# Predicting Customer Shopping Lists from Point-of-Sale Purchase Data

Chad Cumby, Andrew Fano, Rayid Ghani, Marko Krema
Accenture Technology Labs
161 N Clark St
Chicago, USA
{chad.m.cumby,andrew.e.fano,rayid.ghani,marko.krema}@accenture.com

## ABSTRACT

This paper describes a prototype that predicts the shopping lists for customers in a retail store. The shopping list prediction is one aspect of a larger system we have developed for retailers to provide individual and personalized interactions with customers as they navigate through the retail store. Instead of using traditional personalization approaches, such as clustering or segmentation, we learn separate classifiers for each customer from historical transactional data. This allows us to make very fine-grained and accurate predictions about what items a particular individual customer will buy on a given shopping trip.

We formally frame the shopping list prediction as a classification problem, describe the algorithms and methodology behind our system, its impact on the business case in which we frame it, and explore some of the properties of the data source that make it an interesting testbed for KDD algorithms. Our results show that we can predict a shopper's shopping list with high levels of accuracy, precision, and recall. We believe that this work impacts both the data mining and the retail business community. The formulation of shopping list prediction as a machine learning problem results in algorithms that should be useful beyond retail shopping list prediction. For retailers, the result is not only a practical system that increases revenues by up to 11%, but also enhances customer experience and loyalty by giving them the tools to individually interact with customers and anticipate their needs.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

## General Terms

Design, Experimentation

## Keywords

Applications, Machine learning, Classification, POS Data

## 1. INTRODUCTION

Retailers have been collecting large quantities of point-of-sale data in many different industries. One area that has been particularly active in terms of collecting this type of data is grocery retailing. Loyalty card programs at many grocery chains have resulted in the capture of millions of transactions and purchases directly associated with the customers making them.

Despite this wealth of data, the perception in the grocery industry is that this data has been of little use. The data collection systems have been place for several years but systems to make sense of this data and create actionable results have not been very successful. This is not to say that there has been no work attempted on retail transaction data. Research in mining association rules [3] has led to methods to optimize product assortments within a store by mining frequent itemsets from basket data [6]. Customer segmentation has been used with basket analysis in the direct marketing industry for many years to determine which customers to send mailers to. Additionally, a line of research based on marketing techniques developed by Ehrenberg [9] seeks to use a purchase incidence model with anonymous data in a collaborative filtering setting [10].

Traditionally, most of the data mining work using retail transaction data has focused on approaches that use clustering or segmentation strategies. Each customer is "profiled" based on other "similar" customers and placed in one (or more ) clusters. This is usually done to overcome the data sparseness problem and results in systems that are able to overcome the variance in the shopping behaviors of individual customers, while losing precision on any one customer. We believe that given the massive amounts of data being captured, and the relative high shopping frequency of a grocery store customer, we can develop individual consumer models that are based on only a single customer's historical data. Our hypothesis is that by utilizing the detailed transaction records to build separate classifiers for every unique customer, we can improve on the performance of clustering and segmentation approaches.

A major reason that individually targeted applications have not been more prominent in retail data mining research is that in the past there has been no individual channel to the customer for brick & mortar retailers. Direct mail is coarse-grained and not very effective as it requires the attention of customers at times when they are not shopping and may not be actively thinking about what they need. Coupon based initiatives given at checkout-time are seen as irrelevant as they can only be delivered after the point of sale. However with the advent of PDA's and shopping cart mounted displays such as the model Symbol Technologies is piloting with a New England grocer [1], retailers are in a position now to deliver personalized information to each customer at several points in the store.

In fact, a few systems of this sort have been developed. For example the IBM Easi-Order system [4] and a system developed at Georgia Tech [13] use PDA's to display personalized grocery information to each shopper before, and during their shopping trips. In the first system, a list is first developed on the PDA, then sent to the store to be compiled and picked up. In the second, the PDA was used as an aide during the shopping trip to show locations and information on items in a list. In each, the shopping list was emphasized as the essential artifact of a grocery trip, enabling all other interactions. Both also stated as a design goal that it should be possible to compile or augment a shopping list per customer based on previous purchase history. The *1:1Pro* system described in [2] was designed to produce individual profiles of customer behavior in the form of sets of association rules for each customer which could then be restricted by a human expert. Theoretically these profiles could then be used to develop personalized promotions and predict certain purchases. However, nowhere has there been a thorough experimental attempt to predict and evaluate customer grocery shopping lists from transactional data with a large set of customers.

