NET Institute*

www.NETinst.org

Working Paper #08-18

October 2008

Modeling the Product Space as a Network

Nitesh Chawla, Troy Raeder, Marina Blanton, and Keith Frikken
University of Notre Dame

* The Networks, Electronic Commerce, and Telecommunications ("NET") Institute, http://www.NETinst.org, is a non-profit institution devoted to research on network industries, electronic commerce, telecommunications, the Internet, "virtual networks" comprised of computers that share the same technical standard or operating system, and on network issues in general.

Modeling the Product Space as a Network

Nitesh V. Chawla*, Troy Raeder, Marina Blanton, Keith Frikken
nchawla@cse.nd.edu
Department of Computer Science and Engineering
University of Notre Dame

October 22, 2008

Abstract

In the market basket setting, we are given a series of *transactions* each composed of one or more *items* and the goal is to find relationships between items, usually sets of items that tend to occur in the same transaction. Association rules, a popular approach for mining such data, are limited in the ability to express complex interactions between items. Our work defines some of these limitations and addresses them by modeling the set of transactions as a network. We develop both a general methodology for analyzing networks of products, and a privacy-preserving protocol such that product network information can be securely shared among stores. In general, our network based view of transactional data is able to infer relationships that are more expressive and expansive than those produced by a typical association rules analysis.

1 Introduction

Understanding consumer purchase behavior and even the product spread is becoming paramount in today's digital economy. Companies routinely collect data about their consumers, be it for banking, for online retailing or for supermarket retails. Loyalty cards have become commonplace as retailers try to understand the buying patterns of consumers [26]. These buying patterns influence the products, and the associated promotions, carried in the store.

The so-called *market basket* problem is typically understood as follows: the typical retail store sells a tremendously large set of products P. In a single transaction, a customer of the store buys a comparatively small subset, p, of those

*Corresponding Author

products. A *market basket* or *transactional* dataset is simply a list of transactions, where each transaction is a list of products.

This paper addresses two fundamental questions regarding the analysis of transactional data. First, how can we quickly uncover the important relationships in a large and heterogeneous set of transactions? The most common approach is to find association rules in the data. An association rule is a statement of the form $A \rightarrow B$, meaning "if A is bought, it is likely that B is also bought." Most association rules algorithms work by first generating a series of frequent itemsets, defined as sets of items that appear together in a large number of transactions. Exactly how large is a user-specified threshold called the minimum support. Then, they iterate through the itemsets and discover rules that meet a specified minimum confidence. The confidence of a rule $A \rightarrow B$ is defined as $P(B|A)$, the observed probability that B is bought given that A is also bought. The output of the algorithm is the set of rules that meet both criteria.

It has long been known, however (see, for example, [25]) that finding interesting rules, rather than obvious ones, in the output of an association rules algorithm is not a trivial task, The first part of this paper describes three properties of association rules that contribute to this difficulty, and explains a simple framework based on a network view of products that seems to overcome these limitations. We describe a procedure for modeling transactional data as a network, methods for analyzing the network, and finally we apply our methods to real-world data and show results.

The second half of the paper addresses a different problem: sample selection bias. As a store analyzes its own data, there is very little assurance that the data provide a representative sample of the desired population. It is merely a representation of the data coming into that store or the company. This biased view of the consumer base can quickly become limiting if the store or company tries to expand and reach out to a broader market.

Such sample selection bias may be overcome by the sharing of data among competing stores, but this data tends to be (justifiably) highly proprietary. Therefore, the stores need to be able to share nformation privately and yet capitalize on the benefits of information sharing to improve generalization of the models. We conjecture that it can be to the benefit of the marketplace to share information — the stores can be more strategic in product acquisition and placement, with respect to the items they already carry, and consumers can have a broader choice.

It is with this motivation that we develop a protocol for sharing networks of products securely. Specifically, we develop a protocol with which a series of n stores, selling ℓ products between them, participate in joint computation to securely determine c_{ij}, the number of times product i and product j have sold together in all stores combined (without revealing any information about the products that any one store, individually, sells). Each store chooses the products about which it would like to learn, and the stores jointly compute the c_{ij} without leaking unintended information to any of the participants. That is, each store only learns the total counts for products of its interest and other stores do not know what products were requested or what the counts were. We conduct a case study using real-world transaction data and demonstrate experimentally that our protocol allows stores to identify profitable products.

2 Related Work

The development of network models for commercial purposes is not, in itself, novel. Domingos and Richardson [10, 37] study the spread of influence through a social network algorithmically, as it relates to the adoption of products. The idea is that people tell their friends about products they have bought and therefore, marketing to a few influential individuals can cause widespread awareness about a product without a massive marketing campaign. Kempe et al. [22] take the idea further, developing a heuristic to determine the maximally influential set of nodes in a network.

Community detection in networks, in particular, has found application in a wide variety of domains including social network analysis [19, 40], the study of protein-protein interactions [2, 13], and discovering websites with common themes [24]. There is relatively little work, however, on the use of community detection in commercial domains. Perlich and Rosset [31] develop a community detection method to identify common sets of product options on trucks. Clauset et. al. [6] detect communities of products in a network of purchases from Amazon.com. This particular work is probably the most similar to ours, but our analysis goes far beyond what was done in that paper. Whereas [6] provides a brief overview of the communities found in the data, with relatively little discussion of their effectiveness, we develop general framework for the analysis of any market basket dataset and analyze the results in detail. For a survey of community detection methods in general, see [9].

With regard to information sharing, 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, [21] gave the first protocol followed by a large body of other works [1, 16, 18, 39, 46, 32, 41, 43, 20, 44, 34, 42, 45, 47]. 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 decide information about which items 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 to 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 forward 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

Table 1: Our transaction data, broken down by the number of unique items in a transaction. Single-item transactions are by far the most common.

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%

Table 2: Revenues for the full set of transactions, broken down by transaction size. Single-item transactions generate more revenue than any other single size. However, we see that most of the store's money comes from multi-item transactions

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%

by using threshold encryption, but other expensive zero-knowledge techniques can be left out.

3 Data

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. The only data we have is a listing of transactions without any association with any customer or type of payment mode or any other identifying information. Tables 1 and 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.. 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 than 4 items, account for almost 8% of revenues and less than 2% of checkouts. There is clearly an advantage, then, to understanding the interactions between products, as the store has great growth potential. By making even a fraction of the single-item transactions larger, we could earn a great deal of money. We would like to note that this is also a nature of the convenience store on campus. We believe larger departmental stores and/or supermarkets will comprise transactions with 2 or more items as being most frequent as well as being most profitable.

4 Limitations of Association Rules

Before describing our framework for market basket analysis using networks, we outline a few of the limitations of association rules analysis that we aim to adderss. The first of these is *parameter sensitivity*. One needs to choose values of support and confidence that bring out the interesting patterns of interaction among products without simply stating the obvious. However, small changes in either of the parameters can lead to significant differences in the number of rules discovered, and there is no generally-accepted method for finding appropriate values.

The second limitation is an inherent *completeness-granularity tradeoff*. Frequent itemsets may not capture effectively the interactions between products if there are too many distinct products in the same category. For example, Coke, Pepsi, Dr. Pepper, and various store brands are all types of cola. If we consider each of these products individually, association rules may not discover anything meaningful without unreasonably low support and confidence limits. One could overcome this limitation by aggregating all types of cola into a single product, but doing so will undoubtedly compromise the granularity of the analysis. There are certain situations where it is beneficial to capture interactions at the level of individual products rather than classes of products.

One association that is often cited is beer and diapers: people who buy beer also often buy diapers. But, what if we want to ask – is it Budweiser and Pampers or Stella Artois and Pampers or replace Pampers with Huggies? These questions are important as their answers may reflect on the demographic of an individual and they enable more personalized marketing campaigns.

