RECOMMENDATION METHOD ON HADOOP AND MAPREDUCE FOR BIG DATA APPLICATIONS

T.M.S.MEKALARANI, M.KALAIVANI

# ME, Computer Science and Engineering, Dhanalakshmi College of Engineering, Tambaram, India.
* ME, Computer Science and Engineering Asst Prof of Dhanalakshmi College of Engineering, Tamabaram, India.

Abstract

In recent years, the amount of data in our world has been increasing explosively, and analyzing large data sets the so-called “Big Data” becomes a key basis of competition underpinning new waves of productivity growth, innovation, and consumer surplus. Then, what is “Big Data”? Big Data refers to datasets whose size is beyond the ability of current technology, method and theory to capture, manage, and process the data within a tolerable elapsed time. With the growing number of alternative services, effectively recommending services that a user preferred has become an important research issue. Service recommender systems have been shown as valuable tools for providing appropriate recommendations to users. In the last decade, the amount of customers, services and online information has grown rapidly, yielding the big data analysis problem for service recommender systems. Consequently, traditional service recommender systems often suffer from scalability and inefficiency problems when processing or analyzing such large-scale data. Moreover, most of existing service recommender systems present the same ratings and rankings of services to different users without considering diverse users’ preferences, and therefore fails to meet users’ personalized requirements.

Key words: Recommendation, Scalability, Inefficiency, Big Data, Hadoop, MapReduce.
INTRODUCTION

Current recommendation methods usually can be classified into three main categories: content-based, collaborative, and hybrid recommendation approaches. Content-based approaches recommend services similar to those the user preferred in the past. Collaborative filtering (CF) approaches recommend services to the user that users with similar tastes preferred in the past. Hybrid approaches combine content-based and CF methods in several different ways. In CF based systems, users receive recommendations based on people who have similar tastes and preferences, which can be further classified into item-based CF and user-based CF. In item-based systems; the predicted rating depends on the ratings of other similar items by the same user. While in user-based systems, the prediction of the rating of an item for a user depends upon the ratings of the same item rated by similar users. And in this work, we will take advantage of a user-based CF algorithm to deal with our problem. The one example of big data analysis problem Alice and Tom are respectively browsing a hotel reservation website to reserve a hotel in Kowloon, Hong Kong. But the ratings and recommendation list of the hotels provided by the website to them are the same. Assuming there are three hotels in Kowloon: A, B and C. Comparing the three hotels, A is convenient to the airport and has a shopping mall nearby; B has convenient transportation with an underground station nearby and owns comfortable accommodation equipment; the breakfast and food of C is delicious and its view is very good. According to the overall ratings provided by the website, B is better than A and A is better than C. However, in this travel, Alice prefers a shopping mall near the hotel and good location, while Tom is concerned about food and wishes good view around the hotel. So hotel B may not be the best choice for them, and hotel A and C may be more appropriate to Alice and Tom respectively. Over the last two and a half years we have designed, implemented, and deployed a distributed storage system for managing structured data at Google called Bigtable. Bigtable is designed to reliably scale to petabytes of data and thousands of machines. Bigtable has achieved several goals: wide applicability, scalability, high performance, and high availability. Bigtable is used by more than sixty Google products and projects, including Google Analytics, Google Finance, Orkut, Personalized Search, Writely, and Google Earth. These products use Bigtable for a variety of demanding workloads, which range from throughput-oriented batch-processing jobs to latency-sensitive serving of data to end users. The Bigtable clusters used by these products span a wide range of con_gurations, from a handful to
thousands of servers, and store up to several hundred terabytes of data. In many ways, Bigtable resembles a database: it shares many implementation strategies with databases. Parallel databases and main-memory databases have achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of the data represented in the underlying storage. Data is indexed using row and column names that can be arbitrary strings. Bigtable also treats data as uninterrupted strings, although clients often serialize various forms of structured and semi-structured data into these strings. Clients can control the locality of their data through careful choices in their schemas.

RELATED WORK

Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the user’s purchased and rated items. The algorithm aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommends the remaining items to the user. Two popular versions of these algorithms are collaborative filtering and cluster models. Other algorithms — including search-based methods and our own item-to-item collaborative filtering — focus on finding similar items, not similar customers. For each of the user’s purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them. Traditional Collaborative Filtering A traditional collaborative filtering algorithm represents a customer as an N-dimensional vector of items, where N is the number of distinct catalog items. The components of the vector are positive for purchased or positively rated items and negative for negatively rated items. To compensate for best-selling items, the algorithm typically multiplies the vector components by the inverse frequency (the inverse of the number of customers who have purchased or rated the item), making less well-known items much more relevant. For almost all customers, this vector is extremely sparse. The algorithm generates recommendations based on a few customers who are most similar to the user. It can measure the similarity of two customers, A and B, in various ways; a common method is to measure the cosine of the angle between the two vectors. The algorithm can select recommendations from the similar customers’ items using various methods as well, a common technique is to rank each item according to how many similar
customers purchased it. Using collaborative filtering to generate recommendations is computationally expensive. It is \( O(MN) \) in the worst case, where \( M \) is the number of customers and \( N \) is the number of product catalog items, since it examines \( M \) customers and up to \( N \) items for each customer. However, because the average customer vector is extremely sparse, the algorithm’s performance tends to be closer to \( O(M + N) \). Scanning every customer is approximately \( O(M) \), not \( O(MN) \), because almost all customer vectors contain a small number of items, regardless of the size of the catalog. But there are a few customers who have purchased or rated a significant percentage of the catalog, requiring \( O(N) \) processing time. Thus, the final performance of the algorithm is approximately \( O(M + N) \). Even so, for very large data sets — such as 10 million or more customers and 1 million or more catalog items — the algorithm encounters severe performance and scaling issues. It is possible to partially address these scaling issues by reducing the data size. We can reduce \( M \) by randomly sampling the customers or discarding customers with few purchases, and reduce \( N \) by discarding very popular or unpopular items. It is also possible to reduce the number of items examined by a small, constant factor by partitioning the item space based on product category or subject classification. Dimensionality reduction techniques such as clustering and principal component analysis can reduce \( M \) or \( N \) by a large factor.

CONCLUSION

Collaborative Filtering algorithm is adopted to generate appropriate recommendations. More specifically, a keyword-candidate list and domain thesaurus are provided to help obtain users' preferences. The active user gives his/her preferences by selecting the keywords from
the keyword-candidate list, and the preferences of the previous users can be extracted from their reviews for services according to the keyword-candidate list and domain thesaurus. Our method aims at presenting a personalized service recommendation list and recommending the most appropriate service(s) to the users. Moreover, to improve the scalability and efficiency of KASR in “Big Data” environment, we have implemented it on a MapReduce framework in Hadoop platform. Finally, the experimental results demonstrate that KASR significantly improves the accuracy and scalability of service recommender systems over existing approaches. In our future work, we will do further research in how to deal with the case where term appears in different categories of a domain thesaurus from context and how to distinguish the positive and negative preferences of the users from their reviews to make the predictions more accurate.

REFERENCE


