Feature-Based Recommendation System

Eui-Hong (Sam) Han
iXmatch Inc.
5555 West 78th Street, Suite E
Minneapolis, MN 55439
sam@ixmatch.com

George Karypis*
iXmatch Inc.
5555 West 78th Street, Suite E
Minneapolis, MN 55439
karypis@cs.umn.edu

ABSTRACT

The explosive growth of the world-wide-web and the emergence of e-commerce has led to the development of *recommender systems*—a personalized information filtering technology used to identify a set of N items that will be of interest to a certain user. User-based and model-based collaborative filtering are the most successful technology for building recommender systems to date and is extensively used in many commercial recommender systems. The basic assumption in these algorithms is that there are sufficient historical data for measuring similarity between products or users. However, this assumption does not hold in various application domains such as electronics retail, home shopping network, on-line retail where new products are introduced and existing products disappear from the catalog. Another such application domains is home improvement retail industry where a lot of products (such as window treatments, bathroom, kitchen or deck) are custom made. Each product is unique and there are very little duplicate products. In this domain, the probability of the same exact two products bought together is close to zero. In this paper, we discuss the challenges of providing recommendation in the domains where no sufficient historical data exist for measuring similarity between products or users. We present feature-based recommendation algorithms that overcome the limitations of the existing *top-N* recommendation algorithms. The experimental evaluation of the proposed algorithms in the real life data sets shows a great promise. The pilot project deploying the proposed feature-based recommendation algorithms in the on-line retail web site shows 75% increase in the recommendation revenue for the first 2 month period.

Categories and Subject Descriptors

D.m [**Software**]: Miscellaneous; J.m [**Computer Applications**]: Miscellaneous

*also with the Department of Computer Science and Engineering, University of Minnesota

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

General Terms

Algorithms

Keywords

E-Commerce, Collaborative filtering, Recommender systems, Product Features, Web Retailer

1. INTRODUCTION

The explosive growth of the world-wide-web and the emergence of e-commerce has led to the development of *recommender systems* [16]. Recommender systems are personalized information filtering technology used to either predict whether a particular user will like a particular item (*prediction problem*) or to identify a set of N items that will be of interest to a certain user (top-N *recommendation problem*). In recent years, recommender systems have been used in a number of different applications [22, 9, 12, 23, 21, 11, 15, 3] such as recommending products a customer will most likely buy; movies, TV programs, or music a user will find enjoyable; identifying web-pages that will be of interest; or even suggesting alternate ways of searching for information. An excellent survey of different recommender systems for various applications can be found in [21, 16].

Over the years, various approaches for building recommender systems have been developed that utilize either demographic, content, or historical information [9, 1, 2, 22, 23, 12]. Among them, collaborative filtering (CF), which relies on historical information, is probably the most successful and widely used technique for building recommender systems [17, 12]. The first system to generate automated recommendations was the GroupLens system [17, 12], it provided users with personalized recommendations on Usenet postings. The recommendations for each individual were obtained by identifying a neighborhood of similar users and recommending the articles that this group of users found useful.

Two approaches have been developed for building CF-based *top-N* recommender systems. The first approach, referred to as *user-based* [22, 12, 18, 17, 8, 19], relies on the fact that each person belongs in a larger group of similarly-behaving individuals. As a result, items (e.g., products, movies, books, etc.) frequently purchased/liked by the various members of the group can be used to form the basis of the recommended items. The second approach, known as *model-based* [22, 4, 18, 25, 11, 20, 10, 6], analyzes the historical information to identify relations between the different items such that the purchase of an item (or a set of items) of-

ten leads to the purchase of another item (or a set of items), and then use these relations to determine the recommended items. One of the *model-based* approach called *item-based* top-N *recommendation algorithms* [20, 10, 6] provide recommender systems that can scale to very large datasets and compute recommendations in almost real-time. In addition, these algorithms produce recommendations and predictions whose quality is either comparable or better than those produced by other much slower algorithms.

