An Online Recommendation System for E-commerce Based on Apache Mahout Framework

Mr. Sachin Walunj
Information Technology Department
Sinhgad College of engineering, vadgaon, pune, India
[1]sachinwalunj2008@gmail.com

Mr.Kishor Sadafale
Information Technology Department
Sinhgad College of engineering, vadgaon, pune, India
[2]kishor_sadafale@yahoo.com

ABSTRACT

Selecting a foundational platform is an important step in developing recommender systems for personal, research, or commercial purposes. This can be done in many different ways the platform may be developed from the ground up, an existing recommender engine may be contracted (OracleAS Personalization), code libraries can be adapted, or a platform may be selected and tailored to suit (LensKit, MymediaLite, Apache Mahout, etc.). In some cases, a combination of these approaches will be employed. For E-commerce projects, and particularly in the E-commerce website t, the ideal situation is to find an open-source platform with many active contributors that provides a rich and varied set of recommender system functions that meets all or most of the baseline development requirements. Short of finding this ideal solution, some minor customization to an already existing system may be the best approach to meet the specific development requirements. Various libraries have been released to support the development of recommender systems for some time, but it is only relatively recently that larger scale, open-source platforms have become readily available. In the context of such platforms, evaluation tools are important both to verify and validate baseline platform functionality, as well as to provide support for testing new techniques and approaches developed on top of the platform. Apache Mahout as an enabling platform for research and have faced both of these issues in employing it as part of work in collaborative filtering recommenders.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *Information Filtering*

General Terms

Algorithms, Performance, Experimentation

Keywords

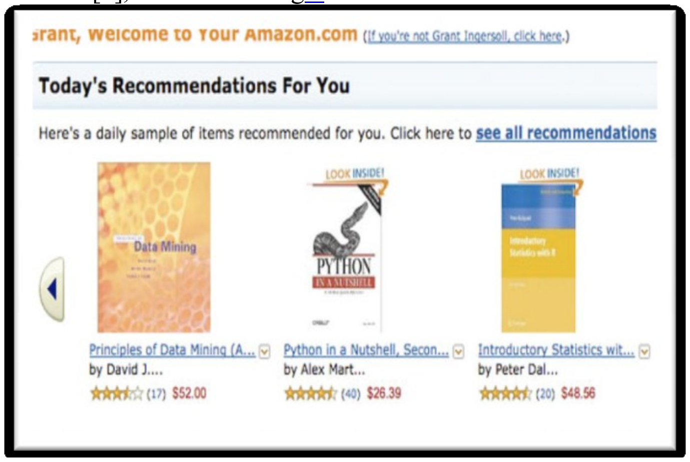
Personalized recommendation, apache mahout framework, recommendation system, collaborative filtering.

1. INTRODUCTION

In everyday life, people rely on recommendations from other people by spoken words, reference letters, and news reports from news media, general surveys, travel guides, and so forth. Recommender systems assist and augment this natural social process to help people sift through available books, articles, webpages, movies, music, restaurants, jokes, grocery products, and so forth to find the most interesting and valuable information for them[1][2]. The developers of one of the first recommender systems, Tapestry (other earlier recommendation systems include rule-based recommenders and user-customization), coined the phrase "collaborative filtering (CF)," which has been widely adopted regardless of the facts that recommenders may not explicitly collaborate with recipients and recommendations may suggest particularly interesting items, in addition to indicating those that should be filtered out.

Collaborative Filtering

Collaborative filtering (CF) is a technique, popularized by Amazon and others, that uses user information such as ratings, clicks, and purchases to provide recommendations to other site users. CF is often used to recommend consumer items such as books, music, and movies, but it is also used in other applications where multiple actors need to collaborate to narrow down data. CF in action on Amazon[3], as shown in Fig.1


rant, Welcome to Your Amazon.com (If you're not Grant Ingersoll, click here.)

