601	167:49	6.54
5	400	1231	338:49	14.14
10	100	375	336:15	2.79
10	200	744	844:42	2.23
10	400	1502	1506:02	4.77

- If the final count $E(c_{jk}) = 1$, this count does not need to be included in the decryption protocol (step 5).

Table 6.2 shows timing results if we allow for the disclosure of zero-count pairs. We used real data for this analysis, forming individual "stores" out of subsets of our store data. As we mentioned above, each fictitious store carrying ℓ products chose its subset of products randomly from a larger set of the store's best-selling products. This method of construction was chosen to insure that the number of observed zero counts remained reasonable, since allowing unpopular products to be chosen would result in a very significant proportion of zeros.

Improvement as a result of leaking zero counts is substantial, achieving at least 2.23 times speedup even with ten participants. In general, the benefit to leaking zeros increases with the number of products in the union, where zeros are more likely, and decreases with the number of participants, where the overhead of each computation is larger. The only exception is the case with 10 stores and 200 products, in which we observe a decrease in speedup from 2.79 to 2.23. Based on the other results, this seems like an aberration, caused perhaps by unusual activity on the machine on which the experiment was run. In all, we observe that leaking zero counts makes the computation substantially more tractable. Whereas computing all $O(\ell^2)$ counts takes almost five days with ten participants and 1500 products in the union, leaking zero counts reduces computation time to just over one day.

6.8 Incorporating Additional Analysis Techniques

The broader analysis framework of Chapter 5 can be (almost completely) adapted into PPC with very little effort. The only necessary information that is not provided by the protocol discussed above is a method of calculating the confidence of the association $p_i \rightarrow p_j$. This section addresses how to provide for this calculation.

Recall that the count c^i_{jk}, provided as input by participant $\mathcal{P}_i$ represents the number of times products p_j and p_k sell together in $\mathcal{P}_i$'s data, and c_{jk}, calculated by the protocol, is the sum of the c^i_{jk} across all participants.

Suppose we define the count c^i_{0k} as the number of transactions in which product k appears in $\mathcal{P}_i$'s data, and analogously define c_{0k} as the sum of the c^i_{0k} across all participants. If we require that no product be assigned the number zero and that no participant may commit to product zero (i.e. the participants prove in zero knowledge that their commitment polynomial $D_i(x)$ does not have zero as a root) then the confidence of the association $p_j \rightarrow p_k$ can be trivially calculated as $c = \frac{c_{jk}}{c_{0k}}$.

From these calculations, participants may build a network with the edges weighted by confidence, as suggested in Section 5.2 and may build an Association Rules Network at any level of support and confidence that they desire. There are two limitations to this solution: first, minimum support counts for association rules will have to be absolute rather than relative, because the size of the aggregate transaction database is unknown. Second, the ARN can only be constructed from 2-itemsets (that is, only association rules with single-item antecedents can be discovered). Nevertheless, our market basket analysis strategy can be adapted into our privacy-preserving framework with only minimal effort and loss of information.

6.9 Conclusion

This chapter presents the Private Product Correlation (PPC) protocol for the secure exchange of product network information. It differs from relevant prior work (such as privacy-preserving association rule computation) in that it provides more information to participants, (namely, a sum of counts) and is secure against malicious as well as semi-honest adversaries.

Additionally, we present timing results which show that our framework is tractable for participants with sufficient computing power. Our results suggest that execution of the protocol would take on the order of a few days for participants with a modest number of products using widely-available off-the-shelf hardware. Furthermore, if participants are comfortable with leaking pairs of products that they do not sell together at all, they will see significant gains in performance.

We hope that the above-described analysis techniques and privacy-preserving protocol lead to the increased availability of transactional datasets for scientific study. Product networks contain significantly less information than transaction databases, because some information is lost in the aggregation process. The combination of several product networks, then, should reveal almost nothing about one store's marketing model. As such, retailers may consider an aggregated product network devoid of proprietary information, and may release it for study.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This thesis deals primarily with the application of data mining techniques to the solution of the *market basket problem*: the location of meaningful associations in customer purchase data. There is an overwhelming abundance of prior research in the mining of mining market basket data in general, and the use of association rules in particular. The bulk of this research has focused on developing algorithms for mining association rules [2, 12, 11, 72, 73], techniques for visualizing association rules [7, 33, 41, 67], techniques for eliminating redundant rules [41, 31, 70, 71], objective measures of association interestingness [23, 44, 59], or comparing the performance of association rule algorithms on either real or synthetic datasets [35, 76].