We present our shopping list predictor as an integral part of a larger system targeted at presenting personalized information to a customer in a retail setting. Our *Shopping Assistant* system uses the list as a starting point for interacting with the customer. Instead of promoting random products, we uses the items on the predicted list to deliver personalized promotions.

The work presented in this paper uses machine learning techniques to predict and populate a shopping list for each customer, on the day they visit the store. We use two years of customer purchase data from a major grocery chain and train classifiers for each customer in our data set. When a particular customer enters the store, they identify themselves (through a loyalty card, for example) to a shopping cart-mounted display device (see Fig. 1). The predicted shopping list is then presented to the customer on the display as the "suggested" list. Since the number of items an average person buys in a grocery trip can be large, we only show the entire list at the beginning of the trip. From that point on, the device can detect which aisle the customer is in and only show them items that are in their list in that particular aisle. While the main rationale behind this task remains to provide the convenience and personalized service that a customer should expect in return for their personal information, we view it in the wider context of differentiating and improving business for the retailer. By suggesting a realistic shopping list for a customer, we remind them of purchases that might otherwise be forgotten. These suggestions translate into recovered revenues for the store that might otherwise be transferred to a competitor, or foregone as the customer goes without the item until the next trip.

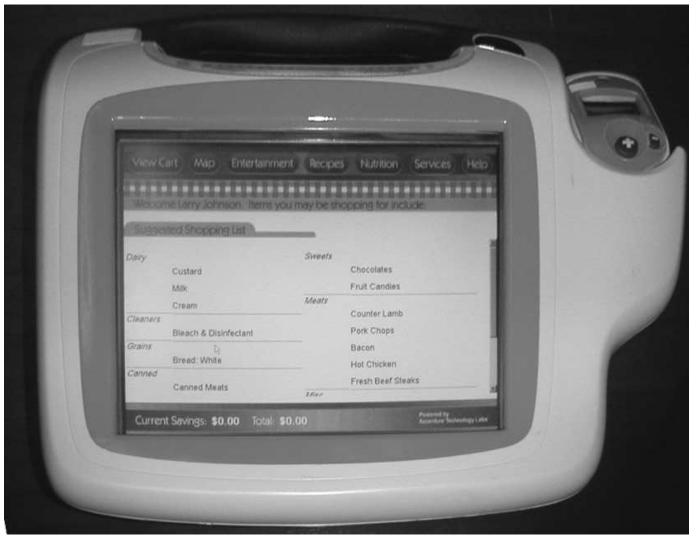

**Figure 1: Cart-mounted display device used in our prototype.**

The remainder of the paper is organized as follows: Sec. 2 describes the way in which we frame shopping list prediction as a classification problem, along with the criteria for success and organization of the datasource. Sec. 3 describes the results of the classification experiments using the loyalty card data from a grocery store chain. Sec. 4 discusses the implications the experimental results in the context of our larger system and possible directions for future research on this topic. In Sec. 5 we conclude.

## 2. PREDICTION METHODOLOGY

This section explains the methodology behind our shopping list predictor, as well as the evaluation criteria we use to judge its success. In order to define the problem of grocery shopping list prediction in a rigorous way, we first introduce some terms and explain their meanings.

We define a set $C$ of customers, a set $T$ of transactions made by those customers, and a fixed set $P$ of product categories bought by these customers equivalent to those normally used on shopping lists. Within $T$ and $P$ we define for each $c \in C$ sets $T_c \subseteq T$ and $P_c \subseteq P$, consisting of the transactions of each customer $c$ and the product categories bought by customer $c$ respectively. For each transaction $t \in T_c$ our task then becomes to output a vector $y \in \{0,1\}^{|P_c|}$ where $y_i = 1$ if for a given order of all categories in $P_c$, customer $c$ bought $p_i \in P_c$ in transaction $t$, and where $y_i = 0$ if customer $c$ did not buy $p_i$. We can then formulate the overall problem as $|P_c|$ binary classification problems for each customer and derive a separate classifier for each. We experimented with many different types of methods.

### 2.1 Baseline Methods

We present some baseline methods to predict customer shopping lists that we can hopefully improve on using data mining techniques.

### 2.1.1 *Random baseline*

The most basic baseline we use is random guessing. In this scheme for each transaction of a given customer, we output a prediction vector $y'$ where each $y'_i$ is equal to 0 or 1 with an equal probability. This results in a method where every category that the customer has purchased before has 50% chance of being included in the shopping list.