Today's Recommendations For You

Here's a daily sample of items recommended for you. Click here to see all recommendations

Principles of Data Mining (A... by David J.... (17) $52.00

Python in a Nutshell, Secon... by Alex Mart... (40) $26.39

Introductory Statistics wit... by Peter Dal... (20) $48.56

Figure 1. Example of collaborative filter on Amazon

1. **User-based**: Recommend items by finding similar users. This is often harder to scale because of the dynamic nature of users.
2. **Item-based**: Calculate similarity between items and make recommendations. Items usually don't change much, so this often can be computed offline.
3. **Slope-One**: A very fast and simple item-based recommendation approach applicable when users have given ratings (and not just Boolean preferences).
4. **Model-based**: Provide recommendations based on developing a model of users and their ratings.

All CF approaches end up calculating a notion of similarity between users and their rated items. There are many ways to compute similarity, and most CF systems allow you to plug in different measures so that you can determine which one works best for your data [3].
Collaborative Filtering algorithm is a classic personalized recommendation algorithm; it's widely used in many commercial recommender systems. Collaborative filtering algorithm is an algorithm based on the following assumptions idea, People have similar preferences and interests their preferences and interests are stable we can predict their choice according to their past preferences. Because of the above assumptions, the collaborative filtering algorithm is based on the comparison of one user's behavior with other user's behavior, to find his nearest neighbors, and according to his neighbor's interests or preferences to predict his interests or preferences.

The first step of collaborative filtering algorithm is to obtain the users history profile, which can be represented as a ratings matrix with each entry the rate of a user given to an item. A ratings matrix consists of a table where each row represents a user, each column represents a specific product, and the number at the intersection of a row and a column represents the user's rating value. The absence of a rating score at this intersection indicates that user has not yet rated the item. Owing to the existence problem of sparse scoring, use the list to replace the matrix. The second step is to calculate the similarity between users and find their nearest neighbors. There are many similarity measure methods. The Pearson correlation coefficient is the most widely used and served as a benchmark for CF. Generally use the Cosine similarity measure method[4][5].

2. CHARACTERISTICS AND CHALLENGES OF COLLABORATIVE FILTERING

2.1 Data Sparsity

In practice, many commercial recommender systems are based on large datasets. As a result, the user-item matrix used for collaborative filtering could be extremely large and sparse, which brings about the challenges in the performances of the recommendation.

One typical problem caused by the data sparsity is the cold start problem. As collaborative filtering methods recommend items based on users' past preferences, new users will need to rate sufficient number of items to enable the system to capture their preferences accurately and thus provides reliable recommendations.

Similarly, new items also have the same problem. When new items are added to system, they need to be rated by substantial number of users before they could be recommended to users who have similar tastes with the ones rated them. The new item problem does not limit the content-based recommendation, because the recommendation of an item is based on its discrete set of descriptive qualities rather than its ratings[5].

2.2 Scalability

When numbers of existing users and items grow tremendously, traditional CF algorithms will suffer serious scalability problems, with computational resources going beyond practical or acceptable levels. For example, with tens of millions of customer and millions of distinct catalog items, a CF algorithm with the complexity is already too large. As well, many systems need to react immediately to online requirements and make recommendations for all users regardless of their purchases and ratings history, which demands a high scalability of a CF system.

2.3 Synonyms

Synonymy refers to the tendency of a number of the same or very similar items to have different names or entries. Most recommender systems are unable to discover this latent association and thus treat these products differently For example, the seemingly different items "children movie" and "children film" are actually referring to the same item. Indeed, the degree of variability in descriptive term usage is greater than commonly suspected. The prevalence of synonyms decreases the recommendation performance of CF systems. Topic modeling (like the Latent Dirichlet Allocation technique) could solve this by group different words belonging to the same topic.

2.4 Grey Sheep

Grey sheep refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering. Black sheep are the opposite group whose idiosyncratic tastes make recommendations nearly impossible. Although this is a failure of the recommender system, non-electronic recommenders also have great problems in these cases, so black sheep is an acceptable failure.

2.5 Shilling attacks

In a recommendation system where everyone can give the ratings, people may give lots of positive ratings for their own items and negative ratings for their competitors. It is often necessary for the collaborative filtering systems to introduce precautions to discourage such kind of manipulations.

