



How to Develop Online Recommendation Systems that Deliver Superior Business Performance

Executive Summary

Over the past two decades, the Internet has emerged as the mainstream medium for online shopping, social networking, e-mail and more. Corporations also view the Web as a potential business accelerator. They see the huge volume of transactional and interaction data generated by the Internet as R&D that informs the creation of new and more competitive services and products.

Several "e-movement" crusaders have discovered that customers spend significant amounts of time researching products they seek before purchasing. In a bid to assist customers in these efforts, and conserve precious time, these organizations offer users suggestions of products they may be interested in. This serves the dual purpose of not just attracting browsers but converting them into buyers. For instance, an online bookstore may know that a customer has interest in mobile technology based on previous site visits and suggest relevant titles to purchase. An uninitiated user may be impressed by such suggestions. Suggestions (or "recommendations" as they are popularly known) predict likes and dislikes of users. To offer meaningful recommendations to site visitors, these companies need to store huge amounts of data pertaining to different user profiles and their corresponding interests. This eventually culminates in information overload, or difficulty in understanding and making informed decisions. One solution to

combating this issue is what is known as a recommendation system.

Many major e-commerce Websites are already using recommendation systems to provide relevant suggestions to their customers. The recommendations could be based on various parameters, such as items popular on the company's Website; user characteristics such as geographical location or other demographic information; or past buying behavior of top customers.

This white paper presents an overview of how we are helping a Fortune 500 organization to implement a recommendation system. Moreover, this paper also sheds light on key challenges that may be encountered during implementation of a recommendation system built on the open source Apache Mahout,¹ a large-data library of statistical and analytical algorithms.

Recommendation Systems

Recommendation systems can be considered as a valuable extension of traditional information systems used in industries such as travel and hospitality. However, recommendation systems have mathematical roots and are more akin to artificial intelligence (AI) than any other IT discipline. A recommendation system learns from a customer's behavior and recommends a product in which users may be interested. At the heart of recommendation systems are machine-



learning constructs. Leading e-commerce players use recommendation engines that sift users' past purchase histories to recommend products such as magazine articles, books, goods, etc. Here is how major e-commerce companies use recommendation engines to improve their sales and their customers' shopping experience.

- **Amazon.com:** Depending on past purchases and user activity, the site recommends products of user interest.²
- **Netflix:** Recommends DVDs in which a user may be interested by category like drama, comedy, action, etc. Netflix went so far as to offer a \$1 million³ prize to researchers who could improve its recommendation engine.
- **eBay:** Collects user feedback about its products which is then used to recommend products to users who have exhibited similar behaviors.⁴

Online companies that leverage recommendation systems can increase sales by 8% to 12%.⁵

Companies that succeed with recommendation engines are those that can quickly and efficiently turn vast amounts of data into actionable information.

Anatomy of a Recommendation Engine

The key component of a recommendation system is data. This data may be garnered by a variety of means such as customer ratings of products, feedback/reviews from purchasers, etc. This data will serve as the basis for recommendations to users. After data collection, recommendation systems use machine-learning algorithms to find similarities and affinities between products and users. Recommender logic programs are then used to build suggestions for specific user profiles. This technique of filtering the input data and giving recommendations to users is also known as "collaborative filtering."⁶

Along with collaborative filtering, recommendation systems also use other machine-learning techniques such as clustering and classification of data. Clustering is a technique which is used to bundle large amounts of data together into similar categories. It is also used to see data patterns and render huge amounts of data simpler to manage. For instance, Google News⁷ creates clusters of similar news information when grouping diverse arrays of news articles. Many other search engines use clustering to group results for similar search terms.

Classification is a technique used to decide whether new input or a search term matches a previously observed pattern. It is also used to detect suspicious network activity. Yahoo! Mail⁸ uses classification to decide if an incoming message is spam. Image sharing sites like Picasa⁹ use classification techniques to determine whether photos contain human faces. They then offer recommendations of people that are identified in the user contacts list.¹⁰

A Robust System to Counter Information Overload

We are working with a leading multinational manufacturing company that has numerous product research labs with many scientists and researchers in numerous countries working on different technologies. To help facilitate scientific research, and to buy the latest technology information, this client partnered with information providers such as Scopus, Knovel, etc. But despite these data sources, scientists and researchers were often unable to find the right information to improve their research. Also, scientists across the globe were unable to collaborate and share technical information with each other. This situation is typical of companies dealing with information overload.