### 2.1.2 *Same as Last Trip baseline*

The next baseline method just produces a shopping list that consists of products bought in their previous shopping trip. To define it more formally, let us impose an ordering on the set $T_c$ for each customer $c$ corresponding to the temporal sequence of each transaction. Then for each transaction $t_k$, we output a prediction vector $y'$ equal to the purchase vector seen for transaction $t_{k-1}$. Let this method be called the *same as last trip* predictor.

### 2.1.3 *Top N baseline*

Finally we describe an approach we call the *top n* method, where we aggregate all the transactions of the given customer, and select the top $n$ products from their history as their current shopping list, ranked by the quantity/frequency of purchase. We define a new ordering on the set $P_c$ for each customer, corresponding to the frequency with which each category is purchased within $T_c$. Specifically for each $p_i \in P_c$ let $freq(p_i) = \frac{\sum_{j=1}^{|T_c|} y_i^j}{|T_c|}$. Then the top $n$ method, for each transaction $t$, outputs a vector $y'$ for which the values corresponding to the top $n$ categories in $P_c$ as ordered by $freq$ are valued 1, with all else valued 0. A variation on this method would be to only use the past $m$ transactions to create the Top N list which might account for some of the temporal changes a customer might exhibit.

## 2.2 Machine Learning Methods

The second group of approaches we experiment with are all machine learning classification methods. As mentioned before, the problem of predicting the overall assortment of categories purchased $y$ can be broken down into $|P_c|$ individual binary classifications. Each class can be thought of as a customer and product category pair. If our data set consists of $|C|$ customers and an average of $q$ categories bought by each customer, we construct $|C| \times q$ classes (and as many binary classifiers). For each of these classes $y_i$, a classifier is trained in the supervised learning paradigm to predict whether that category will be bought by that customer in that particular transaction. Here we present a series of examples of the form $(x, y_i)$, where $x$ is a vector in $\Re^n$ for some $n$, encoding features of a transaction $t$, with $y_i \in \{0, 1\}$ representing the label for each example (*i.e.* whether the category corresponding to $y_i$ was bought or not).

We experimented with two kinds of machine learning methods to perform this task. First we trained decision trees (specifically using C4.5 [14]) to predict each class label. Next we tried several linear methods (Perceptron[15], Winnow[11], and Naive Bayes) to learn each class . These linear methods offer several advantages in a real-world setting, most notably the quick evaluation of generated hypotheses and their ability to be trained in an on-line fashion.

In each case, a feature extraction step preceded the learning phase. Information about each transaction $t$ is encoded as a vector in $\Re^n$. For each transaction, we include properties of the current visit to the store, as well as information about the local history before that date in terms of data about the previous 4 transactions. The assumption here is that examples and their labels are not independent, and that we can model this dependence implicitly by including information about the previous visits. This tactic is similar to methods in Natural Language Processing (NLP) for tasks such as part-of-speech tagging, where tags of preceding words are used as features to predict the current tag [16].

The features we include in example $(x^j, y_i^j)$ about transaction $t^j$ are:

1. Number of days at $t^j$ since product category was $p_i$ bought by that customer. We call this the *replenishment interval* at $t^j$.
2. Frequency of *interval* at $t^j$. For each category $p_i$ we build a frequency histogram per customer for the *interval* at purchase binned into several ranges (*eg* 3-5 days, 7-9 days). This histogram is normalized by the total number of times items in that category were purchased.
3. The interval range that the current purchase falls into. These are the same ranges as mentioned above.
4. Day of the week of the current trip.
5. Time of the day for the current trip broken down into six four-hour blocks.
6. Month of the year for the current trip.
7. Quarter of the year for the current trip.

We also include all of the above attributes for the previous 4 transactions, $t^{j-1}, t^{j-2}, t^{j-3}, t^{j-4}$ in $(x^j, y_i^j)$. Additionally we include four additional features with respect to each transaction in the local history. These are:

1. Whether category $p^i$ was bought in this transaction.
2. The total amount spent in this transaction.
3. The total number of items bought in this transaction.
4. The total discount received in this transaction.

Note that the previous 4 features are only used for the local history of the current transaction and not for the current transaction itself. Since we are predicting the products bought for the current transaction when the customer enters the store, we obviously do not have access to these features.