Finally, association rules can have *limited scope*. An individual association rule can only express a relationship between two different products or sets of products. For more complex relationships, association rules become unwieldy. To express a tight mutual correlation between n products, for example, may require as many as $O(n^2)$ different association rules.

5 Association Rules from Store Data

To discover association rules on our data, we used the Apriori algorithm implementation provided by Borgelt[3]. The default values of support and confidence did not generate any rules. We then tried a variety of support values, until we settled at the support of 0.01%. Figure 1 shows the top 50 association rules found on an entire year's worth of purchase data, with minimum support of 0.01 percent, where the minimum confidence comes in at just under 30%.

There are many definitions that we could have chosen for "top 50", but we feel that this particular set of rules presents an honest view of the store. Since most of our transactions are single-item, we chose low support and high confidence over high support and low confidence. Increasing support and confidence results in a much sparser rule set. As Figure 3 will also demonstrate, high support and low confidence do not result in a significant rule-set. The rules are dominated by some of the store's top sellers, with Bagel, Cream Cheese, and Coffee appearing in 31 of 50 rules. Most of the rules generated are no surprise as they match the intuition. However, we posit that the key here is to be able

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 1: The top 50 association rules on one year's worth of purchase data at our store. A ¡- B means If B is bought then A is bought. Minimum support was held at 0.01 percent, and the numbers in parentheses represent the confidence of the rule.

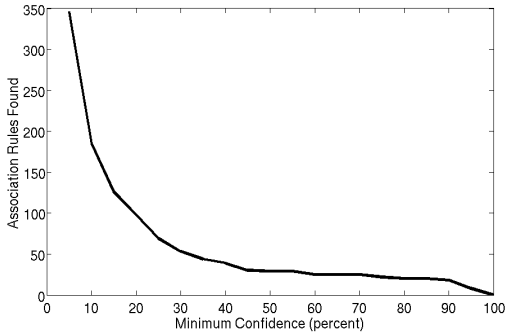


Figure 2: Number of association rules found at various levels of minimum confidence when minimum support = 0.01%.

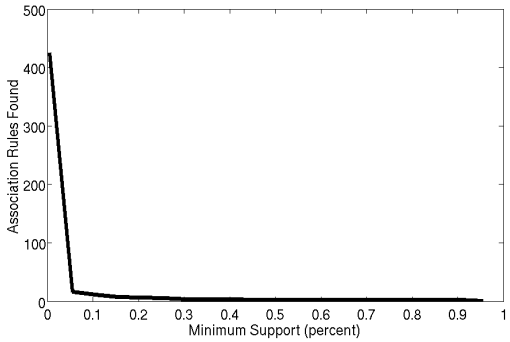


Figure 3: Number of association rules found at various levels of minimum support when minimum confidence = 10%.

to discover interactions among items or products that may miss our intuition and provide new business opportunities. This motivates the exploration of network-based view of the products space.

The following series of figures further highlights the difficulty of finding effective association rules in our data. Figure 3 shows the number of association rules found in the data as minimum confidence is held constant at 10% and minimum support is varied from 0.005% to 1%. We see that the number of rules decreases dramatically in an incredibly small window below 0.1%. In practice, determining the cutoff point where useful rules end and superfluous rules begin is difficult. Figure 2 shows a similar result where minimum support is held constant at 0.01% and minimum confidence varies from 5% to 100%. Figure 4 shows the number of unique 2-item association rules found under the same conditions (a 2-item association rule is one of the form $A \rightarrow B$ where A and B are each single-item sets). In combination, they show that there is a non-trivial number of association rules at very high confidence, but almost all of them are multi-item. Looking further, we find that most of them are of the form {Cream Cheese, X} $\rightarrow$ Bagel, where X is some other breakfast product. These rules are all explained adequately by the single rule Cream Cheese $\rightarrow$ Bagel, which has a confidence of 93%. Beyond this initial spike, we see something very similar to what we see in Figure 3: a very rapid increase in the number of association rules in a very small window. Taken together, these observations suggest that finding appropriate values for the minimum support and confidence of association rules is a fairly difficult task. Our framework attempts to remove this burden from the user.

6 Products as a Network

We posit that the network view of products or items is more compelling than the popular methodology of association rules. We proceed as follows. First, the set of transactions is modeled as a network, in which each product is a vertex and two products are connected by an edge if and only if they have been bought together. Next, we prune low-weight edges from the network. We define the weight of an edge connecting A and B to be the number of times products A and B appear in the same transaction. We have found that our product networks tend to be very dense and that removing low-weight edges helps to accentuate the network's inherent community structure. More specifically, we define a minimum edge weight w, such that an edge exists between two vertices if and only if they have appeared

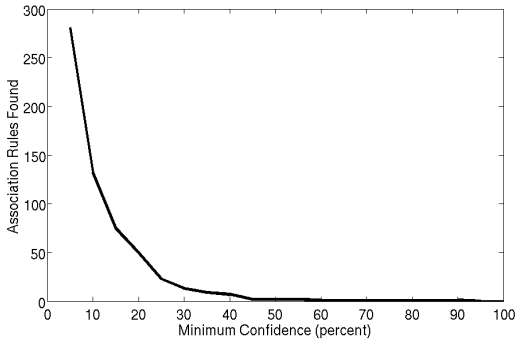


Figure 4: 2-item association rules found at various levels of minimum confidence when minimum support = 0.01%.

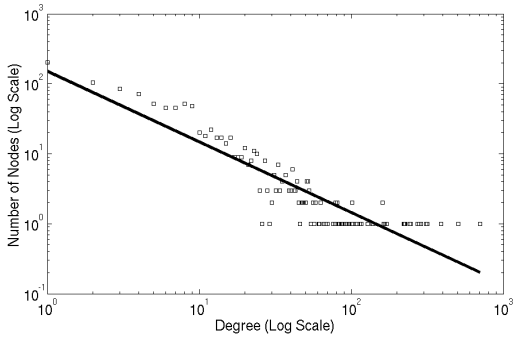


Figure 5: The number of nodes with a given degree (number of neighbors) in our network of products, plotted along with the line of best fit on a log-log scale. The linear fit shows that the network is scale-free.

together in at least w different transactions.

We observe that purchasing behavior at a store carries a long tailed effect — fewer products are bought by many and many products are bought by few. The consequence of this, from a network perspective, is that the network is composed of a few popular hubs with hundreds of neighbors each, and hundreds of items that are bought with only a few others. Figure 5 shows this trend.

Once we have the network in its final form, we apply a community detection algorithm to the resulting product space network. It is the analysis of the resulting communities that reveals the important structure and relationships among the products. We also assert that one may run association rules within these communities. Since elements of individual communities will tend do be more closely related, this will give higher support and confidence to more meaningful rules. We also propose a maximum spanning tree construction for large communities, which will prune edges while maintaining the essential structure. Specifically, we construct a *maximum-influence spanning tree*, which is a maximum spanning tree, where the edge from A to B is weighted with co-occurrence of the two items or products (we capture the conditional of $P(B|A)$). The tree is constructed such that the sum of all edge weights is maximized

and the root is chosen arbitrarily. It is not guaranteed that the parent of any given item in the tree will be the product which has the most influence on it — if, for example, A has the greatest influence on B and B has the greatest influence on A. Still, this construction seems like the most tractable way to encourage closely-related products to be near each other in the tree.

6.1 Constructing a Community of Products

Our goal is to find some sort of regular structure in the network or graph of products that we can exploit to gain a level of understanding about products in our store. Once a network of products is constructed, we are interested in discovering communities or clusters of interest. The problem of *community detection* or *clustering*[1] in graphs is usually defined as follows: Given a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, find a series of *clusters* or *communities* $C = \{G_1, G_2, ..., G_n\}$ which maximizes a utility function $f(C)$. Define $G_i = (V_i, E_i)$, with $V_i \subset V, E_i \subset E$, and all $\{V_i\}$ and $\{E_i\}$ disjoint. Typically, it is assumed that $V = \bigcup_{i=1}^{n} V_i$, meaning that each node is included in a cluster, although this need not be the case.