These algorithms build the recommendation model by analyzing the similarities between the various items and then use these similar items to identify the set of items to be recommended. The basic assumption in these algorithms is that there are sufficient historical data for measuring similarity between items. However, this assumption does not hold in various application domains. For example, when a new item X is introduced to the market, the similarity between X and other existing items cannot be measured because none of the customers bought X before. The problem compounds if X has limited quantity and runs out in short time span. By the time enough historical data are gathered to measure the similarity between X and other items, X is no longer available. Another example can be found in a domain such as home improvement retail industry where a lot of products (such as window treatments, bathroom, kitchen or deck) are custom made. Each item is unique and there are very little duplicate items. In this domain, the probability of the same two items bought together is close to zero.

In this paper, we will discuss the challenges of providing recommendation in the domains where no sufficient historical data exist for measuring similarity between items. We present feature-based recommendation algorithms that overcome the limitations of the existing *item-based* top-N *recommendation algorithms*. We will provide experimental evaluation of the proposed algorithms in the real life data sets. We also provide results from the pilot project deploying the proposed feature-based recommendation algorithms in the on-line retail web site.

The rest of this paper is organized as follows. Section 2 presents a brief survey of the related research on collaborative filtering-based recommender algorithms. Section 3 describes the feature-based recommendation algorithms. Section 4 provides the experimental evaluation of the proposed algorithms and shows the results from the pilot project. Finally, Section 5 provides some concluding remarks.

2. RELATED WORK

User-based collaborative filtering is the most successful technology for building recommender systems to date and is extensively used in many commercial recommender systems. In general, user-based systems compute the *top-N* recommended items for a particular user by following a three-step approach [22, 12, 19]. In the first step, they identify the k users in the database that are the most similar to the active user. During the second step, they compute the union of the items purchased by these users and associate a weight with each item based on its importance in the set. Finally, in the third step, they select the N items from that union that have the highest weight and have not already been purchased by the active user as the items to be recommended. Within that framework, the method used to determine the k most similar users and the scheme used to determine the importance of the different items play the most critical role in the overall performance of the algorithm. Commonly, the similarity between the users is computed by treating them as vectors in the item-space and measuring their similarity via the cosine or correlation coefficient functions [18, 19], whereas the importance of each item is determined by how frequently it was purchased by the k most similar users. However, alternate approaches for both of these steps have been explored and shown to lead to somewhat better results. A detailed survey of different user-based algorithms and a comparison of their performance can be found in [18, 8, 19].

Despite the popularity of user-based recommender systems, they have a number of limitations related to scalability and real-time performance. The computational complexity of these methods grows linearly with the number of customers that in typical commercial applications can grow to be several millions. Furthermore, even though the user-item matrix is sparse, the user-to-user similarity matrix is quite dense. This is because, even a few frequently purchased items can lead to dense user-to-user similarities. Moreover, real-time *top-N* recommendations based on the current basket of items, utilized by many e-commerce sites, cannot take advantage of pre-computed user-to-user similarities. Finally, even though the throughput of user-based recommendation algorithm can be increased by increasing the number of servers running the recommendation algorithm, they cannot decrease the latency of each *top-N* recommendation that is critical for near real-time performance. One way of reducing the complexity of the nearest-neighbor computations is to cluster the users and then to either limit the nearest-neighbor search among the users that belong to the nearest cluster or use the cluster centroids to derive the recommendations [24, 15]. These approaches, even though they can significantly speed up the recommendation algorithm, they tend to decrease the quality of the recommendations.

To address the scalability concerns of user-based recommendation algorithms a variety of model-based recommendation techniques have been developed. Billsus and Pazzani [4] developed a model-based recommender system by treating the *top-N* recommendation problem as a classification problem, in which the goal was to classify the items purchased by an individual user into two classes like and dislike. A classification model based on neural networks was built for each individual user where the items purchased by the user were thought of as the examples and the users as the attributes. To reduce the dimensionality a singular value decomposition of the user-item matrix was obtained. The prediction on an item was computed by constructing an example for that item and feeding it to the classifier. The authors reported considerable improvements over the traditional user-based algorithms. Though this approach is quite powerful it requires building and maintaining a neural network model for each individual user in the database, which is not scalable to large databases. Breese et al. [18] presented two model-based algorithms for computing both predictions and *top-N* recommendations. The first algorithm follows a probabilistic approach in which the users are clustered and the conditional probability distribution of different items in the cluster was estimated. The probability that the active user belongs to a particular cluster given the basket of items was then estimated from the clustering solution and the probability distribution of items in the cluster. The clustering solution for this technique is computed using the expectation maximization (EM) principle.