2.6 Diversity

Collaborative filters are expected to increase diversity because they help us discover new products among myriad choices. Some algorithms, however, may unintentionally do the opposite. Because collaborative filters recommend products based on sales or ratings, they cannot usually recommend products with limited historical data. This creates a rich-get-richer effect for popular products, leading to less diversity. A Wharton study details this phenomenon and several ideas which may promote diversity in collaborative filtering recommendations.

3. APACHE MAHOUT SOLUTION

To support research in collaborative filtering, several recommender system platforms were surveyed, including LensKit, easyrec5, and MymediaLite. Apache Mahout provides many of the desired characteristics required for a recommender development workbench platform. Mahout is a production-level, open-source, system and consists of a wide range of applications that are useful for a recommender system developer collaborative filtering algorithms, data clustering, and data classification. Mahout is also highly scalable and is able to support distributed processing of large data sets across clusters of computers using Hadoop. Mahout recommenders support various similarity and neighborhood formation calculations, recommendation prediction algorithms include user-based, item-based, Slope-One and Singular Value Decomposition (SVD), and it also incorporates Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) evaluation methods. Mahout is readily extensible and provides a wide range of Java classes for customization. As an open-source project, the Mahout developer/contributor community is very active; the Mahout wiki also provides a list of developers and a list of websites that have implemented Mahout[4][7].

3.1 Taste Architecture:

Taste is the Recommender System part of Mahout and it provides a very consistent and flexible collaborative filtering engine. It supports the user-based, item-based and Slope-one recommender systems. It can be easily modified in due to its well-structured modules abstractions. The package defines the following interfaces:

1. DataModel
2. UserSimilarity and ItemSimilarity
3. UserNeighborhood
4. Recommender

With these interfaces, it's possible to adapt the framework to read different types of data, personalize your recommendation or even create new recommendation methods. Below is presented a figure with Taste's architecture.

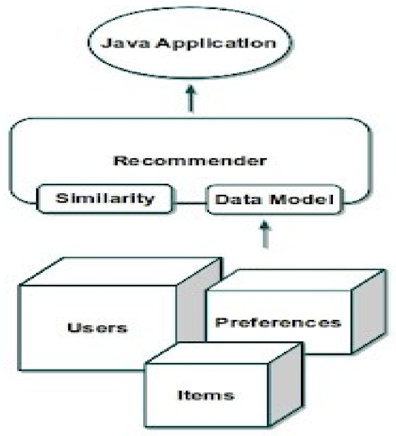


Figure 2. Taste Architecture

The User Similarity and Item Similarity abstractions are here represented by the box named "Similarity". These interfaces are responsible for calculating the similarity between a pair of users or items. Their function usually returns a value from 0 to 1 indicating the level of resemblance, being 1 the most similar possible. Through the DataModel interface is made the access to the data set. It is possible to retrieve and store the data from databases or from filesystems (MySQLJDBCDataModel and FileDataModel respectively). The functions developed in this interface are used by the Similarity abstraction to help computing the similarity.

The main interface in Taste is Recommender. It is responsible for actually making the recommendations to the user by comparing items or by determining users with similar taste (item-based and user-based techniques). The Recommenderaccess the similarity interface and uses its functions to compare a pair of users or items. It then collect the highest similarity values to offer as recommendations.

The UserNeighborhood is an assistant interface to help defining the neighborhood into the User-Based recommendation technique.

It is known that for greater data sets the item-based technique provides better results. For that, many companies choose to use this approach, such as Amazon. With the Mahout framework it's not different; the item-based method generally runs faster and provides more accurate recommendation.

