RC 21651 (Log 97590) 01 Feb 2000 Computer Science/Mathematics

IBM Research Report

The Intelligent Recommendation Analyzer

Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, Philip S. Yu
IBM T. J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

The Intelligent Recommendation Analyzer

Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu and Philip S. Yu
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{charu, jlwolf, klwu, psyu}@us.ibm.com

Abstract

We introduce a novel approach to web personalization known as the Intelligent Recommendation Analyzer (IRA). In this paper we describe the IRA architecture and basic algorithms, which are fast, scalable, accurate, and require small learning curves and setup costs. We also describe some new and novel features which make IRA stand out from the rest of the personalization software pack.

1 Introduction

Personalization algorithms are in their infancy, but represent an extremely important e-commerce opportunity. Personalization is an important feature to attract customers in e-commerce. In this paper we introduce the Intelligent Recommendation Analyzer (IRA) for personalization in Web applications. IRA contains new, fast and accurate algorithms for personalization, and introduces a number of additional features not previously covered in the literature. The approach spans both collaborative and content based filtering. IRA is a flexible architecture which incorporates multiple recommendation engines. There is no single universal technique for personalizing the sale of all products. Indeed, different products (such as books, CDs, videos, software, groceries, computers, apparel and so on) require inherently different selling techniques. So different combinations of recommendation engines are used in IRA.

The collaborative filtering technique used by Netperception and Firefly is most suitable for selling homogeneous products such as books, CDs, videos or software. IRA has a new and novel collaborative filtering algorithm for such items. Rule based systems such as the one employed in Broadvision shift the responsibility to the store administrator (most likely with the help of expert consultants) to decide on the appropriate rules. In IRA, these rules are built into the recommendation engines. Details are transparent to the store administrator, who only needs to pick the best engine to invoke.

We will describe three recommendation engines employed in IRA. The first one is a new type of rating based engine [2] using a context based collaborative filtering approach. This engine can provide context sensitive recommendations. It is fast, scalable and, compared to conventional collaborative filtering techniques, better able to address rating biases. The second and third engines [1, 4] use a content and a product based product-peer filtering approach, respectively. Consumer ratings of products are not required. In the context based product-peer filtering approach, the product name, description or key words are captured in making recommendations. In contrast to Knowledge Utility, the content is captured automatically. This avoids the large initial setup effort encountered by conventional content based approaches.

A prototype IRA system is available for demonstration purposes. This prototype consists of two 'stores.' The first is a computer store, based on *Shop IBM*. The other is a book/CD/video/software store similar in spirit to *Amazon.com*. We are going to show snapshots of sample queries designed for these two stores from the prototype system.

2 Architecture Overview

Figure 1 presents an overview of the IRA architecture. An e-commerce application receives a client request through a Web server interface, such as the Web-Sphere studio of IBM, and composes a personalized web page for that client. Web-Sphere studio is an IBM Web server product. If the store has a list of promotion items, the personalization objective might be to place the appropriate item or items on the web page for each specific customer. The e-commerce application can invoke the IRA recommendation engine via the IRA front end integrator routine to determine the most suitable promotion item. Standard IRA APIs allow this to be accomplished easily.

In the IRA front-end integrator, the 'situation analyzer' routine can determine the appropriate recommendation engine to invoke. It can also invoke multiple engines and combine the results. Alternatively, the e-commerce application can use a 'thin interface' to invoke the appropriate recommendation engine directly. The 'dynamic session information interface' routine collects data from Web-Sphere. This session information could be the web pages browsed and/or products selected in the shopping basket. The 'rule specification' routine allows the customer to specify rules. The 'real-time information capture and collection' routine captures and collects other real-time information.

Each of the multiple recommendation, target and advertising engines can use different techniques or rules to address different types of product/store requirements. Examples include homogeneous versus heterogeneous product mixes, unique versus commodity products, simple versus complex products, consumable versus durable products and so on. The advertising engine [3] can consider different revenue objectives, such as the number of exposures, clicks or a mixture of the