In the case of the decision tree learners, the above is the entire set of features used. For the set of linear classification methods we utilized, it is often difficult to learn a linear separator function using a relatively low-dimensional feature space such as we have constructed. By combining basic features, effectively increasing the dimensionality of each example vector $x$, we increase the chance of learning a linear function that separates all the positive and negative examples presented. Once again this tactic is similar to those used to learn classifiers in NLP contexts where combinations of words such as bi-grams and tri-grams are used as features in addition to the basic words.

Therefore for the linear methods, several combinations of the basic features listed above were added to each example to improve learnability. For each numbered feature type above, we combine it with those of the same type in the customer's previous four transactions (local history). For example, feature 4 ( day of the week for the current transaction) is combined with feature 4 of the previous transaction to produce a new feature. For the set-valued feature types above such as 4, boolean features are instantiated for each value (*eg* one feature per day). The combinations of these features used are simple boolean conjunctions. For the feature types corresponding to continuous valued attributes such as 2, we create a single real valued feature. To create combinations of these features we use a non-linear transformation.

## 2.3 Hybrid Methods

The final set of methods explored in our experiments took the form of several *hybrid* methods. As we mention below, due to the large number of output classes we are trying to predict over all customers, we would like to evaluate the performance of our prediction strategies in aggregate with a single measure. However, as we treat each class as independent of each other for a given transaction (a simplifying albeit untrue assumption), different classification methods can be used for different classes. This is our hybrid approach. In the experiments, we combined the *top n* baseline classifier with the various learned classifiers in the following fashion. If the *top n* predictor (for given $n$) is positive for a given class, then we predict positive, otherwise we predict according to the output of a given learned predictor.

## 2.4 Evaluation

The problem of predicting grocery shopping lists is an interesting learning problem because of the sheer number of classes that must be predicted. Abstracting from the product level (around 60,000 products) to the level of relatively specific categories useful for grocery lists reduces this number to some degree. However for real world datasets such as the one we explore in Sec. 3, this number could be from fifty to a hundred classes per customer, with tens of thousands of regular customers per store, resulting in millions of classification categories and classifiers.

In general the metrics we use to judge the performance of our list predictors per class are the standard *recall*, *precision*, *accuracy* and *f-measure* quantities. For a set of test examples, *recall* is defined as the number of true positive predictions over the number of positive examples; *precision* is the number of true positive predictions over the total number of positive predictions; *accuracy* is the number of correct predictions over the total number of examples; and *f-measure* is the harmonic mean of *recall* and *precision*: $\frac{2 \cdot recall \cdot precision}{recall + precision}$.

In obtaining an overall measure of performance by which we measure our success in predicting shopping lists for large groups of customers, there are many considerations to take into account. Typically in a learning scenario with a large number of output classes, the above quantities can be aggregated in several ways. *Microaveraged* results are obtained by aggregating the test examples from all classes together and evaluating each metric over the entire set. The alternative is to *macroaverage* the results, in which case we evaluate each metric over each class separately, and then average the results over all classes. The first strategy tends to produce higher results than the second. When the number of classes is large and very unbalanced, the microaveraged results are implicitly dominated by the classes with a large number of examples, while the macroaveraged results are dominated by the smaller classes. Macroaveraging is intuitively more attractive for our purposes as it gives us an idea of how we are performing for the majority of customers rather than just those with a large number of transactions.

However, the transactional nature of the purchase datasource gives us additional methods to aggregate our results. One option is to aggregate all examples associated with a single customer, obtain results for the above metrics for each set, and average them. This approach lets us know how we are performing for the average *customer*. Although these aggregate sets are still unbalanced, given some customers shop more than others, the average results for this approach are generally between micro and macro-averaging. We call this *customer averaging*. The last type of aggregating we can do is on the transaction level. Here we aggregate all the examples from each transaction, calculate each metric, and average the results over all transactions. We call this method *transaction averaging*. This averaging technique is perhaps most attractive of all in light of its ability to gauge how many categories per trip predicted are bought, and how many bought per trip are predicted. However since it breaks up example sets within classes, it is difficult to compare this approach with the other aggregation techniques.