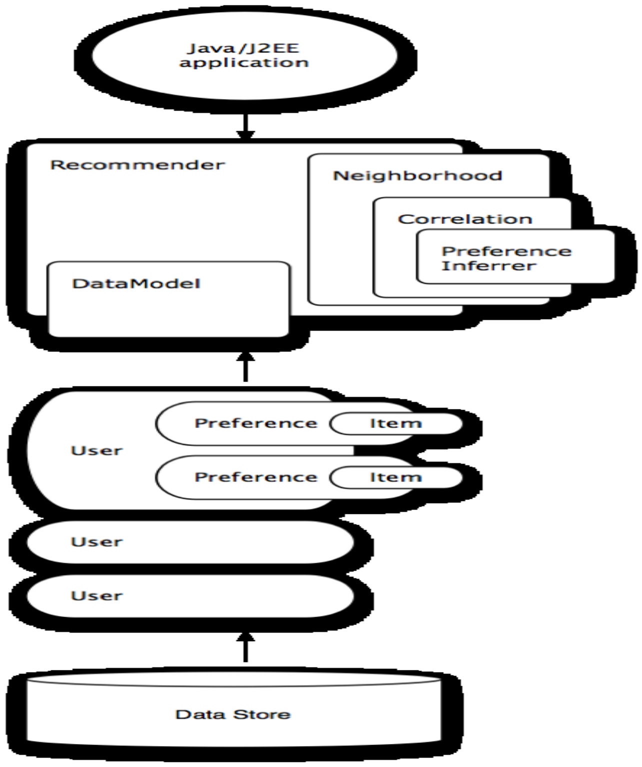


Figure 3. Mahout Architecture

4. AN INTELLIGENT RECOMMENDATION SYSTEM

The main purpose of a recommendation engine is to make inferences on existing data to show relationships between objects. Objects can be many things, including users, items, products, and so on. Relationships provide a degree of likeness or belonging between objects. For example, relationships can represent ratings of how much a user likes an item (scalar), or indicate if a user bookmarked a particular page (binary).

To make a recommendation, recommendation engines perform several steps to mine the data. Initially, begin with input data that represents the objects as well as their relationships. Input data consists of object identifiers and the relationships to other objects. Figure shows this at a high level.

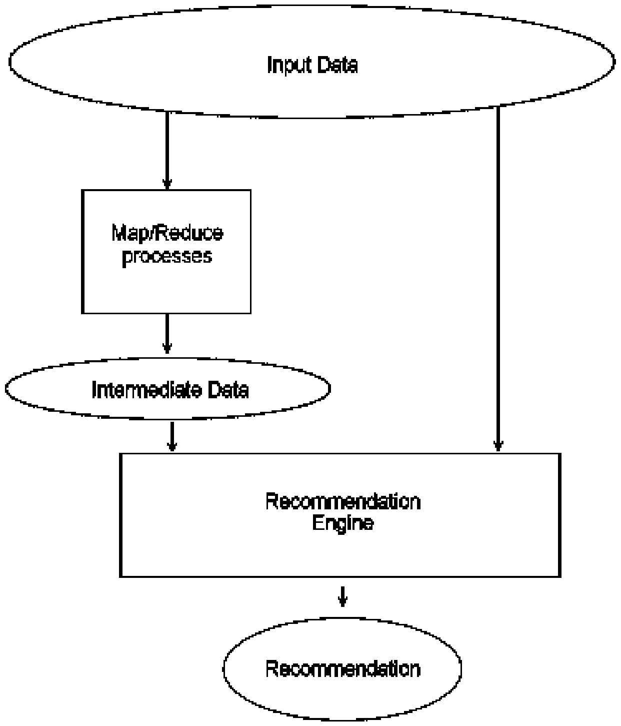


Figure 4. Intelligent Recommendation System

Consider the ratings users give to items. Using this input data, a recommendation engine computes a similarity between objects. Computing the similarity between objects can take a great deal of time depending on the size of the data or the particular algorithm. Distributed algorithms such as Apache Hadoop can be used to parallelize the computation of the similarities. There are different types of algorithms to compute similarities. Finally, using the similarity information, the recommendation engine can make recommendation requests based on the parameters requested.

It could be that believed recommendation engines were helpful, but stayed away because thought they were too complicated to try. The recommendation engine domain is in fact large and can be very complex.