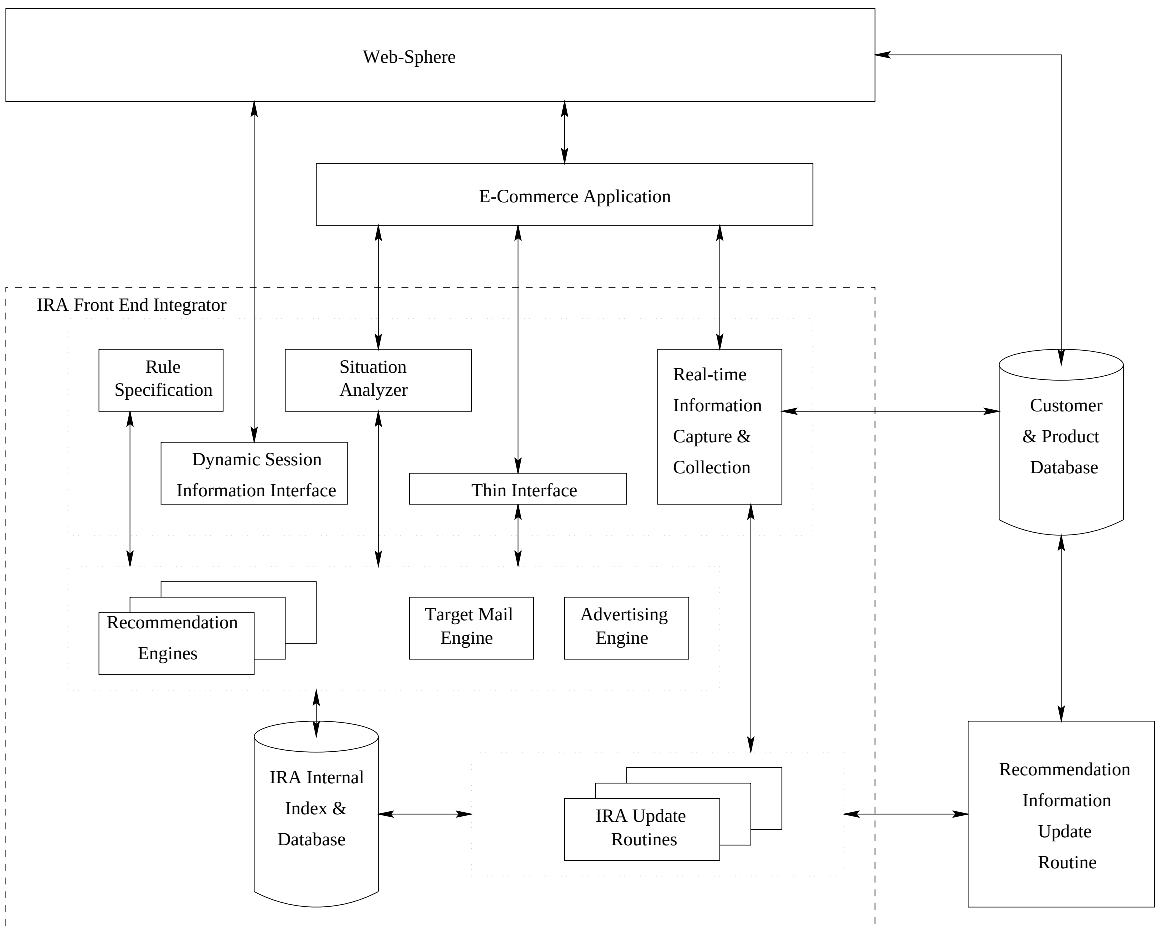

Figure 1: IRA architecture.

two. The target mail engine identifies customer list to send promotional e-mail for specific products or product categories.

The IRA engines maintain their own internal indexing [4] and databases, similar to Netperception. The different engines have different requirements. There is a set of update APIs to invoke the IRA routines which maintain internal IRA data. This update can be done either in batch mode or in real-time.

3 New and Novel Features

We have indicated that IRA is fast, scalable, accurate, requiring small learning curves and set-up costs. Below are a brief selection of other novel IRA features. These features and a number of others differentiate IRA from other personalization packages.

- IRA pre-stores single or multiple recommendations per customer. For scenarios in which customer data is not updated this feature allows for essentially immediate recommendations. Otherwise, the update event can be used to trigger a new pre-store computation.

- IRA keeps track of recommendations previously made to the customer. This can be used to avoid or limit the number of repetitive recommendations, as desired.

- IRA employs a feedback learning process in the personalization algorithm itself. After presenting a recommendation in the rating based engine, for example, IRA gives the customer the opportunity to input the rating the customer would have given to that item. Although this input is optional, it helps generate continually improved recommendations if the customer chooses to respond.

- IRA can optimize its recommendations by revenue or other criteria, if desired, with constraints on the minimum number of times each of the various items are recommended.