The second algorithm is based on Bayesian network models where each item in the database is modeled as a node having states corresponding to the rating of that item. The learning problem consists of building a network on these nodes such that each node has a set of parent nodes that are the best predictors for its rating. They presented a detailed comparison of these two model-based approaches with the user-based approach and showed that Bayesian networks model outperformed the clustering model as well as the user-based scheme. Heckerman et al. [7] proposed a recommendation algorithm based on dependency networks instead of Bayesian networks. Though the accuracy of dependency networks is inferior to Bayesian networks they are more efficient to learn and have smaller memory requirements. Agrawal et al. [25] presented a graph-based recommendation algorithm where the users are represented as the nodes in a graph and the edges between the nodes indicate the degree of similarity between the users. The recommendations for a user were computed by traversing nearby nodes in this graph. The graph representation of the model allows it to capture transitive relations which cannot be captured by nearest neighbor algorithms and the authors reported better performance than the user-based schemes.

A number of different model-based approaches have been developed that use item-to-item similarities as well as association rules. Shardanand and Maes [22] developed an item-based prediction algorithm within the context of the Ringo music recommendation system, referred to as *artist-artist*, that determines whether or not a user will like a particular artist by computing its similarity to the artists that the user has liked/disliked in the past. This similarity was computed using the Pearson correlation function. Sarwar et al. [20] further studied this paradigm for computing predictions and they evaluated various methods for computing the similarity as well as approaches to limit the set of item-to-item similarities that need to be considered. The authors reported considerable improvements in performance over the user-based algorithm. Mobasher et al. [14] presented an algorithm for recommending additional webpages to be visited by a user based on association rules. In this approach, the historical information about users and their web-access patterns were mined using a frequent itemset discovery algorithm and were used to generate a set of high confidence association rules. The recommendations were computed as the union of the consequent of the rules that were supported by the pages visited by the user. Lin et al. [13] used a similar approach but they developed an algorithm that is guaranteed to find association rules for all the items in the database. Finally, within the context of using association rules to derive *top-N* recommendations, Demiriz [5] studied the problem of how to weight the different rules that are supported by the active user. He presented a method that computes the similarity between a rule and the active user's basket as the product of the confidence of the rule and the Euclidean distance between items in the antecedent of association rule and the items in the user's basket. He compared this approach both with the item-based scheme described in [10] and the dependency network-based algorithm [7]. His experiments showed that the proposed association rule-based scheme is superior to dependency networks but inferior to the item-based schemes.

3. FEATURE-BASED *top-N* RECOMMENDATION ALGORITHMS

The basic knowledge that the feature-based recommendation algorithms tries to discover can be summarized as "people who bought products *with features like* these also bought a product *with features like* these." An example of this kind of knowledge in an electronics retail shop can be "people who bought a TV *with features like* HDTV, Rear-Project, HDMI Input, Built-In HDTV Tuner, and IEEE 1394 (FireWire) DV interface also bought a DVD player *with features like* Progressive-Scan Multiformat, HDMI Output, 3D virtual surround sound, Dolby Digital 2-channel down-mixing, MultiMediaCard, and Secure Digital." Note that this statement is about features of products not about particular products. Now assume a customer is checking out a brand new TV that significantly matches the product features of this statement. The recommendation system will be able to recommend a DVD that matches the product features of this statement regardless of whether that particular DVD is an existing product or a new product. Consider another example in a home improvement retailer: "people who bought windows *with features like* clad-wood, safety glass, and bay window also bought window treatments *with features like* natural woven shade, bamboo, semi-sheer, and scarves." When a customer is considering custom windows with the features of this statement, the salesperson can recommend window treatments with features in this statement.

In this section, we will describe three feature-based recommendation algorithms that capture the knowledge discussed here. The first two algorithm is designed to provide recommendations in the domain where the product catalog changes frequently (e.g., electronics retail). The third algorithm is designed to provide recommendations for product catalog with custom products.