Collaborative filtering is an easy and popular technique. It's easy because customers do the important work they drive the criteria of what to highlight. Collaborative filtering analyzes ratings from other users or items to make recommendations. There are two approaches within collaborative filtering: the main difference between them lies in the ability of each to scale as the number of users in the system grows

4.1 User-based recommendation

This type of recommendation builds similarities between users by looking at the commonalities of the items rated by each user.

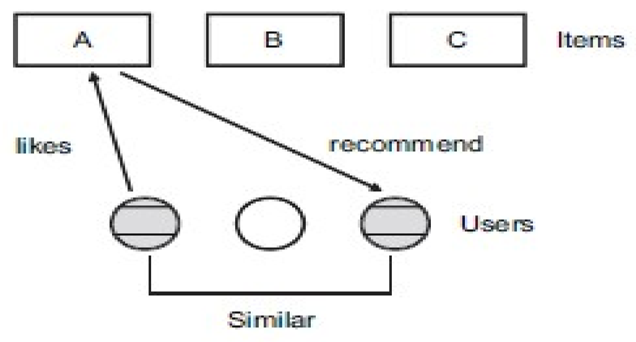


Figure 5. User –based recommendation

If a user likes item A, then the same item can be recommended to other users who are similar to user A. For example, if the items are courses, two users could be considered very similar if they both took the same courses. In the other extreme, their similarity would be low if they did not take any similar course. To make recommendations, the algorithms rely on the ratings that similar users gave to those courses not taken by the user. This recommendation is the most basic one however, its main limitation is that in order to generate the similarities, it needs to compare each user to every other user. This is acceptable for an application with a low number of users, but if the number of users increases, the time to perform this evaluation increases exponentially[9].

4.2 Item-based recommendation

Item-based recommendation, on the other hand, begins by looking at the items that are associated with the user.

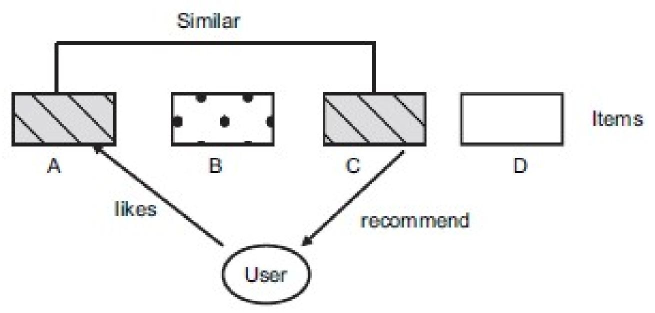


Figure 6. Item-based recommendation

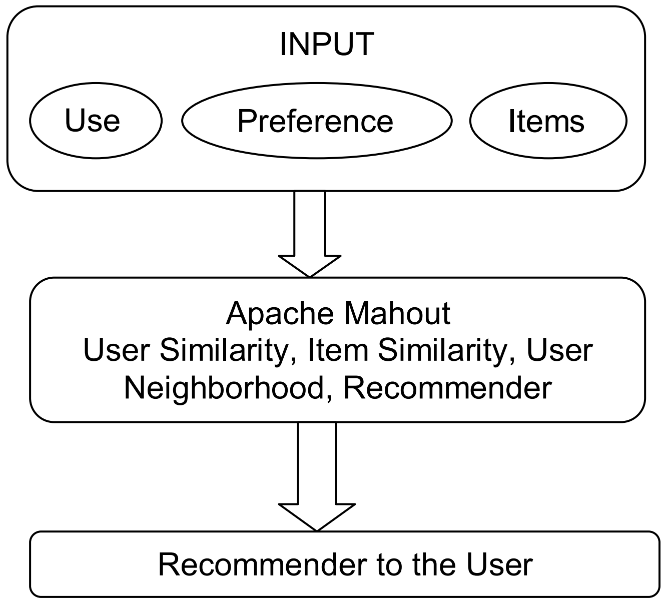
If items A and C are highly similar and a user likes item A, then item C is recommended to the user. For each item associated with the user, the algorithm computes how similar it is to the other items in the collection to build the list of recommendations. In order to determine how likely the user is to like a recommended item, the algorithm looks at the ratings that the user has given to the item and gives a weighted rating to each recommended item. The main issue with item-based recommendation is that it needs to build a similarity index for every available item. Changes in the items, however, are less frequent than changes in users and, therefore, it is feasible with this type of recommendation to pre-compute similarities offline and update them at specific periods [9].