Conspicuously absent, however, is a thorough study of exactly *how* market basket data can be effectively mined. That is to say, no prior work answers the question *Given an unseen market basket dataset, what set of steps should I follow to conduct a thorough, complete analysis?* This line of research is hampered by two distinct factors:

- **Lack of Available Data**: Real-world market basket datasets are owned by the retailers who collect them and generally are not available for academic analysis. As a result, most techniques are validated on either generic (non market-basket) data or on synthetic data. It is difficult to evaluate the effectiveness of any framework in such a setting.
- **Lack of Diversity in Analysis Techniques**: Many of the analysis techniques that have been developed, (i.e. closed [70] and maximal [31] itemset

> mining and correlation rules [11]) are very similar to association rules in that they have similar strengths and weaknesses. Since no analysis technique is perfect, an effective framework should employ a diverse set of tools.

Our work contains several novel contributions aimed at answering exactly this question, all of which are validated on real-world data. First, we study the properties of *networks of products* and show that detecting *communities* within these networks can uncover expressive relationships between products that may be difficult to find with association rules. We show that, in addition to being more expressive than association rules (in that relationships can be expressed more compactly) the structural information available in communities can assist with financial decisions such as the location of profitable promotions. Finally, we develop a novel measure of interestingness for communities of products and show that it favors communities which intuitively seem interesting.

Next, we study the application of two existing techniques, Association Rules Networks [15, 16, 51] and Center-Piece Subgraphs [60] to the market basket problem. We find that these algorithms complement community detection in the sense that they can be used effectively to find relationships that communities are unlikely to discover. On the basis of this observation, we propose a very general framework for the mining of unseen market basket data in the absence of background knowledge. The framework employs community detection as an initial exploratory step, using Association Rules Networks to uncover relationships within the dense *core* of the network and Center-Piece Subgraphs to validate hypotheses or explore individual relationships that require more explanation.

Finally, we develop a protocol, called PPC, for securely sharing product network data. Specifically the protocol allows n participating stores to compute the number of transactions (c_{jk}) in which product j and product k are sold together in all n stores combined. The computation is secure in the sense that the participants learn only the aggregate transaction counts c_{jk} without learning the individual counts for any of the other participants. We posit that our PPC protocol is useful not only for combating potential sample selection bias, but also could increase the availability of real-world network data for study, since an aggregate network contains substantially less information than a single transaction dataset.

7.1 Future Work

Though this thesis addresses a diverse array of topics, there remain a number of open questions that merit further exploration. Most importantly, we would like to apply our techniques to other datasets, in order to guarantee that the approach is fully generalizable. Second, the use of modularity-maximization algorithms for community detection in product networks is probably sub-optimal given the differences between product networks and social networks that we discussed in Chapter 4. We would like to develop and validate new community detection methods designed specifically for the market basket problem.

One weakness of our community detection framework is that it requires the specification of a support threshold. There is little existing work regarding the selection of support and confidence thresholds for association analysis, and by formulating the market basket problem on networks we have access to a number of graph-theoretic measures that may be able to aid in the selection process. A principled study in this vein would represent a major contribution to the field.

Finally, we would like to develop a formulation of the PPC protocol that is "fully" malicious-model secure in the sense that it does not offer any advantage to lying adversaries. Such a formulation would not only improve the security of the protocol as it stands but may also lead trivially to the most secure protocol for privacy-preserving association rule mining to date.

BIBLIOGRAPHY

[1] G. Adomavicius and A. Tuzhilin. User profiling in personalization applications through rule discovery and validation. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–381. ACM New York, NY, USA, 1999.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in very large databases. In *Proceedings of the 20th International Conference on VLDB*, pages 487–499, Santiago, Chile, 1994.

[3] M. Ashrafi, D. Taniar, and K. Smith. Reducing communication cost in a privacy preserving distributed association rule mining. In *DASFAA*, pages 381–392, 2004.

[4] Sitaram Asur, Duygu Ucar, and Srinivasan Parthasarathy. An ensemble framework for clustering protein-protein interaction networks. In *ISMB/ECCB*, pages 29–40, 2007.

[5] O. Baudron, P.A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 274–283. ACM New York, NY, USA, 2001.

[6] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM New York, NY, USA, 2008.

[7] J. Blanchard, F. Guillet, and H. Briand. Exploratory visualization for association rule rummaging. In *KDD-03 Workshop on Multimedia Data Mining (MDM-03)*, 2003.