Intuitively, communities are sets of nodes that are more strongly connected to each other than they are to the rest of the graph or network. For our purposes, we wish to find groups of products that are closely related to one another so that we might identify relationships are unclear in an analysis of the full product space. Numerous algorithms for community detection exist in the literature. We used the Fast Modularity algorithm of [6] for community detection in this work, mostly because it is efficient, effective, and parameter-free.

At a high level, the algorithm attempts to find an assignment of elements into communities such that the *modularity* of the decomposition is maximized. The modularity[28] Q of a set of communities is defined as:

$$Q = \sum_i (e_{ii} - a_i^2) \tag{1}$$

where e_{ii} is the fraction of edges that join vertices in community i to other vertices in community i and a_i is the fraction of edge endpoints that lie in community i. Modularity measures the difference between the number of in-cluster edges in a given clustering and the expected number of in-cluster edges in a completely random clustering with the same number of nodes and clusters. The intuition behind modularity is that, in a strong set of communities, we would like to see a large number of edges within communities and a small number of edges between communities. Specifically, strong communities are those in which the number of inter-community edges is significantly smaller than the *expected value* of the number of inter-community edges in a random network of the same size and community decomposition. Modularity has become the de facto standard measure for the quality of a community decomposition and has been widely adopted for both validation and comparison of community structures [29, 33], and as an objective function for optimization algorithms to identify communities [6, 11, 12, 29, 35, 38].

[1] It is worth noting that we use the terms "cluster" and "community" (similarly "clustering" and "decomposition into communities") interchangeably in this paper.

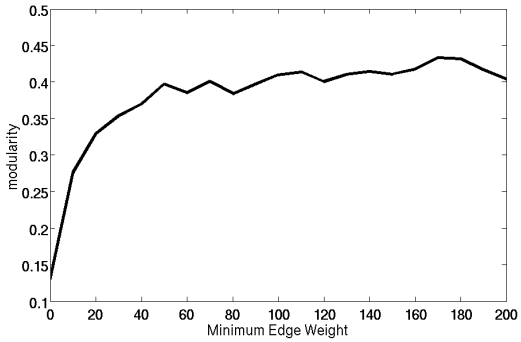


Figure 6: Modularity of the communities found by the Fast Modularity algorithm as a function of minimum edge weight.

Since optimizing modularity directly (i.e. by trying all possible partitions) is computationally infeasible [4], the Fast Modularity algorithm uses modularity to guide a hierarchical agglomeration process. Initially, each node is assigned to its own cluster. The task is thereby reduced to (repeatedly) finding an appropriate pair of nodes or communities, connected by an edge, to merge into a single new community.

In its raw form, the density of our network makes it difficult for community-detection algorithms to find meaningful communities. Therefore, we wish to remove insignificant edges in order to accentuate the inherent network's inherent community structure. To do this, we define parameter w, the minimum edge weight, such that an edge appears between two products only if they have appeared together in at least w transactions. To determine an appropriate value for w, we propose an empirical wrapper search that evaluates different values with the goal of optimizing the quality of clustering. We use modularity as an objective optimization criterion to select the appropriate value of w for a dataset. Figure 6 presents an evaluation of the effects of w on community structure or discovering clusters. It presents the *modularity* of the clusters found by the fast community detection algorithm of Clauset et. al. [6] as w varies from 0 to 200. Modularity increases steadily up until $w = 50$, at which point it begins to oscillate. We retained $w = 50$ for our experiments. Qualitatively, a higher value of w would have led to an elimination of certain products that are tied together by less frequent purchases, but are still interesting. A lower value will preserve edges between products or items that have been bought together in a much smaller number of transactions, potentially noise.

In our experience, the process that we have outlined above — continuously increasing the minimum edge weight threshold until the modularity of the resulting clustering plateaus, has produced very good clusters (including the ones presented later in the paper). As a result, we believe that this procedure is a valid way to choose the parameter w and that a practitioner should not have to guess a value for the parameter himself.

Given the pruned network, the Fast Modularity algorithm found a total of 16 communities of products, each of which was non-trivial, meaning that it contained at least two products. The visualization and analysis of these 15 communities provides the basis of the results which we discuss in the results

7 Secure Sharing of Network Data

Now that we have outlined our basic procedure for constructing and manipulating product networks, we introduce our protocol, PPC, for sharing this information securely. Specifically, we share the information necessary to construct a product network from the aggregate data of all participating stores. In other words, for each pair of products, p_i and p_j, the participants, at the end of the protocol receive c_{ij}, the count of the total number of times product i and product j are bought together in the combination of all stores.

8 Preliminaries of the Private Product Correlation (PPC) Protocol

8.1 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 [30]. 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 $E_{pk}(m_1) \cdot E_{pk}(m_2) = E_{pk}(m_1 + m_2)$ and $E_{pk}(m)^c = E_{pk}(m \cdot c)$. 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 0.

Our protocols use a threshold version of homomorphic encryption. Threshold Paillier encryption was developed in [14, 8, 7]. 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.

8.2 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. [7] give a zero-knowledge protocol for this proof using Paillier homomorphic encryption, which is the type of encryption used in this work.

8.3 Privacy-preserving set operations

Prior literature [15, 23, 17] 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 [17], which is the fastest protocol for computing the union of two sets.

9 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 number of products to which we refer as L_i. We assume that a unique naming convention is used, and different participants will use the same name for a particular product.

9.1 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 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 of recover the counts for the products of interest.

The existing techniques, however, do not allow this functionality to be implemented in the above form due to the following reasons. 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).

9.2 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 $\mathcal{P}_i$[2]. $\mathcal{P}_i$ 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 $E(q_{m_i}), \ldots, E(q_0)$ 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 $0 \leq j, k \leq \ell$ $(j \neq k)$.
4. Each $\mathcal{P}_i$ locally computes the encryption of the sum of all counts for each product pair p_i, 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 $p_j \in L$.
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 $E_{pk}(Q_i(p_1))$, $E(Q_i(p_2))$, …, $E_{pk}(Q_i(p_\ell))$. 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 $E_{pk}(b^i_j)$. Note that b^i_j 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 $E_{pk}(b^i_j) \cdot E_{pk}(c_{jk}) = E_{pk}(b^i_j + c_{jk})$. Note that the encrypted value is c_{jk} if $p_j \in D_i$ and is a random value otherwise.

[2]It goes without saying that this set can be inflated with fake items to hide the actual size of the set of items in which the participant is interested.

(d) The parties engage in a joint decryption protocol to reveal the values $b^i_j + c_{jk}$ to the $\mathcal{P}_i$ for all $0 \leq j, k \leq \ell$.

9.3 Sub-protocols

NZProof Protocol: A user has a ciphertext c and would like to prove that the corresponding ciphertext 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, so must a.

NZRM Protocol: Given a ciphertext $c = E_{pk}(a)$, a participant encrypts 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.

9.4 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 $O(n^2\ell^2 + n^3\ell)$.

10 Security Analysis

In this paper we argue security using the standard simulation argument. That is, we show that for any adversary in our protocol, there exists a poly-time adversary in the ideal model (where the participants reveal their information to a fully-trusted third party (TTP)) that achieves the "same" effect as the real adversary. Thus the protocol is as secure as the ideal protocol. 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 set 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 $p_j \in D_i$.

Essentially, the above ideal model allows participants to change their count values based on the result of the set union and it also reveals an upper bound on the size of the D_i and L_i sets. 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 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 the adversary submits to the TTP. In Step 2, the adversary must submit a set of items to the set union protocol (we are assuming the existence of a maliciously secure set union protocol), which is exactly what it submits to the TTP. After receiving the results from Step 2, the adversary submits a list of count values, but whatever values he submits can also be submitted to the TTP in the ideal model. Step 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 [5], 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 poly-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).