## 3. EXPERIMENTS & RESULTS

In this section we describe the results of our initial experiments predicting customer grocery shopping lists, using data for several thousand customers. The dataset used contains transaction based purchase data for over 150,000 customers from a grocery store collected over two years. From this overall set, 22,000 customers shopped between 20 and 300 times, which was judged to be the legitimate population for whom to predict lists. This population was sampled to produce a dataset of 2200 customers with 146,000 associated transactions. Since the number of transactions for each customer follow a power law, uniform random sampling to select 10% of the customers would result in a sample skewed towards customers with a small number of transactions. In order to get a representative sample, we first split the population into deciles along three attributes: total amount spent, total number of transactions, and $\frac{\#transactions}{amountspent}$. For each set of deciles, 10% of the data was selected with uniform probability from each decile. The 10% samples obtained for each attribute were found to be statistically similar to the other two (details omitted), and the final sample used was taken from total amount spent. The motivation behind working with a somewhat smaller working sampled set in our case is purely computational. As we are building classifiers for each <customer, category> individually, the running time to train and evaluate predictors scales linearly with the number of customers. By performing the type of stratified sampling mentioned above, we preserve the wide variation in the quality of training data available to each individual predictor, *i.e.* the number of examples and different purchasing behavior.

The transactional information present in the data includes the attributes described in the previous section and lists of products purchased in each transaction. Products are arranged in a hierarchy of categories. At a fairly specific level of this hierarchy, categories resemble grocery shopping list level items. Examples of these categories include: **cheddar cheese, dog food, sugar, laundry detergents, red**

**wine, heavy cream, fat-free milk, tomatoes, etc**. 551 categories are represented in the dataset forming the set $P$ as defined in Sec. 2. Customers within our sample bought 156 distinct categories on average (w/ standard deviation of 59). Of these categories, we restrict the set $P_c$ for each customer to include only the categories bought on greater or equal to 10% of their trips. This brings the average size of $P_c$ for a given $c$ to 48 with standard deviation of 27.59.

For each transaction for the customers in the sample, examples are constructed as detailed in Sec. 2. The datasets for each class[1] ranged from 4 to 240 examples. For each class in the resulting dataset, the example sets are split into a training set composed of the first 80% of examples in temporal order, and a test set composed of the last 20%. We run the baseline methods only on the test sets for consistency in evaluation. For the *top-n* methods we chose a cutoff of 10 categories. For the decision tree classifier, C4.5 was used with 25% pruning and default parameterization. For the linear classification methods, the SNoW learning system was used [7]. SNoW is a general classification system incorporating several linear classifiers in a unified framework. In the experiments shown, classifiers were trained with 2 runs over each training set.

Results are shown in Table 1 for each approach, broken down by the *transaction* and *customer* averaging methods mentioned in the previous section. Fig.2 (top) shows two graphs with the performance of the top-$n$ method for different values of $n$. For the linear classification methods, the activation values output by the classifier can be normalized to produce a confidence score for each class. We can then choose a different threshold than the one used in training to test the classifier's performance. In Fig.2 (bottom), we show the performance of the Winnow and Perceptron classifiers at different confidence thresholds. Activations are normalized to confidence values between -1 and +1, with the original training threshold mapped to 0.

## 3.1 Fixing Noisy Labels

As mentioned in Sec. 1, a major motivation for predicting shopping lists stems from the goal of reclaiming forgotten purchases. Of course the data collected does not include information on the instances in which categories are forgotten. A priori we would not like to make any assumptions about the instances in which forgetting has occurred. This artifact produces a data set that has noisy labels: examples where a customer "should" have bought a particular product, but forgot to do so, shows up as a negative example in our data. This not only creates noise in the training set (which can be overcome to some extent by robust learning algorithms), but also reduces the reported accuracy of our results. Examples where our system predicts an item is on the list are judged to be incorrect if the customer didn't buy that item (even if they forgot it). However, we would hope that the algorithms we examine should be somewhat robust to label noise as long as they are not overfitting the data. In order to estimate this robustness and determine the value of our suggestions via reclaiming forgotten purchases, we make some assumptions about the distribution of these instances and correct noisy label values in the test data. By training on the noisy data and then evaluating on the corrected test data, we hope to see the number of true positive predictions go up without a serious increase in false negatives.