To increase the informational awareness of scientists and other employees, the client wanted to create a system to recommend resources like patents, articles and journals from paid content providers. A successful system needed to learn from user searches and be intelligent enough to recommend popular resources similar to the ones that a user is currently working from. The system was also expected to provide scientists with useful insights on information other scientists across the globe are using. Finally, the system was to serve as a platform to connect scientists working on similar technologies.

We helped to design and develop the system, which was dubbed "intelligent recommendation system" (IRS). Many of the problems the organization faced originated from the multiple preferences and needs of users pertaining to their individual research topics. To make the system adaptive to specific user requirements, the solution proposed was to use a recommendation system. As a first step towards the solution, the large information base possessed by the client was categorized/grouped by specific criteria. After much contemplation of data and size of the user base, we decided to implement the system using the Apache Mahout framework.

The Mahout framework is highly flexible and lets developers customize outcomes according to their ad hoc requirements. We then developed a customized algorithm to recommend relevant resources to scientists and researchers.

Building an Intelligent Recommendation System

The Solution

The most important purpose of an intelligent recommendation system (IRS) is to increase awareness among scientists about the areas in which they are exploring, technologies on which their colleagues are working, and information about experts and their views on that particular discipline. Despite the possession of huge amounts of data, there was very little insight on the information scientists were seeking. There are fundamental differences designing a recommendation system compared with traditional software design. The overall system architecture depends heavily on the choice of algorithms and the system architecture employed. By using Apache Mahout, the team selected a conventional open source framework that implements machine-learning algorithms.

Apache Mahout is a new Apache Software Foundation (ASF) project whose primary goal is to create scalable machine-learning algorithms that are free to use under the Apache license. The term "Mahout" is derived from the Hindi word that means elephant driver. The Mahout project started in 2008 as a subproject of Apache's Lucene project, which is a popular search engine. Given the amount of data that the client possessed, it was imperative that the IRS be

highly scalable. Mahout is considered a superior way of building recommendation systems because it implements all the three machine learning techniques – collaborative filtering, clustering and classification. Collaborative filtering is the primary technique used by Apache Mahout to provide recommendations. Given rating data along with a set of users and items, collaborative filtering generates recommendations in one of the following four ways.

- **User-based:** Recommendations are made based on users with similar characteristics.
- **Item-based:** Recommendations are based on similar items.
- **Slope-one:** A fast technique that offers recommendations based on previous user-rated items.
- **Model-based:** This approach compares the profile of an active user to aggregate user clusters, rather than the concrete profiles.

There are many algorithms that are used to calculate similarities between two entities. The choice of algorithm plays a vital role in deciding the quality of the recommendation that is most suited for a given scenario. Since the IRS for this client needed to offer recommendations to scientists based on their multiple preferences, we adopted the user-based collaborative filtering (see Figure 1).

Mahout is considered a superior way of building recommendation systems because it implements all the three machine learning techniques – collaborative filtering, clustering and classification.