11 Efficiency Improvements

11.1 Polynomial multiplication and evaluation

As described in [15], polynomial evaluation can be performed more efficiently by applying Horner's rule. Recall from section 8.3 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 [17] 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.,[27] 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.

11.2 Packing

Assume that the c_{ij} values are no larger than 2^M. It is likely that such a bound can be found, where $M \ll \rho$ where ρ is the number of bits in the modulus of the homomorphic encryption scheme. In Step 3 of the above protocol the number of encryptions that a participant posts is ℓ, however we can compress these values by storing $s = \lfloor \frac{\rho}{M} \rfloor$ values in a single encryption, and thus this new scheme 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 $E(\sum_{i=1}^{s} 2^{M(i-1)} x_i)$. Note that as long as individual results do not get larger than M bits, that we can add to such compressed encryptions (to obtain the value representing the pairwise values) and we can multiply by constants. After doing this compression it 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.

12 Results

Having outlined in detail the constructions of product networks, our methods for community detection in such networks, and our protocol for sharing this network data securely, we now present some experimental results. First, we look at the communities of products created by the methods in Section 6, and examine the extent to which the conclusions that can be drawn from these communities corroborate and extend the conclusions that follow from association rules analysis.

Next, we simulate participation in the PPC protocol among several fictitious stores built from subsets of our dataset. Here, we examine the extent to which participation in the protocol improves a store's impression of the "true" customer distribution by examining the change in revenue as a result of decisions made based on the protocol output.

Both sets of experiments were run on the Notre Dame on-campus store dataset described in Section 3

12.1 Analysis of Product Networks

Given a series of communities, constructed as we outlined in section 6, we seek to draw definitive conclusions about the behavior of customers and the role of products in our store. The following experiments were conducted on networks constructed from transactions occurring in the calendar year 2006 with a minimum edge weight w = 50 transactions. We use the year 2006 because it has a large number of transactions and is relatively recent. The store introduces and phases out products over time, such that older data are more likely to contain items no longer available in the store. The year in consideration should not make a difference to the relative observations — we wanted to be (more) current in our observations for the store.

COFFEE	<- NEWSPAPER_CHICAGO_TR BAGEL	(85.6)
BAGEL	<- NEWSPAPER_CHICAGO_TR COFFEE	(73.9)
COFFEE	<- NEWSPAPER_NEW_YORK_T BAGEL	(75.2)
BAGEL	<- NEWSPAPER_NEW_YORK_T COFFEE	(41.5)
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	<- 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 COFFEE	(96.6)

Figure 7: The 13 association rules whose constituent products did not land in the same community. Again, A ¡- B means if B is bought then A is bought, and the numbers in parentheses are the confidence of the rule.

12.1.1 Association Rules and Communities of Products

Before evaluating the structure and contents of individual communities, we investigate the preservation of association rules within communities. We say a set of communities *preserves* the association rule $A \rightarrow B$ if all elements of the sets A and B are in the same community. Clearly, if a set of communities preserves the association rules for a dataset, then analysis of the individual communities will reveal at least as much information as the association rules themselves (that is to say, nothing will be lost in the analysis).

The communities found by the *FASTQ* algorithm on our store data preserved 37 of the top 50 association rules laid out in figure 1. The rules that were not preserved appear in figure 7. We see that 9 of the 13 "broken" association rules are either of the form $\{$Bagel, X$\} \rightarrow$ Cream Cheese or $\{$Cream Cheese, X$\} \rightarrow$ Bagel. These rules are actually just redundant indicators of the incredibly strong relationship between Bagel and Cream Cheese, which is ably represented by the preserved association rule Cream Cheese $\rightarrow$ Bagel, at a confidence of 93%.

The remaining four rules are relationships between bagels, coffee, and newspapers. They break simply because Coffee and Bagel are not placed within the same community. This may seem odd, since coffee and bagels are both major breakfast foods. The difficulty, however, arises from the fact that coffee is bought a great deal throughout the day and bagel is bought mostly in the morning. This means that customers buy coffee with a much more diverse array of products than they do bagels. As a result, coffee is actually very difficult to place in any community, whereas bagel groups more naturally in with breakfast foods. Notice that the simple rule "Coffee $\rightarrow$ Bagel" is not one of the top 50 rules. These two items are not as well connected to each other as they are to other products.

As a result of the preceding analysis, we contend that our decomposition into communities preserves the vast majority of the important association rules within our data. As a result, we can apply our framework without fear of destroying the insights obtained from association rules analysis.

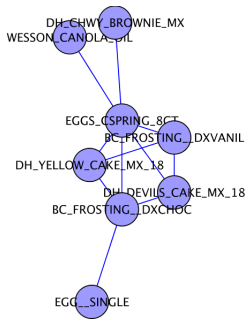


Figure 8: A small cluster found by all three clustering algorithms. The Eggs in the center connect to every other product except the single egg.

12.1.2 Analysis of Discovered Communities

Figure 8 shows one of the smaller clusters found in the data. The structure of this particular cluster is simple, with Eggs as a hub to which every other product connects. Each item in the cluster is a baking product: two types of frosting, two types of cake mix, cooking oil, and brownie mix. The immediate implication is that when customers buy eggs in this particular store, they buy them for baking. Further examination of the numbers shows that this conclusion is overwhelmingly true.

There were 541 distinct products bought with Eggs[3] 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 six neighbors appears in over 39% of all transactions containing eggs. This is an incredibly strong connection in a store where the majority of transactions contain only one unique item.

The rest of the cluster is extremely strong as well. In particular, frosting of some type is bought in over 70% of the transactions containing cake mix, while brownie mix and oil both share a very significant connection with eggs. As such, figure 8 demonstrates the ability of the community detection algorithms to succinctly capture complex relationships. The structure of this cluster is described by 11 association rules on the original dataset, ranging in support from 0.011% to 0.008%. In fact, there are 390 rules found in the dataset at the level of support and confidence that is required to entirely describe the cluster.

For a completely effective analysis, we need to quantify the impact of the items in this cluster. Intuitively, it seems that 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:

$$E(Sales) = P(Eggs|CakeMix) * Price(Eggs) + \\ P(Egg|CakeMix) * Price(Egg) + \\ P(Frosting|CakeMix) * Price(Frosting)$$

[3]For simplicity, we use "eggs" to refer to the product EGGS_CSPRING_8CT

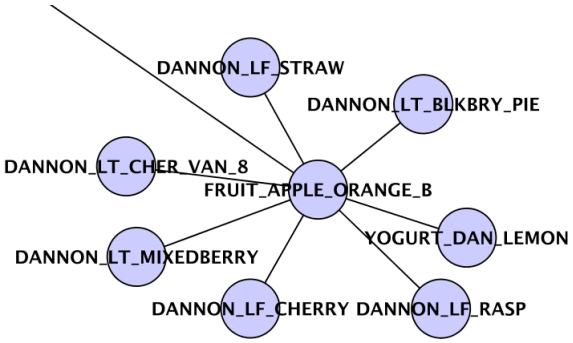


Figure 9: A small subtree, showing fruit connected to several yogurts, in the spanning tree of a large cluster. Yogurt is bought frequently with fruit, but the variety of yogurts in the store obscures this relationship in association rules.

and we find that, even by this fairly pessimistic estimate, the store can expect to generate $2.30 in additional sales from each cake mix sold. Therefore, if the store can run a promotion that increases the sales of cake mix at a cost of less than $2.30 per transaction, it stands to make additional money. 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 clustering can be used effectively to narrow the search for profitable promotions in a store.

Going back to the original top-50 association rules of figure 1, there are two association rules which pertain to this cluster: Canola Oil $\rightarrow$ Eggs, and Vanilla Cake Mix $\rightarrow$ Chocolate Frosting. While these two rules do hint, to some extent, at the relationship depicted in figure 8, the cluster explains it more completely and concisely. The rules allow us to infer that people who buy oil usually buy eggs, and people who buy frosting usually buy cake mix. The relationship of eggs with cake mix and eggs with brownie mix, as well as the existence of the single egg are hidden from view. Whereas the rules show an egg-oil relationship and a cake-frosting relationship existing in isolation, the cluster shows a densely-connected community of products centered around the Eggs. Furthermore, the promotion identified above is not evident in the association rules but is very clear in the cluster. Therefore, we see that our framework's expressive power is superior to that of association rules.