4 Rating Based Engine Algorithms

The rating based recommendation engine [2] uses an enhanced context sensitive collaborative filtering technique. As with conventional collaborative filtering techniques, the assumption is that customers will be able to rate at least some sample products. The technique is most applicable to more or less homogeneous products such as books, CDs, videos and software. The IRA approach to collaborative filtering predictions involves two new concepts which we will call *horting* and *predictability*. Here we describe these concepts and their importance at a high level.

We say that customer 1 *horts* customer 2 provided the number of items both customers have rated in common normalized by the number of items rated by customer 1 exceeds some fixed (fractional) threshold. If customer 1 horts customer 2 it does *not* follow that customer 2 horts customer 1. (That is why we do not use the term *cohorts.*) The basic idea is that if customer 1 horts customer 2 then there is enough commonality among the jointly rated (from customer 1's perspective) to decide if customer 2 predicts customer 1 in some fashion or not.

Thus for customer 2 to *predict* customer 1 it is first required that customer 1 hort customer 2. (Note the orderings of customers in the definition.) Secondly it is required that there exists a linear transformation for which the transformed ratings of customer 2 and the ratings of customer 1 are sufficiently close, using a standard metric. Although we don't go into details here, the notion that customer 2 predicts customer 1 is more general than the simple possibility that customer 1 and customer 2 agree closely on the set of jointly rated items. The new notion of predictability incorporates the possibility that customer 1 is more effusive with praise than customer 2, or less effusive, or has the reverse opinions, and so on. If customer 2 predicts customer 1, one can use the transformation to convert the rating of an item by customer 2 into a predicted rating of that item for customer 1. The IRA algorithm generalizes this via a graph theoretic approach. Two distinct data structures are required.

- For each item the algorithm maintains an *inverted index* of all customers who rate that item, sorted in order of increasing customer id.

- The algorithm also maintains a directed graph in which the nodes are the customers and there is a directed arc from customer 1 to customer 2 provided customer 2 predicts customer 1. The corresponding prediction data is stored in the directed graph, as well as the items rated by the various customers and the rating themselves.

To compute a predicted rating in the case of a customer who has recently updated his ratings the algorithm proceeds in three stages: First, the inverted indexes of all items rated by the customer are examined via a merge and count operation, so that the set of customers who are horted by the target customer can be quickly computed. Second, the subset of customers who predict the target customer are found. Third, a shortest path in the directed graph from any customer in the predictive subset to a customer who rates the item is computed. The length of such a directed path will typically be small, and can be found effectively by breadth first search. This allows for an estimated rating of the item based on the composition of transformations. (In the case of a customer who has not recently updated his ratings only the third step is necessary.)

To provide context sensitive recommendations, a product hierarchy categorization is maintained. (This categorization can be a lattice, since a single product may belong to multiple categories.) A *context* here is the session state, which reflects the set of pages browsed and/or products purchased in the current session. Starting from the lowest context level, the algorithm proceeds upwards whenever no satisfactory recommendation can be made, considering higher and higher context levels in order.

Figure 2 shows a sample rating based query from a screen shot of our prototype IRA system. It demonstrates the application of rating based algorithm for recommending a list of promotional items to a customer. It shows that there are four different product categories, books, CDs, Videos and Software. One or more items from each product category can be chosen for promotion to a specific customer. Given a promotional list of product items, the system shows the top recommended items with the highest projected ratings. Other kinds of queries are also available in IRA, such as recommending a product to a list of customers and finding customers with related behavior.

5 Content and Product Based Engine Algorithms

Two recommendation engines [1, 4] have been developed for the case of non-homogeneous products. Examples include computer or grocery stores. The data involves actual buying information as opposed to ratings. Both engines are capable of resolving queries based on customer identification, context, keyword search and promotion lists.

The product based engine uses a clustering method in order to divide the customers into many clusters. Standard clustering methods are not effective because the data typically consists of thousands of dimensions. Accordingly we employ a novel technique [1] which we refer to as *projected clustering*. In projected clustering the idea is to cluster the products based on small subsets of dimensions which are specific to the individual clusters. The clustering techniques are used in a preprocessing phase as opposed to query time. For a given target customer the algorithm finds the closest clusters to his buying behavior which are also relevant to the specified promotion list and context. The set of customers in these clusters are referred to as his *peer group*. The buying behavior of the customers in the peer group is used in order to make predictions about the target customer.