3.1 Frequently Changing Product Catalog

In this type of catalog (e.g., electronics retail, home shopping network, on-line retail), new products are introduced frequently and existing products become out of stock or discontinue frequently. Challenges of recommendation system in this catalog include how to use new products in the basket for recommendation and how to recommend new products.

We present two methods to address these challenges. We assume that we are given a training basket data corresponding to past sales history. Each basket corresponds to single transaction where a set of products are sold to a customer. We have a corresponding training product catalog containing all the products in the training basket data and their product features. We also have a current product catalog that contain available products (including new products that are not in the training product catalog) and their product features.

In the first method, given a training basket data, we build a recommendation model M using *item-based* top-N *recommendation algorithms* described in [10]. The recommendation steps can be summarized as follows:

1. Given a set of products X in a shopping basket, find similar products S of X using the product features of X from the training product catalog.

2. Find recommended products R of the set S using the recommendation model M.

3. Find recommended features F by summarizing or aggregating product features of R.

4. Find *top-N* matching products using F from the current product catalog.

In step 1, existing products (i.e., products in the training product catalog) in X will find themselves as the matching products. New products in X will find the most similar products from the training product catalog. Hence, S contains the existing products from X and the most similar existing products of new products in X. By using S that contains similar products of new products in step 2, this method utilizes new products of the shopping basket in the recommendation process.

In step 3, we utilize the recommendation score or confidence in the summary/aggregation of product features. The score of one feature is the sum of recommendation score of each recommended product that has this feature. Hence, features that are in many recommended products and are in highly recommended products are scored highly. These product feature scores are in turn used in step 4 for matching products.

In step 4, each product is scored by adding scores of features in F that this product has. By scoring products from the current product catalog using the recommended features and their scores, the best matching products from the currently available products (including new products that do not exist in the training basket) are recommended.

This proposed method utilizes new products in the basket for recommendation and also recommends new products. However, products that are introduced to the training basket recently (or lately) and do not have many shopping baskets with these products are not utilized in full extent in the recommendation process. For instance, assume a very popular product was introduced toward the end of period for training data. There are very small number of baskets with this product. However, there will be many baskets with this product in the deployment. The recommendation with this product will provide no hits or very low confidence recommendation, because there are very small number of baskets in the training set. A better option is to regard this product as a new product and find other similar products from the training product catalog. Now the challenge is how to select some of existing products as new products. A frequency or support threshold-based approach is feasible, but always presents a difficult task of determining the right threshold value.

We propose another method that avoids the issue with threshold. In this method, instead of using *item-based* top-N *recommendation algorithms* to determine recommended products, we use the features of products in the basket directly.

In this method, given a training basket data, we build association rules AR as follows. For each basket in the training basket data, for each product in the basket, construct a rule such that this product is in the consequence of the rule and all other products are in the antecedent of the rule. For example, given a basket {a, b, c}, we will have rules {a, b} $\rightarrow$ c, {a, c} $\rightarrow$ b and {b, c} $\rightarrow$ a. The recommendation steps can be summarized as follows:

1. Given a set of products X in a shopping basket, find matching rules R of X by matching the product features of X against the features of the products in the antecedent of rules in AR.

2. Find recommended features F by summarizing or aggregating the features of products in the consequence of the rules in R.

3. Find *top-N* matching products using F from the current product catalog.

Note that in step 1 we simply use the features of the products in the basket and find association rules that match the features directly. This step eliminates the distinction between existing products and new products and uses all the features of the basket in the recommendation process. In step 2, similar to the first method, the matching scores from the step 1 are used to score the features. Step 3 is the same as the step 4 of the first method.

3.2 Product Catalog with Custom Products

In this type of catalog (e.g., home improvement retail industry, built to order on-line retail), complete enumeration of all possible products is not feasible. Only high-level or categorical listing with sample products is feasible. For example, in the home improvement retail outlet, high level categories like window treatments, bathroom, kitchen, or deck can be listed with some examples of each of these category. From the sales record, we will not find many exact duplicate products or configurations. We might find similar bathroom configurations from different sales records, but they will not match exactly. Hence, any user- or item-based recommendation model will not work in this catalog data set.

