December 2003

Pattern-Oriented Clustering of Web Transactions

Yinghui Yang
University of Pennsylvania

Balaji Padmanabhan
University of Pennsylvania

PATTERN-ORIENTED CLUSTERING OF WEB TRANSACTIONS

Yinghui Yang
The Wharton School
University of Pennsylvania
yiyang@wharton.upenn.edu

Balaji Padmanabhan
The Wharton School
University of Pennsylvania
balaji@wharton.upenn.edu

Abstract

We propose a method for clustering web transaction data based on the idea that patterns generated within a cluster are similar to each other and different from patterns generated from other clusters. To do this, we define the difference between clusters and the similarity of transactions within a cluster using the notion of itemsets. A preliminary experiment on user-centric web browsing data demonstrates that our method is promising in clustering web transaction data and can discover interesting clusters among user transactions on the Web.

Keywords: Data mining, clustering, Web transactions, pattern-oriented clustering

Introduction

As customers interact online at web sites, much data gets collected on their behavior. Firms are increasingly relying on data mining techniques to learn patterns from web transactions in order to understand customer behavior. A specific technique that is particularly useful for this is clustering. Individual web transactions can be in the millions, and clustering groups transactions into a smaller number of groups, making it easier to understand the data. Interest in clustering has increased in recent years due to new applications which include grouping customers and products in massive retail datasets, document clustering, analysis of web usage data, gene expression data analysis and image analysis.

Our paper focuses on clustering customer transaction data, which appears in lots of important applications. Examples for such data are supermarket transaction data, web browsing data etc. It has been shown that clustering transaction data can yield interesting patterns (Heer and Chi 2002) that reveal groupings. Such groupings can be used to understand customer behavior better. Traditional clustering techniques use various "distance measures" between the set of variables in transactions to help cluster the data. The most common distance metric is the Euclidean distance between transactions. In this paper we propose a novel clustering approach based on the idea of "pattern-oriented clustering" that clusters transactions into groups such that "patterns" generated from one group are "similar" to each other but "different" from patterns generated from other groups. This is, fundamentally, a different approach than traditional clustering techniques. In particular, we propose a method for clustering web transactions so that itemsets generated in one cluster are "similar" to each other but "different" from itemsets generated from other clusters.

In order to operationalize this idea, we need to clarify what "patterns" are, and subsequently, what "similar" and "different" mean. In general, patterns could be represented in several ways – they could be a set of co-occurring items, rules that are discovered from the data etc. The choice of representation depends on the specific application. For web transactions, "itemsets" are a natural representation for patterns, since they capture sets of pages that are visited together in a given user's session and also other web browsing features (e.g. most visited site in the session). Hence we use itemsets to represent web browsing behavioral patterns, and use that to guide the clustering process. We define the difference between clusters and the similarity of the transactions within a cluster using the support of frequent itemsets, and cluster the transactions to maximize the difference and the similarity. Note that the idea of pattern-based clustering is broader than just clustering based on itemsets, which is one specific representation that is useful for web transaction data.

Because of our focus on clustering different transactions based on patterns, there are several other potential applications for our method. For example, people can share PCs, shopper cards, cell phones and credit cards. In these applications, just as different people have different physical characteristics such as fingerprints or iris patterns, it has been conjectured that different people have different *behavioral* patterns. If this is indeed the case, clustering techniques, such as the one presented here, may be valuable in separating the transactions generated by different people. By separating different behavioral patterns, it may be possible to build better personalization engines, fraud detection systems and, in general, models.

In recent studies, several new clustering algorithms using frequent items or itemsets are proposed. Itemsets based approaches are promising, because they provide a natural way of reducing the large dimensionality of the vector space. Han et al. (1997) proposes a new methodology for clustering related transactions using association rules. Wang et al. (1999) introduces a new clustering criterion suggesting that there should be many large items within a cluster and little overlapping of such items across clusters. This criterion is then used to search for a good clustering solution. Compared to the few papers using items or itemsets for clustering transaction data, our method takes a new perspective of associating itemsets with behavioral patterns, and uses that concept to guide the clustering process. Wang et al. (2002) uses pattern similarity to cluster. They consider two objects similar if they exhibit a coherent pattern on a subset of dimensions. The definition of a pattern is defined as the correlation between attributes of objects to be clustered. This specific definition of pattern makes it more suitable for numerical data. Guha et al. (1999) propose a technique for clustering categorical data, but this approach is not based on pattern-oriented clustering.