[8] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks, 2008.

[9] T. Brijs, K. Vanhoof, and G. Wets. Defining interestingness for association rules. *International journal of information theories and applications*, 10(4):370–376, 2003.

[10] S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a Web in your Pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998.

[11] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. *Proceedings of the ACM SIGMOD*, pages 265–276, 1997.

[12] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. *ACM SIGMOD Record*, 26(2):255–264, 1997.

[13] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[14] N.V. Chawla and G. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23:331–366, 2005.

[15] S. Chawla, B. Arunasalam, and J. Davis. Mining open source software (oss) data using association rules network. *PAKDD*, pages 461–466, 2003.

[16] S. Chawla, J. Davis, and G. Pandey. On Local Pruning of Association Rules Using Directed Hypergraphs. *20th International Conference on Data Engeneering*, 2004.

[17] Y.H. Cho, J.K. Kim, and S.H. Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23(3):329–342, 2002.

[18] A. Clauset, M.E. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(066111), 2004.

[19] R. Cramer, I. Damgard, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT'01*, volume 2045 of *LNCS*, pages 280–299, 2001.

[20] C. Crepeau. Verifiable Disclosure of Secrets and Applications. *Advances in Cryptology – EUROCRYPT'89*, 434:150–154, 1989.

[21] C. Crepeau, J. Van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. *Advances in Cryptology – CRYPTO '95*, pages 110–110, 1995.

[22] I. Damgard and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography (PKC '01)*, pages 90–104, 2001.

[23] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 67–76, 2001.

[24] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res*, 30(7):1575–1584, April 2002.

[25] P. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography (FC'00)*, volume 1962 of *LNCS*, pages 90–104, 2000.

[26] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – EUROCRYPT'04*, volume 3027 of *LNCS*, pages 1–19, 2004.

[27] J. Frieser and L. Popelinsky. DIODA: Secure mining in horizontally partitioned data. In *Workshop on Privacy and Security Issues in Data Mining*, 2004.

[28] Keith Frikken. Privacy-preserving set union. In *Applied Cryptography and Network Security (ACNS'07)*, volume 4521 of *LNCS*, pages 237–252, 2007.

[29] T. Fukasawa, J. Wang, T. Takata, and M. Miyazaki. An effective distributed privacy-preserving data mining algorithm. In *Intelligent Data Engineering and Automated Learning (IDEAL'04)*, pages 320–325, 2004.

[30] J.A. Garay, P. MacKenzie, and K. Yang. Efficient and universally composable committed oblivious transfer and applications. volume 2951, pages 297–316. Springer, 2004.

[31] K. Gouda and M.J. Zaki. Efficiently mining maximal frequent itemsets. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 163–170. IEEE Computer Society, 2001.

[32] J. Han and J. Pei. Mining frequent patterns by pattern-growth: methodology and implications. *ACM SIGKDD Explorations Newsletter*, 2(2):14–20, 2000.

[33] M.C. Hao, U. Dayal, M. Hsu, T. Sprenger, and M.H. Gross. Visualization of directed associations in e-commerce transaction data. *Proceedings of VisSym*, 1:185–192, 2001.

[34] J.J. Heckman. Sample selection bias as a specification error. *Econometrica: Journal of the econometric society*, pages 153–161, 1979.

[35] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mininga general survey and comparison. *ACM SIGKDD Explorations Newsletter*, 2(1):58–64, 2000.

[36] S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. *Advances in Cryptology – EUROCRYPT'07*, 4515:97, 2007.

[37] W. Jing, L. Huang, Y. Luo, W. Xu, and Y. Yao. An algorithm for privacy-preserving quantitative association rules mining. In *IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 315–324, 2006.

[38] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE TKDE*, 16(9):1026–1037, 2004.

[39] L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology – CRYPTO'05*, volume 3621 of *LNCS*, pages 241–257, 2005.

[40] Jon Kleinberg and Steve Lawrence. The structure of the web. *Science*, 294:1849–1850, 11 2001.

[41] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. *Proceedings of the third international conference on Information and knowledge management*, pages 401–407, 1994.

[42] C.P. Massen and J.P.K. Doye. Identifying communities within energy landscapes. *Physical Review E*, 71(4):46101, 2005.

[43] C. Mauri. Card loyalty. A new emerging issue in grocery retailing. *Journal of Retailing and Consumer Services*, 10(1):13–25, 2003.

[44] K. McGarry. A survey of interestingness measures for knowledge discovery. *The knowledge engineering review*, 20(01):39–61, 2005.