Figure 9 shows a portion of a maximum spanning tree constructed from a much larger cluster of products. This particular section of the tree depicts the parent, a generic Fruit product, surrounded by its children, a series of yogurts. Further investigation reveals that 10% of all fruit sales come in transactions that also contain yogurt. In aggregate, yogurt becomes the most frequently-purchased product with fruit (at 9.5% of all transactions) and fruit, with a very similar number, is the most frequently-purchased product with yogurt. The runner-up in fruit transactions, coffee, appears only 8% of the time. It is worth noting that coffee is easily one of the most popular items in the store, with five times as many sales as yogurt. The large number of co-occurrences of fruit and yogurt, then, is not a fluke brought about by an unreasonably large number of yogurt purchases. It represents a significant association not found in the original data.

The most significant impact of figure 9, though, is that it demonstrates the ability of our framework to find meaningful relationships between products without sacrificing granularity. The clear implication is that fruit and yogurt are

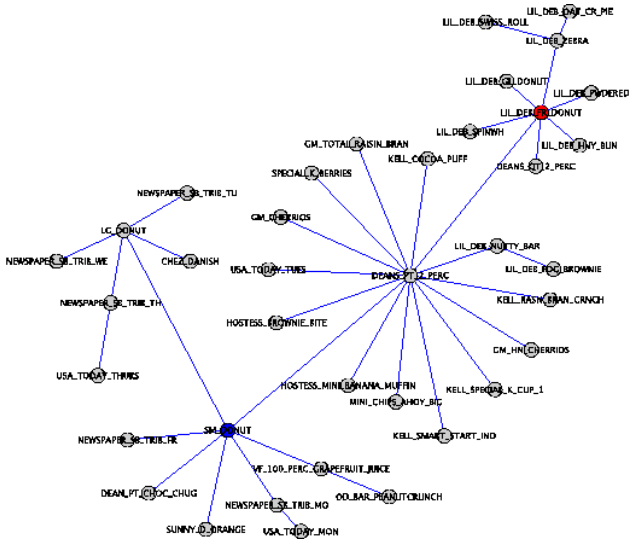


Figure 10: A small portion of the spanning tree for a large cluster.

strongly related, and this is supported by the numbers above. However, this cluster would allow the interested store owner to investigate these five types of yogurt in particular and compare them to those which are not connected to Fruit. In doing so, he may find that there is something special about these six particular products which sets them apart from the rest.

A similar result appears in figure 10: a portion of the spanning tree for a cluster of mostly breakfast food. Of particular interest in this portion of the tree is a series of "donut" products. The small donut (SM DONUT, shown in blue) connects to several products such as newspapers, orange juice, and more donuts, that are typically bought in the morning. Contrast that with the Little Debbie Donut (LIL DEB FR DONUT, shown in red) on the right. It connects mostly to other Little Debbie snack products. These relationships would seem to suggest that people prefer the small donut in the morning, when they are having their breakfast, and the Little Debbie Donut as a snack food at other times of the day.

Figure 11 shows histograms detailing purchases of the Little Debbie Donut and the Small Donut throughout the day. The x-axis is split in half-hour bins beginning at midnight and the y-axis shows the number of times each product was purchased in that half-hour interval. The figure corroborates our intuitions about the role of the two products in our store. Small Donut has a significant maximum in the 9:00-9:30 AM interval and almost nothing before 7:00 AM or after 8:00 PM. The Little Debbie Donut is much more evenly-distributed throughout the day. Its maximum occurs later, from 10:00-10:30 AM, which is approximately in line with the time at which the store, as a whole, sees its peak sales. Furthermore, it has a significant presence in the late-night and early-morning hours. Even though the Small Donut has far greater sales overall, the Little Debbie Donut outsells it by more than a factor of 5 in the half-hour between midnight and 12:30 AM.

The preceeding analysis suggests that, in contrast to the fruit-and-yogurt relationships of figure 9, it would be a mistake

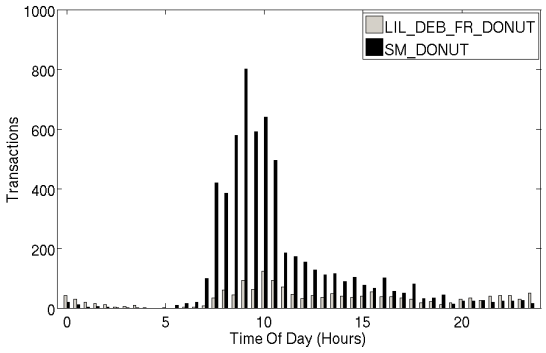


Figure 11: Histograms showing the different distributions for the purchases for Small Donut (dark) and Little Debbie Donut (light) throughout the day.

to combine the Small Donut and Little Debbie Donut into a single entity. The two products are bought at different times of the day for different purposes and do not simply represent two separate instances of the same product.

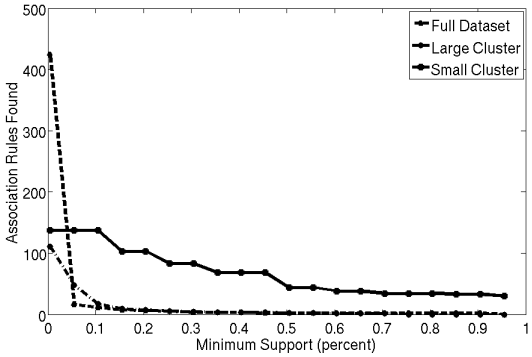
These examples serve to illustrate the power of community detection as applied to market basket data. Whereas association rules capture simple correlations such as "people who buy cake also usually buy frosting", a simple community was able to suggest the broader conclusion that "people who buy eggs are likely to be baking." Additionally we were able to derive, from the structure of a cluster, a potentially profitable promotion. Inspection of a larger cluster showed that the spanning trees that we construct from communities will place similar products near to one another and, therefore, are capable of differentiating between seemingly similar products that have different roles in the store.

A quick inspection of figure 1 shows that nothing relating to Fruit, Yogurt, the Small Donut or the Little Debbie Donut appear in the top 50 association rules. This entire portion of the discussion, then, is made possible by the construction of a product network.

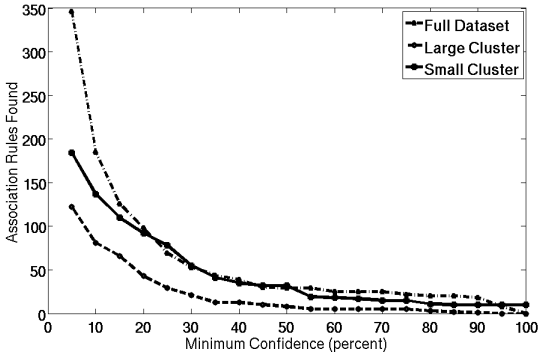
12.1.3 Association Rule Discovery in Communities

Our framework can enable more specific association rules discovery within communities of products, discovering rules and interactions that were not found when applying the Apriori algorithm on the entire data set.

To demonstrate the efficacy of the same, we considered a larger cluster or community of products with 70 items, and a smaller cluster of products with 9 items. Figure 12 shows the number of rules discovered at varying levels of support at a confidence of 10%, and at varying levels of confidence with support at 0.01%. The trend is shown for a small cluster, a large cluster, and the entire dataset (as shown before). It is evident from the figure that there is a huge drop in the number of rules at the cluster level. Moreover, at the small cluster levels, more rules are discovered at higher support as well, which is not surprising as there are tighter interactions among products and items in a smaller and



(a) Number of rules at varying levels of support with confidence of 10%.