The content based recommendation method is based on textual descriptions of the products. This information can be collected automatically from the product name, description in the web pages or catalog/product database, and product directory name/categorization. A preprocessing algorithm is applied in order to find those features in the data which are most relevant to buying

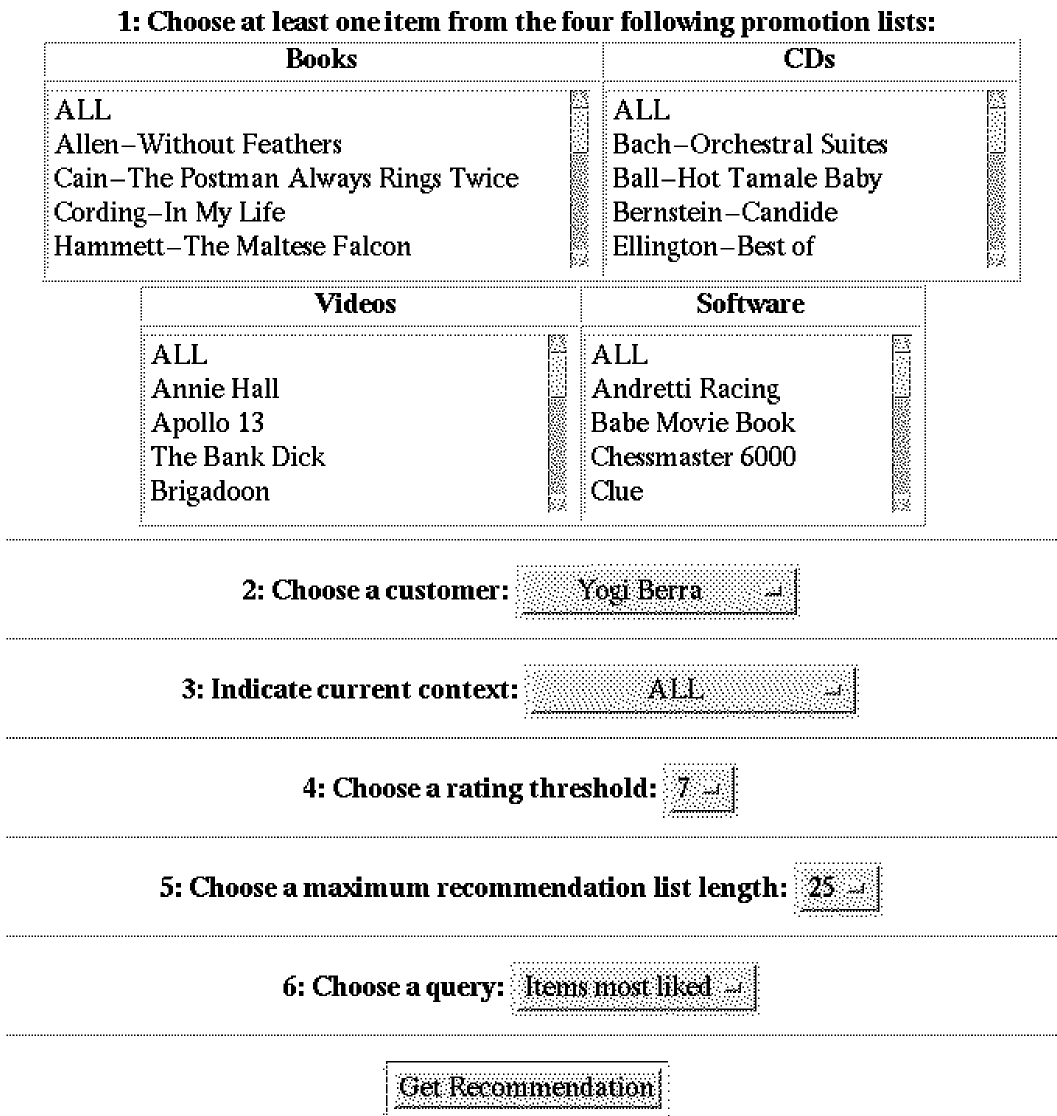

Figure 2: Sample Rating Based Query.