5. BUILDING A RECOMMENDATION ENGINE

Mahout currently provides tools for building a recommendation engine through the Taste library a fast and flexible engine for CF. Taste supports both user-based and item-based recommendations and comes with many choices for making recommendations, as well as interfaces for your own. Taste consists of five primary components that work with Users, Items and Preferences:

1. DataModel: Storage for Users, Items, and Preferences
2. UserSimilarity:Interface defining the similarity between two users
3. ItemSimilarity: Interface defining the similarity between two items
4. Recommender:Interface for providing recommendations
5. UserNeighborhood:Interface for computing a neighborhood of similar users that can then be used by the Recommenders

These components and their implementations make it possible to build out complex recommendation systems for either real-time-based recommendations or offline recommendations. Real-time-based recommendations often can handle only a few thousand users, whereas offline recommendations can scale much higher. Taste even comes with tools for leveraging Hadoop to calculate recommendations offline. In many cases, this is a reasonable approach that allows to meet the demands of a large system with a lot of users, items, and preferences[4][8].



INPUT
Use
Preference
Items
Apache Mahout
User Similarity, Item Similarity, User Neighborhood, Recommender
Recommender to the User

Figure 7. Recommendation engine using apache Mahout

7. CONCLUSION

Online businesses are working very hard to increase their customer bases by providing competitive prices, products and services. In this effort, recommendation systems can be an important means of increasing sales and information awareness among customers. Item based recommendation provides the better result than the user based recommendation. The successful implementation of a mahout framework provided flexibility in using pre-existing algorithms. It solved the problem of scalability by using the hadoop platform because it is built on the hadoop framework. Apache mahout offers testimony that a recommendation system provides customizable recommendations enable online companies to perform business more effectively.

8. REFERENCES

[1] Carlos E. Seminario,DavidC.Wilson. "Case study evaluation of mahout as a recommender platform".ACM 2012.

[2] Zhi-Dan Zhao, Ming-Sheng Shang."User-based Collaborative-Filtering Recommendation Algorithm on Hadoop". third International conference on knowledge Discovery and data mining 2012.

[3] Greg Linden, Brent Smith, and Jeremy York," Amazon.com Recommendations Item-to-Item Collaborative Filtering"Published by the IEEE Computer Society 1089-7801/03/$17.00©2003.

[4] RuiMaximoEsteves, ChunmingRong."Using Mahout for clustering Wikipedia's latest articles- A comparison between k-means and fuzzy c-means" third international conference on cloud computing Technology and science 'In the cloud 2011.

[5] C. Desrosiers and G. Karypis. "A comprehensive survey of neighborhood-based recommendations methods". In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor,editors, Recommender Systems Handbook. Springer, 2011.

[6] M. D. Ekstrand, M. Ludwig, J. A. Konnstan, and J. T. Riedl. "Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit". In Proceedings of the 5th ACM Recommender Systems Conference (RecSys '11), October 2011.

[7] M. Ge, C. Delgado-Battenfeld, and D. Jannach. "Beyond accuracy: Evaluating recommender systems by coverage and serendipity". In Proceedings of the 4th ACM Recommender Systems Conference (RecSys '10), September 2010.

[8] https://cwiki.apache.org/MAHOUT/recommender-documentation.html

[9] SatnamAlag, "Collective Intelligence In Action", Manning Publications Co., first edition, 2009.

[10] Shine Ge, Xinyang Gen "An SVD-based Collaborative Filtering Approach toAlleviate Cold-Start Problems",In 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012)

[11] Min Xiao, BingjieYan,"Collaborative filtering recommendation algorithm based on shift of users' preferences" IEEE 2010.

[12] SutheeraPuntheeranurak, ThanutChaiwitooanukool,"An Item-based Collaborative Filtering Method using Item-based Hybrid Similarity"IEEE 2011.