We propose a recommendation method that is based on clustering and feature matching for this catalog data set. When the product catalog does not have a product hierarchy, we construct a product hierarchy from the sales data using clustering algorithms [26]. Given a product hierarchy, we roll up leaf nodes in the hierarchy if they do not have sufficient number of products in the node and split leaf nodes if they have too many products and the cluster cohesiveness/tightness [26] is low. The goal of rolling up and splitting nodes in the product hierarchy is to have leaf nodes that have sufficient number of products and yet have high cluster cohesiveness.

Given a training basket data and the product hierarchy, we build a recommendation model M using *item-based* top-N *recommendation algorithms* described in [10]. In this model, item corresponds to the cluster (the leaf node) of the product hierarchy that each product belongs to.

1. Given a set of products X in a shopping basket, find the set of clusters C that products in X belong in the product hierarchy.

2. Find recommended clusters R of the set C using the model M.

3. For each recommended cluster in R, find recommended features F of products in the cluster by summarizing/aggregating the features of products in the cluster.

4. For each recommended cluster, either show the cluster and its recommended features F or find the matching custom products from the training basket using F.

The recommended features F of a cluster can be determined as the centroid vector of the product features in the cluster, or the product features of the medoid product of the cluster. Another option is to use some business rules to select representative product of the recommended cluster and use the features of these representative products. The business rules can be the most popular products in terms of sales, most profitable products, products with supplier incentives, or products with most inventory. However, these options lack differentiation power as any combination of products that give same C in step 1 will get the same recommendation. Third option is to select features basket-sensitive way in which the products in the basket of X are considered. In this option, given a recommended cluster r in R, we first select past shopping baskets that contain products from cluster C and the recommended cluster r. We then rank these shopping baskets by matching features of X to the products from cluster C. We then score features of products in r from these baskets utilizing matching scores.

An example using electronic retails data will demonstrate the proposed method. Note that electronic retails is not the proper domain for this kind of catalog, but is used to explain the process because this domain is well known. Assume that a customer is buying a TV and a VCR. The cluster (or leaf node in the product hierarchy) of these products are high-end TV and high-end VCR. Given these clusters, assume that the recommended cluster is high-end HOME AUDIO & SPEAKER using item-based recommendation of clusters of past sales baskets.

Now the question is what are the key features of the high-end HOME AUDIO & SPEAKER to use for final product recommendation. First option is to select features that are most common among all the high-end HOME AUDIO & SPEAKERs. Second option is to select the most popular (in terms of sales, or some other characteristics) high-end HOME AUDIO & SPEAKERs and use the features of these HOME AUDIO & SPEAKERs. Third option is to select features based on the past shopping baskets that contain high-end TV, high-end VCR and high-end HOME AUDIO & SPEAKER. From these baskets, we will select baskets that are most similar in terms of features of TV and VCR of the current shopping basket. Then use the features of HOME AUDIO & SPEAKERs in these baskets for final product recommendation.

Given the set of recommended features, we can either recommend features themselves and/or recommend custom products that match these features. So in the example above, we can recommend features such as Videostage 5 decoding and post-processing circuitry, ADAPTiQ audio calibration system, AM/FM tuner with 50 stations. Then these features can be used to custom build an audio system. We can also find matching custom products from the past sales with these recommended features.

4. EXPERIMENTAL EVALUATION

4.1 Data Sets and Evaluation Metrics

We obtained the sales data from an upscale cable television and on-line retailer for evaluation. The data set with 4.5 million records was a random subset from one year sales data. We took first 9 months as training set and the last 3 months as test set. There were 8,908 products in this sales data. From these data sets, we constructed a shopping basket as products purchased by the same customer within 2 day period. We dropped shopping baskets with single product, because these baskets cannot be used for recommendation evaluations.

Table 1: Basket size distribution.

Size of Basket	Training Basket	Test Basket
2	89,385	67,896
3	22,507	20,066
4	7,729	7,989
5	3,226	3,828
6	1,712	2,108
7	935	1,230
8	591	784
9	374	538
10	259	362
11 and over	836	1,461

In the training set, there were 127,554 shopping baskets with at least 2 products. In this set, there were 5,511 unique products. In the test set, there were 106,263 shopping baskets with average number of products of 2.91. Note that sales per month within the last 3 month of the data was much higher than the previous 9 months. In this data set, there were 7,635 unique products and 2,803 of them are new products that did not exist in the training set. Table 1 shows the number of baskets for different basket size.