Pattern-Oriented Clustering: A New Criteria

Consider a collection of transactions to be clustered { $T_1, T_2, \dots, T_n$ }. Each transaction T_i contains a subset of a list of candidate items { $i_1, i_2, \dots, i_m$ }. A clustering C is a partition { $C_1, C_2, \dots, C_k$ } of { $T_1, T_2, \dots, T_n$ }. Each C_i is called a cluster. The goal of our method is to maximize the difference between clusters and the similarity of transactions within clusters.

$$Maximize : M(C_1, C_2, ..., C_k) = Difference(C_1, C_2, ..., C_k) + \sum_{i=1}^{k} Similarity(C_i)$$

We use itemsets as the representation of the behavioral patterns for web transactions to capture the similarity and difference. We consider itemsets instead of just items, because items are not sufficient to represent behavioral patterns. For example, consider the following transactions in Table 1:

Table 1

	First Page	Last Page
T_1	CNN	P1
T_2	CNN	P2
T_3	P3	MSN
T_4	P4	MSN
T_5	CNN	MSN
T_6	CNN	MSN

If we only consider items, we might get the following clusters { T_1, T_2, T_5, T_6 }(all start from CNN) and { T_3, T_4, T_5, T_6 }(all end with MSN). But if we allow itemsets, we may end up with the following clusters { T_1, T_2 }(all start from CNN, and not end with MSN) , { T_3, T_4 }(all end with MSN, and not start from CNN) and { T_5, T_6 }(all start from CNN, and end with MSN). Obviously, the second clustering in which itemsets are considered is more powerful and meaningful.

The *support* of an itemset is the number of transactions that contain all the items in the itemset. *Frequent* itemsets are itemsets which have support that exceeds some user-specified minimum support threshold.

Let *FIS* indicate the set of all the frequent itemsets (we call it the candidate itemsets) based on the entire transaction data (before clustering). After clustering, we calculate the support of each itemset in *FIS* within each cluster. Note that a candidate itemset is frequent in the entire set of transactions before clustering, but not necessarily frequent within each cluster. Specifically, *FIS* = {is_1 , is_2 , ... , is_p } where is_i is an itemset in *FIS* . See Table 2 for the support matrix where s_{ij} is the support of itemset *i* in cluster *j*.

Table 2

	C_1	C_2	…	C_k
is_1	s_{11}	s_{12}	…	s_{1k}
is_2	s_{21}	s_{22}	…	s_{2k}
…	…	…	…	…
is_p	s_{p1}	s_{p2}	…	s_{pk}

Now, we define the difference between two clusters and the similarity of the transactions within a cluster using the support of the itemsets.

Definition 1 (Difference between two clusters)

$$Difference\ \ (C_i, C_j) = \sum_{a=1}^{p} \frac{\left| \frac{S_{ai}}{|C_i|} - \frac{S_{aj}}{|C_j|} \right|}{\frac{1}{2} \times \left(\frac{S_{ai}}{|C_i|} + \frac{S_{aj}}{|C_j|} \right)}$$

$|C_i|$ is the number of transactions in cluster C_i and p is the number of itemsets in *FIS*. This is a good measure since for every itemset, it calculates the difference between the pattern in each cluster by computing the distance between the support values (adjusted by the actual support values). For example, if the itemset/pattern {cnn.com, slashdot.org} occurs in 1% of the transactions in cluster-1 while it occurs 11% of time in cluster-2, the difference is 1.67. If it occurs 80% and 90% of time respectively the difference is 0.12. We argue that this is desirable since it is based on the relative difference and not the absolute values (in the example above, the absolute differences are both 10%).

Definition 2 (Intra-cluster similarity):

Here, our goal is to define a metric that captures what it means for transactions in one cluster to be "similar" to each other. An itemset is considered representative of a pattern if it is frequent and hence we define similarity based on the number of frequent itemsets within a cluster.

Similarity $S(C_i)$ = the number of frequent itemsets in cluster C_i

Algorithm for Clustering Using Frequent Itemsets

The ideal algorithm will be the one which can maximize M (defined in previous section). However, if there are n transactions and two clusters that we are interested in learning, the number of possible clustering is 2^n. Hence we need heuristic algorithms to search for a good solution.

After we get *FIS*, we convert the initial transactions into the format in Table 3, where the rows represent transactions to be clustered and the columns represents itemsets. A "1" in a cell indicates that a certain transaction contains a certain itemset.

Table 3

	is_1	is_2	…	is_p
T_1	1	0	…	1
T_2	1	1	…	0
…	…	…	…	…
T_n	0	1	…	1

We use $T'=\{ T'_1, T'_2, \dots, T'_n \}$ to represent this new transaction set. Now, instead of clustering the original transactions, we convert the problem into clustering these binary vectors and present a divisive hierarchical algorithm. The entire transaction set is first divided into two clusters. Each cluster is further divided into two clusters if it has more than a predefined number of transactions. This process is repeated until no more cluster is big enough to be divided further. If a cluster's size is bigger than the threshold but its quality is very good (contains very similar transactions), we can also consider stopping dividing that cluster.