Recommendation System Execution Flow

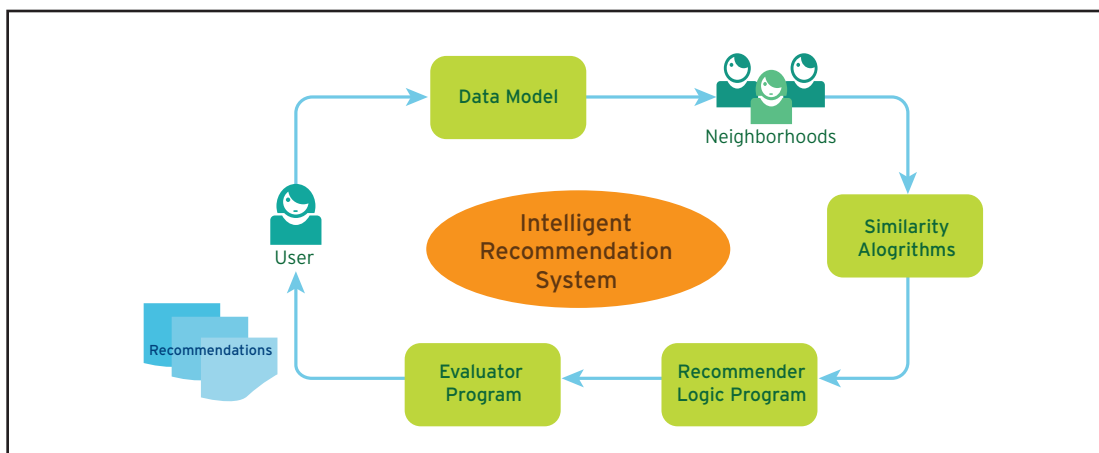


Figure 1

Providing Recommendations

Creating an Input

Within the process of generating recommendations, a crucial step is for the IRS to generate rating and usage data. Such data could be applied to which articles are popular and which ones had the most read counts. To collect and consolidate

Our IRS used a customized recommender logic program which analyses and matches the ratings of one user with similar user ratings and recommends articles/items which match the user interest.

this information, the IRS we used applied the Boolean data model. This technique tracks which article/product was visited by each particular user. This strategy is widely used in “social bookmarking” sites. In the IRS developed for this client, articles were rated between a range of 1 to 2, depending on the number of times an article was visited or read. A decimal number in the range closer to 1 signifies fewer hits while numbers closer to 2 signify higher usage. This rated data is then

grouped into one of the following data models. In this context, data model refers to the abstraction layer which encapsulates product data and corresponding user ratings.

The following are the data models that can be used to create an IRS depending on input data.

- **In-memory data model:** Data is prepared using programs which make objects of the data with product properties like product id, userid and ratings.
- **File-based data:** Data is provided in a comma-separated file along with a preference file. Implementation of a data refresh module will be required to reload data into the comma-separated file.
- **Database-based data:** Relational databases will be a good choice if the data is in the order of gigabytes. However, the implication is that a recommendation engine based on this model will be much slower than in-memory data representations.

For our client, an IRS was created using a file-based data model. This approach delivers superior performance and execution time compared with a database-based model. Also, to sustain the freshness of the file and thereby return newer articles and resources with reduced latency, a scheduled refresh module was used which runs

daily to create the rating data for new resources.

Finding Similarity Between Users/Items

After creating a data model, the next step in building an IRS is finding similarities and patterns between users and items. There are many algorithms which can be used to find item similarities: GenericUserSimilarity, TanimotoCoefficientSimilarity, PearsonCorrelationSimilarity, SpearsonCorrelationSimilarity and EuclideanDistanceSimilarity, to name a few. By far the most common algorithm is TanimotoCoefficientSimilarity as it takes all types of input data, such as binary or numbers. TanimotoCoefficient provides similarity based on the ratings of each item. In the IRS we were building, due to the lack of rating data for all items, our team decided to customize the similarity algorithm based on user demographic properties. The demographic properties such as country, division, job title, department, etc. were used to establish similarity among users. The output of this algorithm is pivotal as it is the base on top of which recommendations are built.

Generating Recommendations

Once similarity is established, special recommender logic programs are used to recommend items to users. This is very similar to user preferences. Our IRS used a customized recommender logic program which analyses and matches the ratings of one user with similar user ratings and recommends articles/items which match the user interest. The output of this recommender logic program are the actual recommendations for the user. In the case of multiple domains such as fields of science (ceramics, nanotechnology, etc.), recommendations might not be accurate since user interests and expertise vary significantly. For such scenarios, recommendations should be evaluated with user domain or user demographics. Programs which aid in cross-verification of the results of recommendation engines are known as “evaluators.” Our IRS validates recommended items with different relevance parameters such as scientist domain, technology and scientist interest. The output of this program proved beneficial in offering concrete recommendations to scientists and researchers and, hence, conserved time.