We also obtained manual recommendations for these products. Out of 8,908 products, 1,912 products have manual recommendations. These recommendations were provided by product suppliers or domain experts.

We compare the following 5 approaches in this experimental evaluation.

- Manual: recommendation using manual recommendations.
- Item: recommendation using *item-based* top-N *recommendation algorithms* described in [10].
- Feature: recommendation using the second feature-based recommendation method discussed in Section 3.1.
- Item+Feature: recommendation using the combined results of Item and Feature.
- Item+Feature+Manual: recommendation using the combined results of Item, Feature, and Manual.

We combined recommendations of different methods by normalizing recommendation scores of each method, adding up the scores of recommended products from different methods, and then rank the added scores of recommended products.

We evaluated these recommendation methods for the following two situations.

- Product Page Case: User is viewing a web page of single product and product recommendation is needed based on this single product.
- Checkout Page Case: User has one or more products in the basket for check out and product recommendation is needed based on the set of products in the basket.

We show the number of hits in top 1, 3, and 6 recommended products of each method. In most real deployment of recommendation system, at most 3 products are presented.

Table 2: Product Page Case: 310,106 possible hits

method	top-1	top-3	top-6
Manual	34,657	46,795	46,795
Item	17,296	33,590	46,672
Feature	11,072	26,201	38,965
Item+Feature	21,490	40,706	58,202
Item+Feature+Manual	43,185	72,634	90,029

Table 3: Product Page Case on Test Baskets with New Products: 239,905 possible hits

method	top-1	top-3	top-6
Manual	28,846	39,288	39,288
Item	7,149	14,791	21,568
Feature	7,092	16,072	24,654
Item+Feature	11,342	22,552	33,465
Item+Feature+Manual	33,160	54,326	65,164

However, some of the recommended products might be out of stock and it might require more recommended products. Top 6 recommendation will cover cases like this.

Note that the manual recommendations were used in the sales process, and the data set we received contains sales influenced by these recommendations. Hence, the recommendation results of the Manual method needs to be understood with this bias.

4.2 Product Page Case

There were 106,263 test baskets and were 310,106 recommendations for this test case. Given a test basket, we made recommendation based on each product in the basket. We consider the recommendation to be correct or a hit if the top-k recommendation contains any of the products in the basket.

Table 2 shows the number of hits in "Product Page Case". For top-3 recommendation results, Manual has 15.1% correct recommendations, Item has 10.8%, Feature has 8.4%, Item+Feature has 13.1%, and Item+Feature+Manual has 23.4%. This result shows that Manual has the best single method result followed by Item and Feature. Item+Feature shows the improvement over Item or Feature. This result demonstrates that each method has different strength and the combined method benefits from these strengths. Item+Feature+Manual has the best performance at 23.4%, which is 50.0% improvement from the Manual result.

We also extracted test baskets that contain new products to see the effectiveness of the Feature method in these baskets. Table 3 shows the number of hits for this subset of test baskets. There were 76,304 baskets and 239,905 recommendations were made. For top-3 recommendation results, Manual has 16.4% correct recommendations, Item has 6.2%, Feature has 6.7%, Item+Feature has 9.4%, and Item+Feature+Manual has 22.6%. This result shows that Feature outperforms Item in this subset as expected.

4.3 Checkout Page Case

There were 106,263 test baskets and were the same number of recommendations for this test case. Given a test basket of size m, we made recommendation based on the first $m-1$ products in the basket. We consider the recommendation to be correct or a hit if the top-k recommendation contains the last product of the basket.