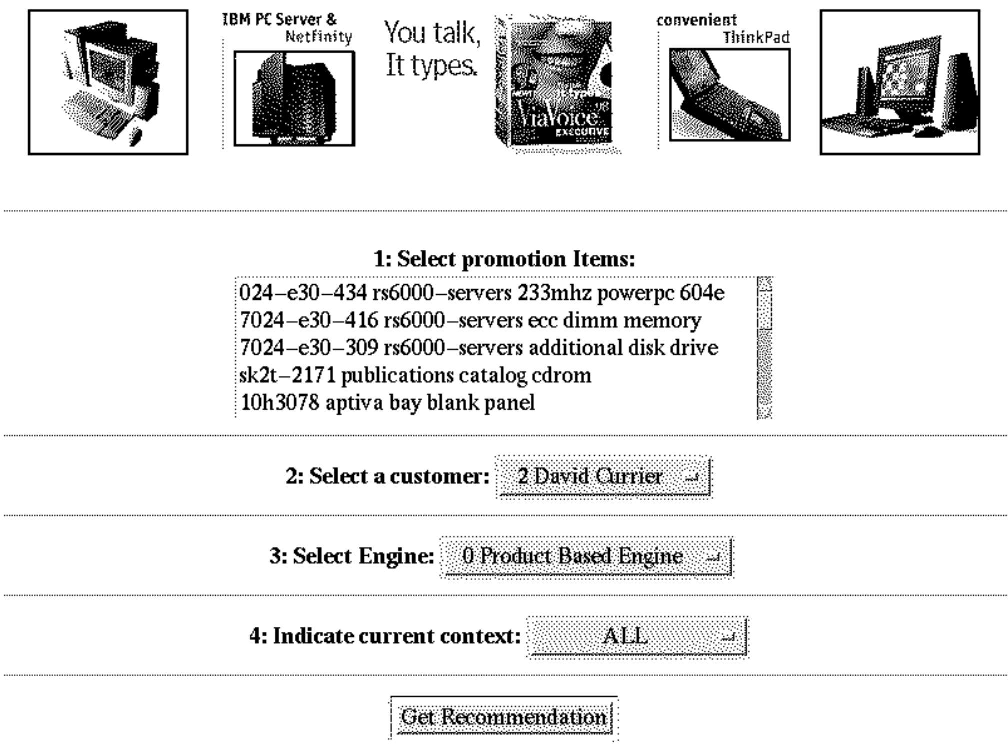

Figure 3: Sample Product Based Query.

behavior. As a result, a reduced textual description is created for each of the products in the database using only the relevant features. Each customer can also be characterized by a textual description of the products that he has bought or browsed. A *taxonomy* of customers is created using this textual description. The taxonomy is a hierarchical clustering of nodes, each node in the taxonomy corresponding to a group of customers who exhibit similar buying behavior.

For a given target customer, the closest nodes in the taxonomy are used in order to find the group of peers who are most relevant to the buying behavior of the target. Care is taken so that the corresponding nodes in the taxonomy are relevant to specified context, promotion lists and/or keywords. The buying behavior of this peer group is analyzed in order to give recommendations for the target customer.

Figure 3 shows a sample product based query from a screen shot of our prototype IRA system. It shows the application of product based algorithm for recommending promotional items to a customer. It starts with a promotional list and recommends top items to a customer. Other applications are also available, such as recommending items with keyword search, finding target customers for a promotion list, finding target customers for a set of keywords, and finding a set of closely associated products given a set of products.

6 Conclusion

Web personalization algorithms are fast becoming an essential component of e-commerce. The new Intelligent Recommendation Algorithm contains novel, fast, accurate and scalable techniques. It is a complete package, including rating, context, content, product and advertising engines. In this paper we have described the architecture of IRA and shown examples of its use.

References

[1] C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu and J. Park, "Fast Algorithms for Projected Clustering", *Proceedings of ACM SIGMOD Conference*, Philadelphia PA, pp. 61-72, 1999.

[2] C. Aggarwal, J. Wolf, K.-L. Wu and P. Yu, "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering", *Proceedings of ACM SIGKDD Conference*, San Diego CA, pp. 352-361, 1999.

[3] C. Aggarwal, J. Wolf and P. Yu, "A Framework for the Optimizing of WWW Advertising", *International IFIP Working Conference on Electronic Commerce*, Hamburg, Germany, 1998. Published in *Trends in Distributed Systems for Electronic Commerce*, W. Lamersdorf and M. Merz, editors, Springer-Verlag Lecture Notes in Computer Science, Vol. 1402, pp. 1-10, 1998.

[4] C. Aggarwal, J. Wolf and P. Yu, "A New Method for Similarity Indexing of Market Basket Data", *Proceedings of ACM SIGMOD Conference*, Philadelphia PA, pp. 407-418, 1999.