Key Challenges

The problems faced during development were compounded by the complexity of the algorithms used, as well as the common problems of software design. Apart from these developments, issues unique to AI applications, such as the need

for good quality input data, also arose. Common algorithms such as Tanimoto and Pearson similarity have several advantages, such as the ability to detect the quality and features of an item at the time of recommendation, especially when a user rating on a scale of 1 to 5 is available. Another advantage is that these algorithms are useful in domains where analyzing data is difficult or costly, such as gathering data about science articles where domain knowledge is required to rate an article.

The first challenge in the development of our IRS is that the system was brand new and lacked user rating information. Huge amount of input data, but few users and no ratings is referred to as "Cold Start Problem." A solution to this problem was to use the TanimotoCoefficient¹¹ algorithm and binary dataset to feed the system with the much-sought-after data during the initial use of the system. In the course of time, user rating data will be obtained after which the similarity algorithm and input data structure can be changed. This initial dataset can be prepared by collecting data such as maximum hits of a particular product or tracking visits of articles. If a binary dataset is adopted then most suitable algorithms are TanimotoCoefficientSimilarity or LogLikelihoodSimilarity¹² algorithms, since these are specially defined for binary data and use the Jaccard¹³ coefficient formula. As the usage of the recommendation system grows, huge amounts of data will be collected. It is advisable at this point to use clustering mechanisms to sustain performance. A solution to deal with massive data by means of data clustering is Hadoop.¹⁴ Data clustering requires multiple machines to handle massive data sets and stores all resources that are required to provide recommendations.

The next challenge is providing recommendations to a user who is not part of a clear group; these users can be called "grey sheep." The challenge of giving recommendations to new or unknown users is called the "grey sheep problem." This problem mostly arises in Internet applications with large and diverse customer bases that access the IRS application. A solution to this problem is to look for similar users based on similar demographic information or characteristics. These users can be technically called "neighbors." To determine similarity, a "classification" machine-learning mechanism can be used. Similar neighbors can be found by implementing different similarity algorithms or by checking similar user properties. The probability of this problem is significantly lower in intranet applications. The classification algorithm trains the computer to classify similar users in a particular group and teaches the IRS to find the specific group when a user is new to the system.

Mahout currently supports two types of classifier approaches: Naive Bayes classifier and Complementary Naive Bayes classifier. Naive's classifier is the faster method for the problems such as grey sheep recommendations. This classifier includes two parts. First, it analyzes the existing documents or data and finds the common features among them. This method is also called as preparing training examples. Secondly, classification is performed using the model that was created in the first step using the Bayes theorem¹⁵ to predict the category of a new item (see Figure 2).

Data clustering requires multiple machines to handle massive data sets and stores all resources that are required to provide recommendations.

Diagram of Classification Technique

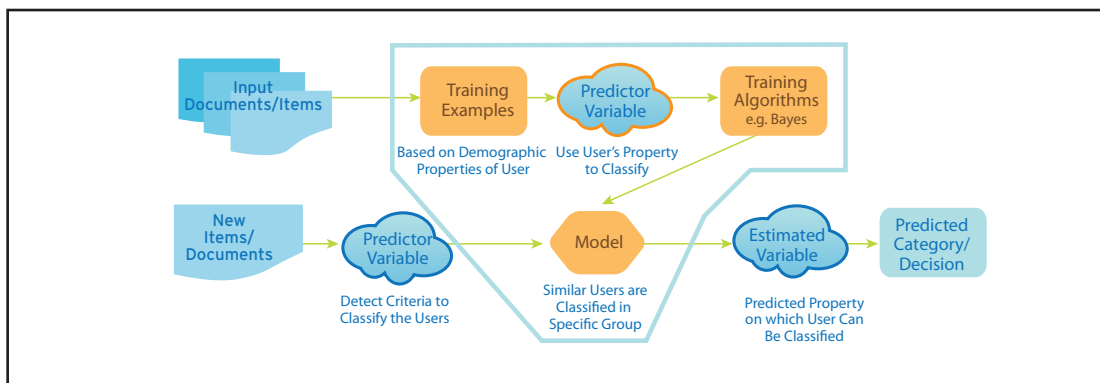


Figure 2

Finally, sometimes providing recommendations to users only on product ratings will not be accurate; problems may arise when the recommended items originate from another domain to which the user is not related. This problem is known as a “blind recommendation” problem. This problem is mostly prevalent in the post-production phase of recommendation systems development. A solution to this problem is to use evaluators that continuously verify and validate recommendations with actual user details.