[1]<customer, product category> pair

| | Recall | Prec | F-Measure | Accuracy |
|---|---|---|---|---|
| random | .20 | .23 | .20 | .63 |
| sameas | .22 | .29 | .25 | .66 |
| top-10 | .37 | .36 | .37 | .65 |
| Perceptron | .42 | .31 | .36 | .61 |
| Winnow | .16 | .40 | .23 | .75 |
| C4.5 | .24 | .42 | .32 | .73 |
| Hybrid-Per | .60 | .32 | .42 | .54 |
| Hybrid-Win | .43 | .41 | .42 | .65 |
| Hybrid-C4.5 | .46 | .38 | .42 | .62 |

**Table 2: Transaction averaged results from corrected label data.**

The manner in which we estimate noisy labels in the test data to correct is described as follows. First, for each class $p \in P_c$ for a given customer, we find the mean $\mu$ and standard deviation $\sigma$ of the replenishment interval $i$.[2] Next, we identify examples for which $i \geq \mu + c \cdot \sigma$ for different constants $c$. For each of these examples that have negative labels, we determine whether any example within a window of $k$ following transactions is positive. We estimate each of these examples to be an instance of forgetting, with noisy negative labels.

To evaluate the robustness of our predictors to this noise, we flip each noisy (forgotten purchase) negative label to be positive, and re-evaluate each classification method on the modified test data. We show the results of this evaluation below in Table 2 for $c = 1$.[3] Here we show only the transaction averaged results.

## 4. DISCUSSION & FUTURE WORK

In this section we discuss the results of the experimental evaluation and their implications for larger issues within machine learning and knowledge discovery.

Many of the results seen in Sec. 3 are promising in the context of predicting shopping lists for a large number of grocery customers. In terms of providing useful suggestions, we would like to obtain results that cover most of the items in a customer's potential shopping list (high recall) while not overloading the customer with a long list of non-relevant items (precision). As we see in the previous section, it is difficult to accurately predict over 50% of the bought categories with a reasonable level of precision. Only the hybrid method combining the *top 10* classifier with the Perceptron based classifier achieves this high level of recall.

In general each hybrid method performs much better than all the other methods. Each obtains a significantly higher level of recall than its individual component classifiers, with comparable levels of precision. We hypothesize the following scenario to explain this phenomenon.

Due to the wildly imbalanced training set sizes across classes both within and without customer groupings, many classes may contain very few positive examples. The baseline *top-10* classifier gives us a basic level of recall across all classes regardless of the training set size, while the learned classifiers would very rarely produce true positives for these

[2]Note that these moments exist without specifying a distribution over the replenishment interval.

[3]This choice of cutoff equates to a customer forgetting 11% of the product categories that they would otherwise buy on an average trip.

| | Recall | Prec | F-Measure | Accuracy |
|---|---|---|---|---|
| random | .19 | .20 | .19 | .65 |
| sameas | .26 | .26 | .26 | .70 |
| top-10 | .37 | .35 | .36 | .59 |
| Perceptron | .38 | .26 | .31 | .65 |
| Winnow | .17 | .36 | .23 | .79 |
| C4.5 | .22 | .34 | .24 | .77 |
| Hybrid-Per | .59 | .28 | .38 | .53 |
| Hybrid-Win | .43 | .36 | .39 | .65 |
| Hybrid-C4.5 | .46 | .35 | .40 | .62 |

| | Recall | Prec | F-Measure | Accuracy |
|---|---|---|---|---|
| random | .20 | .19 | .20 | .65 |
| sameas | .25 | .29 | .27 | .70 |
| top-10 | .41 | .33 | .37 | .65 |
| Perceptron | .40 | .27 | .32 | .66 |
| Winnow | .17 | .38 | .24 | .79 |
| C4.5 | .25 | .28 | .26 | .70 |
| Hybrid-Per | .60 | .27 | .37 | .55 |
| Hybrid-Win | .44 | .32 | .37 | .64 |
| Hybrid-C4.5 | .48 | .34 | .40 | .62 |