[45] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[46] MEJ Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[47] MEJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):36104, 2006.

[48] MEJ Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):26113, 2004.

[49] P. Paillier. Public key cryptosystem based on composite degree residue classes. In *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238, 1999.

[50] C.R. Palmer and C. Faloutsos. Electricity based external similarity of categorical attributes. *Lecture notes in computer science*, pages 486–500, 2003.

[51] G. Pandey, S. Chawla, S. Poon, B. Arunasalam, and J.G. Davis. Association Rules Network: Definition and Applications. *Statistical Analysis and Data Mining*, 1(4), 2009.

[52] H. Polat and W. Du. Privacy-preserving top-n recommendation on horizontally partitioned data. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 725–731, 2005.

[53] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal Graph Alg. App*, 10(2):191–218, 2006.

[54] L. Qiu, Y. Li, and X. Wu. Preserving privacy in association rule mining with bloom filters. *Journal of Intelligent Information Systems*, 29(3):253–278, 2007.

[55] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663, 2004.

[56] A. Schuster, R. Wolf, and B. Gilburd. Privacy-preserving association rule mining in large-scale distributed systems. In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid'04)*, pages 411–418, 2004.

[57] K. Steinhaeuser and N.V. Chawla. Community detection in a large-scale real world social network. In *LNCS*. Springer Verlag, 2008.

[58] C. Su and K. Sakurai. Secure computation over distributed databases. *IPSJ Journal*, 2005.

[59] P.N. Tan, V. Kumar, and J. Srivastava. Selecting the right objective measure for association analysis. *Information Systems*, 29(4):293–313, 2004.

[60] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 404–413. ACM New York, NY, USA, 2006.

[61] H. Tong, C. Faloutsos, and J.Y. Pan. Fast random walk with restart and its applications. In *Proceedings of sixth IEEE International Conference on Data Mining*, pages 613–622, 2006.

[62] S. Urabe, J. Wong, E. Kodama, and T. Takata. A high collusion-resistant approach to distributed privacy-preserving data mining. In *IASTED International Multi-Conference: parallel and distributed computing and networks*, pages 326–331, 2007.

[63] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644, New York, NY, USA, 2002. ACM.

[64] V.S. Verykios, A.K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association Rule Hiding. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, pages 434–447, 2004.

[65] E. Wang, G. Lee, and Y. Lin. A novel method for protecting sensitive knowledge in association rules mining. In *International Computer Software and Applications Conference*, pages 511–516, 2005.

[66] J. Wang, T. Fukasawa, S. Urabe, T. Takata, and M. Miyazaki. Mining frequent patterns securely in distributed system. *IEICE Transactions on Information and Systems*, E89–D(11):2739–2747, 2006.

[67] P.C. Wong, P. Whitney, and J. Thomas. Visualizing association rules for text mining. In *1999 IEEE Symposium on Information Visualization, 1999.(Info Vis' 99) Proceedings*, pages 120–123, 1999.

[68] H. Xiong, P.N. Tan, and V. Kumar. Hyperclique pattern discovery. *Data Mining and Knowledge Discovery*, 13(2):219–242, 2006.

[69] B. Zadrozny. Learning and evaluating under sample selection bias. In *In Proceedings of the 21st International Conference on Machine Learning*, 2004.

[70] M.J. Zaki. Generating non-redundant association rules. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 34–43. ACM New York, NY, USA, 2000.

[71] M.J. Zaki and C.J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *2nd SIAM International Conference on Data Mining*, pages 457–473, 2002.

[72] M.J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, et al. New algorithms for fast discovery of association rules. In *3rd Intl. Conf. on Knowledge Discovery and Data Mining*, volume 20, 1997.

[73] Mohammed J Zaki. Parallel and distributed association mining: A survey. *IEEE Concurrency*, 7(4):14–25, 1999.

[74] J. Zhan, S. Matwin, and L. Chang. Privacy-preserving collaborative association rule mining. *Journal of Network and Computer Applications*, 30(3):1216–1227, 2007.

[75] N. Zhang, S. Wang, and W. Zhao. A new scheme on privacy preserving association rule mining. In *Knowledge Discovery in Databases (PKDD'04)*, pages 484–495, 2004.

[76] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 401–406. ACM New York, NY, USA, 2001.

[77] S. Zhong. Privacy-preserving algorithms for distributed mining of frequent itemsets. *Information Sicences*, 177(2):490–503, 2007.