Conclusion

Online businesses are working very hard to increase their customer bases by providing competitive prices, products and services. In this frantic effort, recommendation systems

can be an important means of bolstering sales and increasing information awareness among customers.

In this paper, we examined how recommendation systems can be applied to the scientific research domain. The successful implementation of an IRS provided our client with flexibility in using pre-existing information resources. It solved the problem of information overload by enabling scientists and researchers to use information resources more efficiently for their research. Our IRS offers testimony that a recommendation system that provides customizable recommendations can enable online companies to perform business more effectively.

Footnotes

- ¹ <http://mahout.apache.org/>
- ² <https://cwiki.apache.org/MAHOUT/mahout-on-elastic-mapreduce.html> as of August 2011
- ³ <http://www.netflixprize.com/>
- ⁴ <http://developer.ebay.com/DevZone/xml/docs/Reference/ebay/AddSellingManagerTemplate.html> as of August 2011
- ⁵ <http://www.practicalecommerce.com/articles/1942-10-Questions-on-Product-Recommendations> as of August 2011
- ⁶ http://en.wikipedia.org/wiki/Collaborative_filtering
- ⁷ <http://www2007.org/papers/paper570.pdf>
- ⁸ http://www.manning.com/owen/Mahout_MEAP_CH01.pdf as of August 2011
- ⁹ <http://blogoscoped.com/archive/2007-03-12-n67.html>
- ¹⁰ http://news.cnet.com/8301-27076_3-10358662-248.html
- ¹¹ http://en.wikipedia.org/wiki/Jaccard_index
- ¹² http://en.wikipedia.org/wiki/Likelihood-ratio_test
- ¹³ http://en.wikipedia.org/wiki/Jaccard_index
- ¹⁴ <http://hadoop.apache.org/>
- ¹⁵ http://en.wikipedia.org/wiki/Naive_Bayes_classifier

References:

Mahout: <http://www.ibm.com/developerworks/opensource/library/j-mahout/index.html>
<http://ilk.uvt.nl/~toine/phd-thesis/phd-thesis.pdf>
GroupLens Research Project: <http://www.grouplens.org/papers/pdf/ec-99.pdf>
http://en.wikipedia.org/wiki/Recommender_system

About the Authors

Anup Prakash Warade is a Cognizant Associate. He develops and supports multiple projects for key clients in the manufacturing and logistics industry. His interests include designing and developing Java EE Web applications. Anup holds a bachelor's of engineering degree in electronics from the University of Pune in India. He can be reached at Anup.Warade@cognizant.com.

Vignesh Murali Natarajan is a Cognizant Senior Associate. He is a technical trainer and a Java architect who is currently managing several Java EE applications for a manufacturing client. Vignesh holds a bachelor's degree in computer science and engineering from the University of Madras. He can be reached at Vigneshmurali.Natarajan@cognizant.com.

Siddharth Sharad Chandak is a Cognizant Project Manager overseeing numerous business-critical projects for clients in the manufacturing and logistics industry. His interests include architecting and designing enterprise software applications, software performance tuning and project management. Siddharth holds a master's of science degree in computer science from Kansas State University. He can be reached at Siddharth.Chandak@cognizant.com.

About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world's leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 130,000 employees as of September 30, 2011, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at www.cognizant.com or follow us on Twitter: Cognizant.



World Headquarters

500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277
Email: inquiry@cognizant.com

European Headquarters

1 Kingdom Street
Paddington Central
London W2 6BD
Phone: +44 (0) 20 7297 7600
Fax: +44 (0) 20 7121 0102
Email: infouk@cognizant.com

India Operations Headquarters

#5/535, Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060
Email: inquiryindia@cognizant.com