**Table 1: Transaction (left) & customer (right) averaged results.**

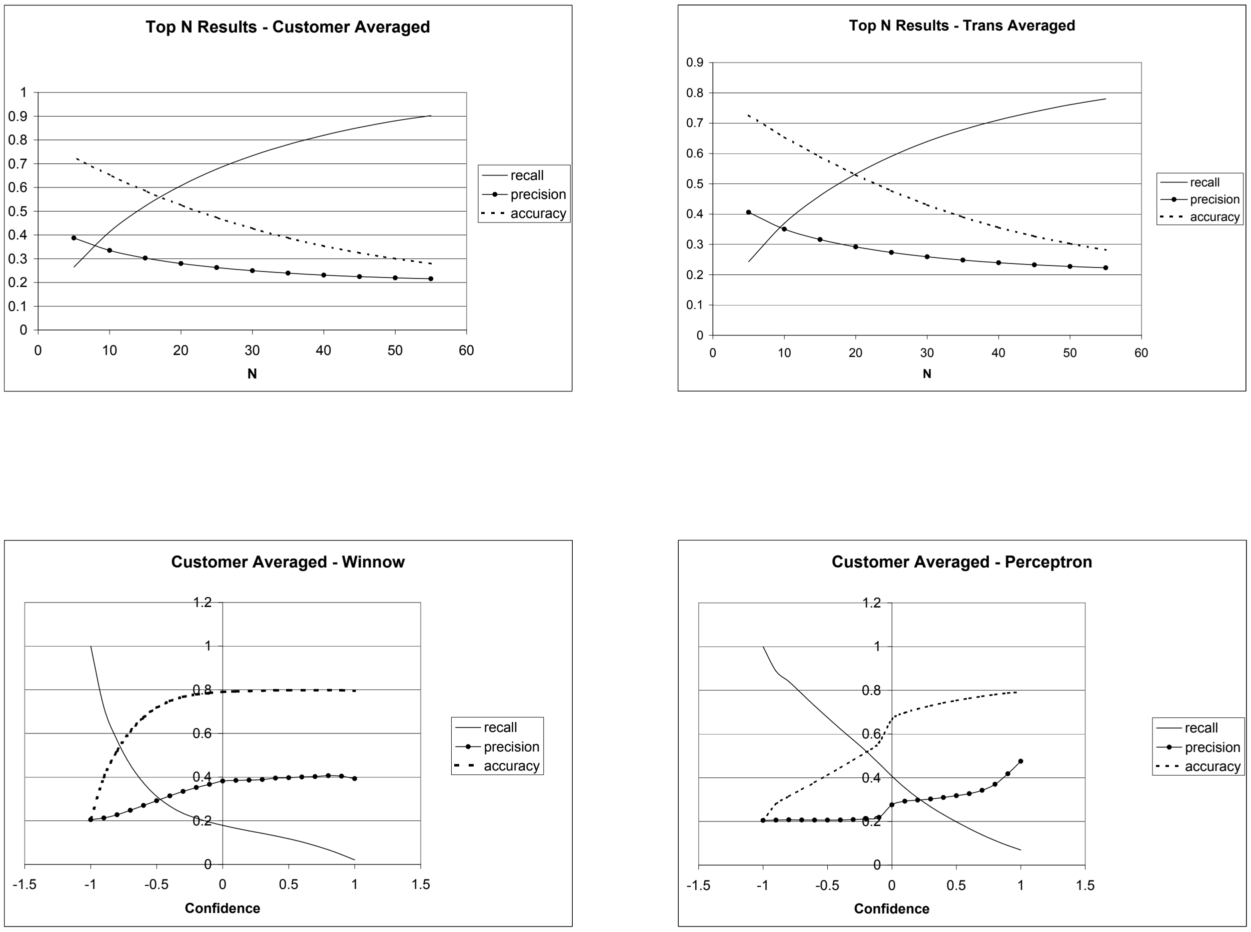

**Figure 2: Top-$n$ results (top), & linear classifier performance at confidence thresholds (bottom).**

classes. For classes with large training set sizes, using the learned classifiers gives us an advantage in terms of precision and accuracy. We can see this advantage by comparing the performance of the hybrid classifiers to the performance of the *top-n* classifier for higher values of $n$ as seen in Fig. 3. This precision gain is important to only show the most relevant suggestions to the customer as they enter the store, as we have limited screen space and want to utilize it intelligently.

Given such an extremely large and imbalanced datasource, metaclassification approaches such as the simple hybrid technique described, along with other metalearning methods current in KDD and machine learning, seem to be particularly suited to prediction tasks involving transactional market basket data. One technique we intend to experiment with in future work to overcome the problem of data sparsity is the shrinkage approach detailed in [12]. In this line of work, a hierarchy is imposed over large numbers of classes, and statistics about groups of classes are used to smooth conditional probability estimations for classes with small numbers of datapoints. These smoothed conditionals can be used in likelihood maximization techniques such as Naive Bayes to improve the accuracy in predicting the subclasses.