(b) Number of rules at varying levels of confidence with support of 0.01%.

Figure 12: Number of Association Rules.

more cohesive cluster. This also results in highly redundant rules, so at the smaller cluster size, we believe a higher level of support and confidence will result in the core rules from the data.

The nature of rules is also very different. While the product communities support most of the rules discovered from the entire dataset, they also discover additional insightful rules. This is compelling as more targeted promotional items can thus be generated. For instance, Figure 12.1.3 shows some of the rules that were discovered from the largest and smallest clusters, but were missed on the entire dataset.

12.2 Simulation of the PPC Protocol

In order to evaluate the usefulness of the information obtained from the PPC protocol, we ran a number of simulations based on our transaction data. Since we do not have different stores available, we partition the existing store into multiple stores to evaluate the efficacy of the protocol. We are able to partition at both the product (or item) level, and

DEAN_PT_SKIM_CHUG	<-	KELL_SPECIAL_K_CUP_1	(30.1)
DEAN_PT_SKIM_CHUG	<-	KELL_SMART_START_IND	(54.0)
DEAN_PT_SKIM_CHUG	<-	GM_HN_CHERRIOS	(33.6)
DEAN_PT_SKIM_CHUG	<-	GM_TOTAL_RAISIN_BRAN	(53.7)
DEAN_PT_SKIM_CHUG	<-	SPECIAL_K_BERRIES	(41.0)
DEAN_PT_SKIM_CHUG	<-	GM_CINN_TOAST_CRUNCH	(33.3)
DEAN_PT_SKIM_CHUG	<-	KELL_RASN_BRAN_CRNCH	(48.7)
DEANS_PT_2_PERC	<-	KELL_FROSTFLAKE_CUP	(30.8)
COFFEE	<-	NEWSPAPER_SB_TRIB_TU	(30.5)
COFFEE	<-	BREAD_WEDGE	(41.2)
COFFEE	<-	IRISH_SODA_BREAD	(30.3)
COFFEE	<-	NUT_BREAD_SLICE	(52.9)
COFFEE	<-	LG_DONUT	(33.4)
COFFEE	<-	MUFFIN	(31.8)
COFFEE	<-	NEWSPAPER_SB_TRIB_MO LG_DONUT	(80.0)
COFFEE	<-	NEWSPAPER_SB_TRIB_WE LG_DONUT	(75.0)
COFFEE	<-	NEWSPAPER_SB_TRIB_TU NUT_BREAD_SLICE	(77.8)
COFFEE	<-	NEWSPAPER_SB_TRIB_TU LG_DONUT	(93.3)
COFFEE	<-	NEWSPAPER_SB_TRIB_TH LG_DONUT	(85.7)
COFFEE	<-	BREAD_WEDGE NUT_BREAD_SLICE	(41.2)
COFFEE	<-	IRISH_SODA_BREAD NUT_BREAD_SLICE	(45.5)

(a) Association Rules from a large cluster or community of products.

BC_FROSTING__DXVANIL	<-	DH_DEVILS_CAKE_MX_18	(35.6)
BC_FROSTING__DXCHOC	<-	DH_DEVILS_CAKE_MX_18	(35.6)
EGGS_CSPRING_8CT	<-	DH_DEVILS_CAKE_MX_18	(35.6)
BC_FROSTING__DXCHOC	<-	WESSON_CANOLA_OIL DH_YELLOW_CAKE_MX_18	(75.0)
DH_YELLOW_CAKE_MX_18	<-	WESSON_CANOLA_OIL BC_FROSTING__DXCHOC	(54.5)
EGGS_CSPRING_8CT	<-	WESSON_CANOLA_OIL BC_FROSTING__DXVANIL	(83.3)
BC_FROSTING__DXCHOC	<-	DH_YELLOW_CAKE_MX_18 EGG__SINGLE	(38.5)
DH_YELLOW_CAKE_MX_18	<-	EGG__SINGLE BC_FROSTING__DXCHOC	(41.7)
BC_FROSTING__DXCHOC	<-	DH_YELLOW_CAKE_MX_18 EGGS_CSPRING_8CT	(52.9)
DH_YELLOW_CAKE_MX_18	<-	BC_FROSTING__DXCHOC EGGS_CSPRING_8CT	(37.5)
EGGS_CSPRING_8CT	<-	DH_DEVILS_CAKE_MX_18 BC_FROSTING__DXVANIL	(34.4)
BC_FROSTING__DXVANIL	<-	DH_DEVILS_CAKE_MX_18 EGGS_CSPRING_8CT	(34.4)
DH_DEVILS_CAKE_MX_18	<-	BC_FROSTING__DXVANIL EGGS_CSPRING_8CT	(57.9)
EGGS_CSPRING_8CT	<-	DH_DEVILS_CAKE_MX_18 BC_FROSTING__DXCHOC	(31.2)
BC_FROSTING__DXCHOC	<-	DH_DEVILS_CAKE_MX_18 EGGS_CSPRING_8CT	(31.2)
DH_DEVILS_CAKE_MX_18	<-	BC_FROSTING__DXCHOC EGGS_CSPRING_8CT	(41.7)

(b) Association rules from a small cluster or community of products.

Figure 13: Association Rules on the Communities of Products

also at the transaction level. That is some stores have some products only or some stores have relatively fewer sales of some products. We believe this reflects the general distribution of participating stores — each may have unique and overlapping products, and there may also be differences in the sales volume. Thus, we consider three over-arching scenarios.

In scenario 1, we naively partition the product space at random, assigning each store an equal number of products. There will be some overlap between the stores' offerings, but it will be infrequent and haphazard. Intuitively, we would expect a substantial benefit from collaboration under these conditions, as stores will learn about more popular items from the other stores that are lucky enough to have them. In scenario 2, we observe that, in reality, no store is ever formed at random. Certain basic, popular products are commonly known to be profitable. Therefore, we start each store with a common set of popular products and they fill their remaining products at random. Thus, each store will have some items unique to it, but there is substantial overlap between stores. In this case, it will be more difficult to improve the store through collaboration, because the best products are already known and accounted for. Finally, in scenario 3, we partition the transaction space. That is, the stores carry identical, or nearly identical, sets of items but differ in the sales volume of these items. In this case, the stores wish to collaborate in the hopes that their combined information will better reflect the "true" purchase distribution. Put differently, the stores wish to compensate for any unknown sample selection bias in their data.

The point here is to simply demonstrate the utility of PPC and information sharing. The goal is to enable a collaborative marketplace environment, wherein there is an incentive for the stores to share their network of products, albeit privately. Not all stores will want to expand in exactly the same dimensions. Thus, at the end of the protocol, the stores pick up whatever products are deemed "best" that they don't already have. The offerings of the stores will naturally overlap, since that is the only situation in which a protocol like ours makes sense. The idea of this simulation is that, after participating in the PPC protocol, each store will add products that sell well with the products in its store, which will increase its total revenue. This also allows for certain stores to continue with their unique or specific offerings.

12.2.1 Division by Products

In our first two experiments, we partition the store's set of available products among our participating fictitious stores. Specifically, given a number of collaborating stores n, we assign a set of ℓ_i products to each store. The stores then acquire additional ℓ_i products to make $2\ell_i$ products in the store. The products are selected either at random or through collaboration, allowing us to compare the advantages offered by collaboration over random. In the first scenario, we assign all ℓ_i products initially at random. For our second experiment, we assume that the "big selling products" in any industry are widely and publicly known, such that each store will automatically stock the best $\ell_i/2$ products. Each store completes its product offerings by choosing $\ell_i/2$ more products at random. [4] Once each store has its initial list of products, we simulate collaboration by the PPC protocol and each store adds to its offerings the ℓ_i "best" products that it does not already have. We define the revenue of the resulting store as the sum of the revenues, in the real store, of each product that the store contains.