Table 4: Checkout Page Case: 106,263 possible hits

method	top-1	top-3	top-6
Manual	6,895	9,696	10,026
Item	3,821	6,716	9,573
Feature	2,568	5,384	7,496
Item+Feature	3,727	6,941	10,434
Item+Feature+Manual	7,383	12,949	17,174

Table 5: Checkout Page Case on Test Baskets with New Products: 76,304 possible hits

method	top-1	top-3	top-6
Manual	4,818	7,257	7,564
Item	768	1,797	2,814
Feature	941	1,984	2,810
Item+Feature	868	2,307	3,763
Item+Feature+Manual	4,195	7,882	10,138

Table 4 shows the number of hits in "Checkout Page Case". For top-3 recommendation results, Manual has 9.1%, Item has 6.3%, Feature has 5.1%, Item+Feature has 6.5%, and Item+Feature+Manual has 12.2% correct recommendations. Item+Feature+Manual has the best performance at 12.2%, which is 34.1% improvement from the Manual result. This result is similar to that of the "Product Page Case".

Table 5 shows the number of hits for the subset of test baskets that contain new products. For top-3 recommendation results, Manual has 9.5% correct recommendations, Item has 2.4%, Feature has 2.6%, Item+Feature has 3.0%, and Item+Feature+Manual has 10.3%. This result is again similar to that of the "Product Page Case".

4.4 Pilot Project

The upscale cable television and on-line retailer that provided the test data for the experiments carried out a pilot project. In this project, this retailer deployed a recommendation scheme of Manual+Item+Feature in "Product Page Case" described in Section 4.1. The pilot shows a great promise as this retailer saw 75% increase in the recommendation revenue for the first 2 months of the evaluation. This result is better than the improvement of 50% shown in Section 4.2 for the similar setting. The difference is due to the bias of the test data we used for the experiments. The manual recommendations were used in the sales process, and the test data set we received contains sales influenced by these recommendations. Hence, the result of Manual shown in the experiment is higher than the real performance, and the improvement in the experiment is lower than the real.

5. CONCLUDING REMARKS

In this paper, we discussed the challenges of providing recommendation in the domains where no sufficient historical data exist for measuring similarity between products or users. We presented feature-based recommendation algorithms that overcome the limitations of the existing *top-N* recommendation algorithms. The experimental evaluation of the proposed algorithms in the real life data sets shows a great promise as the combined methods provide 50% improvement over the manual recommendation method. The pilot project deploying the proposed feature-based recommendation algorithms in the on-line retail web site shows

75% increase in the recommendation revenue for the first 2 month period.

6. REFERENCES

[1] M. Balabanovic and Y. Shoham. FAB: Content-based collaborative recommendation. *Communications of the ACM*, 40(3), March 1997.

[2] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 1998 Workshop on Recommender Systems*, pages 11–15. AAAI Press, 1998.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of ACM SIGKDD International Conference*, pages 407–415, 2000.

[4] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of ICML*, pages 46–53, 1998.

[5] A. Demiriz. An association mining-based product recommender. In *NFORMS Miami 2001 Annual Meeting Cluster: Data Mining*, 2001.

[6] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 2003.

[7] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[8] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithm framework for performing collaborative filtering. In *Proceedings of SIGIR*, pages 77–87, 1999.

[9] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of CHI*, 1995.

[10] G. Karypis. Experimental evaluation of item-based top-n recommendation algorithms. In *Proceedings of the ACM Conference on Information and Knowledge Management*, 2001.

[11] B. Kitts, D. Freed, and M. Vrieze. Cross-sell: A fast promotion-tunable customer-item recommendation method based on conditional independent probabilities. In *Proceedings of ACM SIGKDD International Conference*, pages 437–446, 2000.

[12] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[13] W. Lin, S. Alvarez, and C. Ruiz. Collaborative recommendation via adaptive association rule mining. In *International Workshop on Web Mining for E-Commerce (WEBKDD'2000)*, 2000.

[14] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[15] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, and J. Witshire. Discovery of aggregate usage profiles for web personalization. In *Proceedings of the WebKDD Workshop*, 2000.

[16] Resnick and Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[17] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of CSCW*, 1994.

[18] J. s. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertaintly in Artificial Intelligence*, pages 43–52, 1998.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of ACM E-Commerce*, 2000.

[20] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW10*, 2001.

[21] J. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of ACM E-Commerce*, 1999.

[22] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[23] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[24] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems at the 15th National Conference on Artificial Intelligence*, 1998.

[25] J. wolf, C. Aggarwal, K. Wu, and P. Yu. Horting hatches and egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1999.

[26] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. *Machine Learning*, in press, 2003.