The other major distinctive feature of the data source we use is its high degree of systematic (non-random) noise due to customers forgetting to buy items they intended to buy. Although in this work we do not attempt to modify the classification algorithms used to correct for this noise, it has been shown that linear classifiers such as Perceptron (with some modifications) can learn from examples with label noise [8, 5]. In practice linear classifiers are often able to learn well in the presence of classification noise. Based on the assumptions made about the distribution of forgotten purchases in the dataset, we can estimate the degree to which classifiers used in our experiments are robust to the label noise. For example, several of the algorithms exhibit enhanced precision when labels for instances of forgetting are manually flipped to become positive, while the random baseline technique performs the same. While the number of true positives do increase, not all the added positive examples are classified correctly, so in some cases the overall recall decreases or remains constant. But in Table 3 we show the number of added positive examples "recaptured" by the different classification algorithms, suggesting a measure of their relative robustness. Here as in the earlier results, we estimate the forgetting using a constant $c = 1$ (as explained in Sec. 3), which results in 11% of each customer's purchases per transaction being forgotten. The total number of examples for which we flip labels from negative to positive throughout all test sets in this case is 47916. This number represents a relative upper bound for the amount of purchases we can recapture given our assumptions.

When the system we present in this paper is implemented in a retail store setting, we expect the noise in the new data collected to decrease. As the shopping list predictor is used regularly, the forgetting behavior of customers should take place less often as now they are reminded of purchases that would otherwise be forgotten. An interesting aspect to study in the future would be the effects of such an implementation and how the reduction in noise affects each classifier. In general, we believe that deploying data mining systems may result in changes to the data that is being collected to train those systems and that this change can provide new challenges and open problems for the data mining community

| | **recaptured** |
|---|---|
| top10 | 10620 |
| Perceptron | 20244 |
| Winnow | 5251 |
| C4.5 | 9134 |
| Hybrid-Per | 23489 |
| Hybrid-Win | 12270 |
| Hybrid-C4.5 | 15405 |

**Table 3: Number of forgotten purchases recaptured.**

## 5. CONCLUSION

In this work we consider the difficult and useful problem of predicting customer purchase decisions from transaction based individual purchase history. This problem is interesting from the perspective of the KDD and machine learning community as a large-scale experimental application of well-known classification techniques, in a time-sequence domain with a high degree of systematic noise. From a business perspective the advent of technologies such as shopping cart mounted displays offers us a unique opportunity to apply personalization techniques to present individually tailored information to customers as they shop. Using individual grocery purchase history, we predict shopping lists for a large set of customers of a major grocery chain. These lists are useful as a tool to build customer satisfaction, as well as a means of reclaiming otherwise forgotten purchases. Our calculations show that our shopping list prediction system can result in an increase of revenues for the store by up to 11%.

Additionally the shopping lists we provide become the basis for a larger system for the delivery of individually relevant promotional strategies. We show that we can predict purchase categories with a high degree of recall per transaction and reasonable precision, and that given certain assumptions about the distribution of forgotten purchases, we can reclaim many of these purchases.

## 6. REFERENCES

[1] Symbol Technologies Inc. http://www.symbol.com.

[2] G. Adomavicius and A. Tuzhilin. Using data mining methods to build customer profiles. *IEEE Computer*, 34(2):74–82, 2001.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of 20th Int'l Conference on Very Large Data Bases*, Santiago, Chile, 1994.

[4] R. Bellamy, J. Brezin, W. Kellogg, and J. Richards. Designing an e-grocery application for a palm computer: Usability and interface issues. *IEEE Communications*, 8(4), 2001.

[5] A. Blum, A. M. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *IEEE Symposium on Foundations of Computer Science*, pages 330–338, 1996.

[6] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A

case study. In *Knowledge Discovery and Data Mining*, pages 254–260, 1999.

[7] A. Carlson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May 1999.

[8] E. Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Proc. of 38th Symposium on Foundations of Computer Science*, pages 514–523, Miami, FL, 1997. IEEE.

[9] A. Ehrenberg. *Repeat-Buying: Facts, Theory, and Applications*. Charles Griffin & Company Limited, London, 1988.

[10] A. Geyer-Schulz, M. Hahsler, and M. Jahn. A customer purchase incidence model applied to recommender systems. In *WebKDD2001 Workshop*, San Francisco, CA, Aug. 2001.

[11] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[12] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[13] E. Newcomb, T. Pashley, and J. Stasko. Mobile computing in the retail arena. In *Proceedings of the conference on Human factors in computing systems (CHI2003)*, pages 337–344. ACM Press, 2003.

[14] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992.

[15] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[16] D. Roth. Learning in natural language. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 898–904, 1999.