[4]The decision to choose half the products intelligently and half the products at random is not magical. We could have chosen any other fraction. There needs to be some randomness, however, otherwise the stores would not become aware of any new products and could not increase their offerings as a result.

In order to demonstrate the effectiveness of the collaboration, we calculate the revenue of each store generated in the above manner and compare it to a store of the same size whose first ℓ_i products are identical and whose last ℓ_i products are chosen at random. In the following sections, we will refer to stores built with the help of the protocol as "Collaborative" and the other stores as "Random" or "Guided" depending on the nature of store creation.

We now present the metrics for judging the value of product selection as a result of PPC. We consider two different strategies for determining the best product(s) and judge their merit by comparing their performance to the Optimal Solution (which cannot be calculated from the protocol). The evaluation metrics judge the utility of using the networked view of products versus using the top selling items versus the optimal store. This also helps us assess the overall utility of using the networked view of product space.

Network Value: The Network Value strategy uses the concept of *network value* developed by Richardson and Domingos in [36]. Intended for use in viral marketing scenarios, the network value of a person quantifies the influence that he or she has over the other people in a given social network. The idea is that if a company markets its product to a few influential individuals, those people will tell their friends, and the company gets a large benefit for a relatively small cost. In our case, we are interested in the influence that the purchase of a product has on other products. The network value $\Delta_j Y$ of product i in network Y is defined as: $\Delta_j(Y) = \sum_k w_{kj}\Delta_k(Y)$ where w_{kj} is the influence of product j on product k. Essentially, the network value of product j is the sum, over all items bought with j, of j's influence on that item multiplied by that item's network value. If the total influence on each item is normalized to one, successive iterations of the calculation are guaranteed to converge and we obtain a network value for every product in the network. For our purposes, we define the influence of product j on product k as $P(k|j)$, the observed conditional probability that k is bought given that j is bought.

Products with high network value should be of interest to a store owner. Intuitively, products that are popular (bought frequently) should have higher network values because they have a greater opportunity to influence other products. However, a popular product that is bought with little else should have a smaller network value than a slightly less popular item that is often bought with many other products, since the former exerts far less influence overall. Thus, network value offers an appealing balance between the raw popularity of a product and its effect on the purchase of other products.

It is worth noting that, in order to calculate $P(j|i)$, we need to know how often product i is bought. This can be accomplished without altering the protocol by establishing a special product, say product 0, to which no one is allowed to commit. If c_{0i} is understood to be the total number of times product i is bought, the protocol will produce all the information necessary to calculate network value. The requirement that no one commits to product 0 is necessary in order to prevent the wholesale disclosure of every product that every store sells. From the PPC protocol point of view, we need to ensure that D_i (in step 1) does not contain 0, i.e., $Q_i(0) \neq 0$. This means that q_{m_i} must be non-zero, which is already ensured by the protocol. To carry out the network value strategy, a store owner constructs a network of products based on the output of the protocol. When choosing a new item for the store, the owner chooses the item with the highest network value that is not yet in the store. Intuitively, items with high network value should sell well with a great number of products.

Top Pairs: The Top Pairs strategy does no calculation beyond examining the output of the protocol. When choosing a new product for the store, an owner following the Top Pairs strategy will simply choose the item that has sold the most with another item already in the store.

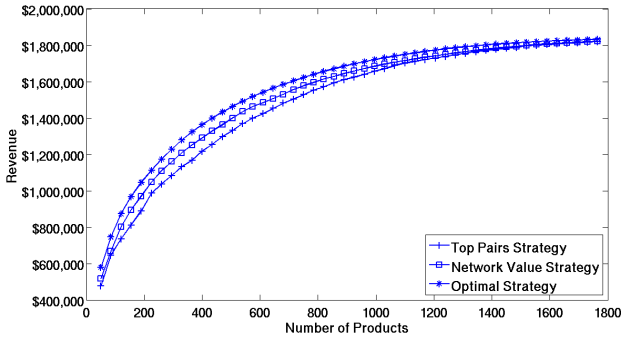


Figure 14: Revenues for different store-constuction strategies as a function of the number of products chosen.

Optimal: The optimal solution chooses, in every instance, the product with the greatest overall sales among all products not yet in the store. The optimal solution cannot actually be computed from the result of the protocol, as it requires knowledge of both the cost of an item and the quantity sold in each transaction, neither of which is revealed. However, the optimal solution provides a useful benchmark against which to compare more practical methods.

Figure 14 shows the revenue of stores constructed from the ground up using each of the three strategies outlined above, as the number of products in the store increases from 50 up to 1,750 products. Predictably, both strategies converge toward the optimal solution as the number of products increases, since there will be fewer differences between the strategies as the number of products in the substore approaches the number of products in the store. The network value strategy outperforms the top pairs strategy at all points and comes within 89.5% of the revenue of the optimal solution even at its lowest point. With 200 products, Network Value achieves 92.7% of the revenue of the optimal strategy and with 500 products, it tops 95%. We submit, then, that constructing stores based on network value is an effective approximation of the optimal construction, given the imperfect information available to protocol participants.

Partitioning Products at Random Table 3 shows the impact of collaboration on stores generated completely at random. Each row of the table represents the average revenue of a series of stores. The "Random" column is the average revenue of a set of stores whose products are chosen entirely at random and the "Collaborative" column shows the revenue for stores of the same size that choose half their products at random and half as a result of collaboration. As we would expect, the value of the purely random stores is incredibly low. The improvement due to collaboration, then, is relatively high as the stores become aware of many better products that they could have chosen. This effect is magnified as the number of participating stores increases, because the stores collectively become increasingly aware of which products are the best to choose.

Partitioning Products with Guidance Table 4 shows the result of the same simulation in the instance where each store starts out with a set of profitable products already on the shelves. We call this "Guided" partitioning as each store is informed about some of the top selling items in the marketplace, and stocks them to start. Each store starts with ℓ products, as in the previous experiment. The main difference being that $\ell/2$ are top-selling products that the store-owner is aware of. Another ℓ products are then added either through collaboration or at random. We then gauge

Table 3: Revenue improvement due to collaboration in completely random stores.

Stores	Products	Revenue		Change
		Random	Collaborative	
4	100	$81,038	$140,290	+179%
8	100	$83,218	$219,124	+275%
16	100	$80,559	$295,585	+384%
4	200	$159,683	$260,670	+166%%
8	200	$163,146	$421,430	+265%
16	200	$161,575	$551,114	+350%
4	400	$334,615	$548,057	+165%
8	400	$327,752	$757,671	+233%
16	400	$328,029	$976,483	+300%

Table 4: Revenue improvement due to collaboration in "guided" stores.

Stores	Products	Revenue		Change
		Guided	Collaborative	
4	100	$387,552	$403,116	+4.08%
8	100	$388,654	$439,324	+13.1%
16	100	$387,336	$487,887	+26.03%
4	200	$607,431	$631,775	+4.05%
8	200	$607,547	$698,818	+15.05%
16	200	$607,185	$763,032	+25.70%
4	400	$889,383	$908,614	+2.16%
8	400	$886,913	$1,027,004	+15.81%
16	400	$886,388	$1,127,075	+27.16%

the impact of collaboration by comparing the overall revenue.

Again, as one would hope, the information gained from increased participation leads to better product choices although the impact is not as dramatic as it was with completely random stores. As the number of participants increases with the number of products in the store held constant, revenue grows monotonically and approaches a limit: the maximum revenue attainable with that strategy. With 200 items, the network value strategy (assuming knowledge of all products in the store) produces a revenue of $998,998.25. The results in table 4 indicate that, by the time 16 stores participate in the protocol, their average revenue is more than 75% of the best possible revenue with the network value strategy even though the stores originally choose many of their products, in this particular simulation, at random.