In order to generate balanced clusters, we introduce another component to *M.* And also because each division creates two clusters, the revised *M* is as follows:

$$M(C_1, C_2) = K_1 \times D(C_1, C_2) + K_2 \times S(C_1) + K_3 \times S(C_2) - K_4 \times |N_1 - N_2|,$$

where

K_1, K_2, K_3, K_4 are user-specified weights (can be decided based on different applications),
$D(C_1, C_2)$ represents the inter-cluster difference,
$S(C_1)$ and $S(C_2)$ are the intra-cluster similarity for cluster-1 and cluster-2 respectively, and
N_1 and N_2 are the number of transactions in cluster-1 and cluster-2 respectively.

The candidate itemsets come from association rule discovery algorithms such as Apriori (Agrawal et al. 1995).

Figure 1 describes the algorithm YACA.

```
Input: $C$ = { $C_0$ }
        $C_0$ = { $T'_1, T'_2, \dots, T'_n$ } - the transactions to be clustered (see Table 3)
Output: Clusters $C = \{ C_1, C_2, \dots, C_f \}$

Repeat {
Choose any $X$ = $\{T_1, T_2, \dots, Ts\}$ from $C$ such that the stopping condition is not satisfied for $X$
$C_1$ = {}, $C_2$ = {}
Assign $T_1$ in $X$ to $C_1$
Assign $T_m$ to $C_2$ | $T_m$ is 'most different' from $T_1$
for $i$=1 to $s$ {
        Allocate $T_i$ to $C_1$ or $C_2$ to maximize $M$
        }
$C$ = ($C$ - $X$) $\cup C_1 \cup C_2$

} While the stopping condition is not satisfied for every $X$ in $C$
```

Figure 1. Algorithm YACA

For each division, the first transaction and the 'most different' transaction (in terms of Euclidean distance) from the first transaction are chosen to form two initial clusters, C_1 and C_2. Then, every other transaction is either assigned to C_1 or C_2 whichever maximizes *M*. The two starting transactions are not necessarily the most different ones among all the transactions. They are used just to speed up the process.

Experiment: Clustering User-Centric Web Transactions

User-centric web transaction data is data collected at the user level and thus captures the entire history of web surfing behavior for each user. We use user-centric data provided by a market data vendor. First, we group individual hits into sessions. A common rule of thumb to determine how to aggregate hits into sessions is to observe the time associated with each hit and group consecutive hits that are within 30 minutes of each other into a session. Then we create features to describe the user's behavior for each session. There are totally 46 features (e.g. most visited site in the session, the first site in the session) constructed for each session. We create an item for each value of the categorical features. We categorize the continuous variables according to their value distribution (uniformly binned), and create an item for each bin.

After the data preprocessing, we pick out the transactions belonging to a certain number of users and create a session (transaction) by item matrix with binary values (see Table 4).

Table 4

	i_1	i_2	…	i_p
T_1	1	0	…	1
T_2	1	1	…	0
…	…	…	…	…
T_n	0	1	…	1

From this matrix, we use Apriori to discovery the candidate itemsets and convert the above transactions to the format described in Table 3 and use the converted transactions as the input to our clustering algorithm.

It's a hard problem to evaluate unsupervised learning. As a proxy, we propose to combine transactions from multiple users repeatedly and test if our clustering method can separate transactions associated with each user. On one data set containing approximately 6000 transactions from two random users, 95% of the clusters generated by YACA are at least 95% pure (at least 95% of the transactions in a cluster belong to one user), while only 60% of the clusters generated by k-means on the original transactions (as in Table 4) are at least 95% pure. We plan to do more experiments on more data sets containing transactions from various number of users. More importantly, we plan to extend this work by using the clusters discovered to build better customer behavior models.

References

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A. I., "Fast Discovery of Association Rules", Advances in Knowledge Discovery and Data Mining, Chapter 12, AAAI/MIT Press, 1995

Guha, S., Rastogi, R., Shim K., "ROCK: A Clustering Algorithm for Categorical Attributes", Proc. 15th International Conference on IEEE Data Engineering, Sydney, Australia, 1999

Han, E., Karypis, G., Kumar, V. and Mobasher, B., "Clustering based on association rule hypergraphs", Proc. SIGMOD'97 Workshop on Research Issues in Data Mining and Knowledge Discovery. 1997, ACM.

Heer, J., Chi, E.H., "Separating the Swarm: Categorization Methods for User Sessions on the Web". *CHI 2002 Conference Proceedings*. ACM, 2002.

Wang, H., Yang, J., Wang, W., and Yu, P.S., "Clustering by Pattern Similarity in Large Data Sets", Proc. ACM SIGMOD Conference, Madison, WI, June 2002.

Wang, K., Xu, C. and Liu, B. "Clustering Transactions Using Large Items", Proc. 8th Int. Conf. on Information and Knowledge Management (ACM CIKM'99), Kansas City, November, 1999.