Table 4 shows the promise that multi-store collaboration holds for helping retailers improve revenue. The information revealed through collaboration can supplement publicly available data (i.e. the original set of top products) and help retailers make profitable business decisions. In a real-world setting, we expect that stores would use our system to make small changes in their product offerings, rather than doubling the size of their store, our protocol will be effective in these situations as well. If one store is missing a single product that is successful in several other stores, the single store will clearly see its omission.

Table 5: Revenue for the top products by network value in each simulated store, by time slice, and also in the combined result of the collaborative protocol. The time slices are aggregated for the year 2005 (training) and the revenues are reported for the year 2006 (testing).

Hours	Top 50	Top 100	Top 200	Top 400
0–6	$500,294	$667,318	$889,357	$1,147,036
6–9	$340,958	$539,771	$717,466	$991,020
9–12	$444,907	$625,299	$793,876	$1,078,385
12–15	$475,822	$682,196	$872,234	$1,130,768
15–18	$509,737	$713,461	$897,924	$1,131,487
18–21	$517,582	$707,881	$900,024	$1,126,043
21–0	$490,878	$731,973	$912,838	$1,177,560
Collaborative	**$539,667**	**$740,965**	**$928,809**	**$1,184,020**

12.2.2 Division by Transactions

The next batch of simulations creates stores by sampling transactions from the larger store by time-slices in a day. That is the 24 hours are divided into different time segments, and each of those time segments aggregated across the year 2005 becomes a store. This ensures a significant overlap in products across stores, albeit differing in the sales volume.

There are subtle differences in distribution throughout the day, but these differences do not necessarily favor any one store. For example, more breakfast products are sold early in the morning than later in the day and candy sells better at night than in the morning, but there is little reason to believe that any one segment of the day or day of the week is closer to the "true" (year-2006) distribution than any other. The separate stores are created as equally as possible. The store is never open from 4:00 AM to 7:00 AM, but we extend the stores out in order to catch any "straggler" transactions that take place sightly outside the official hours.

After building the separate stores, we simulate participation in the PPC protocol. As a result, each store receives the combined purchase frequencies, across all stores, of every pair of items it sells. Using these combined counts, we can calculate new network values for all products and rank them by network value. If the protocol is useful, the combined information should identify profitable products more effectively than the individual stores do. With this in mind, we identified the top 50, top 100, top 200, and top 400 products according to each store's own information and according to the combined output of the protocol. We then calculated the total sales of each of these sets on year 2006 data. By training on 2005 data and testing on 2006 data, we provide evidence that our protocol can help stores isolate products which will be profitable in the future.[5]

Table 5 shows the result of the above experiment. In each case, the collaborative information of all six stores produces better results than the information of any single store. In other words, by participating in the protocol, each store receives a more accurate picture of customers' true purchase behavior than its own data can show. Data from the lower

[5]This evaluation also ensures that the overall salves volume of any one store does not effect the results. By evaluating on all of 2006, we assess instead the ability of a store to predict the future distribution. Any method where sales volume was a factor would unfairly favor the combined results.

revenue stores does not taint the results of the protocol, but rather serves to complete the picture, such that even the most prepared store benefits from contributing its data to the process.

We see, then, that under a reasonable construction scenario, information provided by the PPC protocol is a better indicator of future product profitability than a single store's data in isolation.

13 Conclusion

This paper begins to develop a asystematic approach for the representation of transactional data as a network, inclusing both a general procedure for network-based analysis of market basket data, and privacy-preserving protocol through which such informtion can be shared. First, we construct a network of products and show how, using community detection, one can find interesting relationships among products. More specifically, we state three particular shortcomings in traditional association rules analysis and show how a network view of the product space can mitigate those shortcomings.

First, we take the set of transactions and construct a network in which vertices represent products, and there is an edge between two products if they ever appear in the same transaction. Then, we prune away insignificant edges, determining an appropriate threshold for the minimum number of times that two products must be purchased together before they are connected. We run a community detection algorithm on the resulting network. For each discovered community, we either inspect it directly, if it is small enough, or we construct a spanning tree of the cluster which is then easier to visualize and understand.

We have shown that this framework addresses each of the three issues laid out at the beginning of the paper. Whereas association rules have difficulty representing complex multi-item relationships (*limited scope*), the community detection algorithms isolated a very strong eight-item cluster of eggs and baking items like cake mix and cooking oil. Similarly, while association rules are very sensitive to the choice of support and confidence parameters (*parameter sensitivity*), our framework has only one parameter, the minimum edge weight. Furthermore, this parameter can be determined automatically by continually removing small-weight edges until the modularity of the discovered communities stops increasing.

Finally, association rules suffer a *completeness-granularity tradeoff* meaning that it may be necessary to combine several different brands or instances of the same type of product in order to discover meaningful relationships (sacrificing granularity for the sake of completeness). Our methods, by contrast, were able to isolate a significant relationship between fruit and yogurt even though there are several different types of yogurt available in the store.

Our analysis has shown that, in addition to offering the insights outlined above, the communities produced by our framework will preserve the major association rules within a dataset. As a result, we believe this framework can be applicable to any domain that association rules have found an application in. It gives the practitioner an ability to delve in the product or item space without using a significant number of parameters, and have the ability to visualize and evaluate the influence of a product or item on the other product or items. As part of future work, we are acquiring

additional domain datasets on which to evaluate our approach.

In addition to the general frameork, we develop a privacy-preserving protocol such that multiple stores can share their networks of products, to their mutual benefit, without disclosing any specific information about an individual participant's data.

Our protocol offers a number of benefits over existing work in the area. It is more general than privacy-preserving set union, allowing the transmission of counts along with the individual set elements, and it is more secure than existing approaches for privacy-preserving association rules, offering offering security against malicious, rather than merely semi-honest, adversaries.

Additionally, we empirically demonstrate the effectiveness of joint computation on transaction data. We show that, under our model for store construction, stores see increased revenue when they make decisions based on shared information. More specifically, a store without any knowledge of products outside its own offerings can use data obtained freely from others and, with this data, find products additional products to improve its revenue.

Acknowledgments

We gratefully acknowledge financial support from the NET Institute (www.netinst.org) and the Kauffman Foundation.

References

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

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

[3] C. Borgelt. Apriori implementation. http://www.borgelt.net/apriori.html.

[4] Ulrik Brandes, Daniel Delling, Martin Höfer, Marco Gaertler, Robert Görke, Zoran Nikoloski, and Dorothea Wagner. On Finding Graph Clusterings with Maximum Modularity. In *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07)*, Lecture Notes in Computer Science. Springer, 2007. to appear.

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

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

[7] 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.

[8] I. Damgard and M. Jurik. Efficient protocols based on probabilistic encryption using composite degree residue classes. IACR ePrint Report 2000/008, `http://eprint.iacr.org/2000/008`, 2000.

[9] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 9:P09008, 2005.

[10] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

[11] Luca Donetti and Miguel A Mu noz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(10):P10012, 2004.

[12] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 72(2):027104, 2005.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] M. Girvan and M.E. Newman. Community structure in social and biological networks. In *Proc Natl. Acad. Sci.*, volume 99, pages 7821–7826, 2002.

[20] 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.

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

[22] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, New York, NY, USA, 2003. ACM.

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

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

[25] 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.

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

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

[28] M. E. J. Newman. Modularity and community structure in networks. *PROC.NATL.ACAD.SCI.USA*, 103:8577, 2006.

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

[30] 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.

[31] C. Perlich and S. Rosset. Identifying bundles of product options using mutual information clustering. In *SIAM Data Mining*, 2007.

[32] 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.

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

[34] 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.

[35] Joerg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74:016110, 2006.

[36] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, 2002.

[37] Matt Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Eighth Intl. Conf. on Knowledge Discovery and Data Mining.*, 2002.

[38] Jianhua Ruan and Weixiong Zhang. An efficient spectral algorithm for network community discovery and its applications to biological and social networks. *ICDM 2007. Seventh IEEE International Conference on Data Mining*, pages 643–648, Oct. 2007.

[39] 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.

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

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

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

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