International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 10, Issue 6 (June 2014), PP.49-54

Survey on Recommender systems

Pallavi K. Deshpande[1], Chitrakant Banchhor [2]
Department of Information Technology

Abstract:- Social networks have become a great source of information, for that several applications have been proposed to extract information from social networks such as: recommender systems. The speed and scalability of such a recommender algorithm is as important as the actual logic behind the algorithm because such algorithms generally run over a "huge" graph and implementing these normally would probably take a lot of time for recommending items even for just one user. The basic idea of the algorithm is to recommend items to users connected via social network. In this paper various categories in which recommender systems are classified are discussed. Also various open sources, scalable, distributed and non distributed graph processing platforms are also discussed. The concept of recommender system with Giraph which is an open source distributed graph processing platform for highly reliable, scalable graph datasets is introduced in this paper.

Keywords:- large scale graph, recommender system, giraph, hadoop, bulk synchronous parallel model, recommendation algorithm, and social network analysis.

I. INTRODUCTION

It is known to everyone that the information available on the web increases rapidly with time. As a result human beings are not able to efficiently understand, exploit or even handle such a large amount of information. The concept of recommender system is thus introduced to overcome this information overhead, by filtering the information, so that users can easily make choices according to their interests.

The aim of Recommender Systems is providing personalized recommendations. They help users by suggesting useful items to them, usually dealing with enormous amounts of data. Recommender systems are mainly classified into three categories: content-based, collaborative filtering or social and hybrid.

Many recommender systems approaches have been developed in the past decade, but a considerable amount of them were constructed and evaluated with small datasets. Furthermore, the volume of web information has greatly increased in the last years, and for that, several recommender systems suffer from performance and scalability problems when dealing with larger datasets.

Moreover, in recent years, social networks like Facebook, twitter, MySpace are common trends where worldwide users are connected together. These social networks are becoming the best source of information regarding user's interest.

Graph is an abstract data structure with numerous applications. Social networks can be effectively expressed using graph data structure, in which vertices represent people and edges represent social interactions between these people. Social networks are usually a large graphs usually having millions of vertices and billions of edges.

Giraph is a new rising large-scale graph processing platform which follows the Google's Pregel system and could be seen as a variant of the Bulk Synchronous Parallel (BSP) Model. Giraph is built as a layer on top of Hadoop to leverage its parallelism and cluster management capabilities. Giraph overcomes the inefficiencies of MapReduce framework while running an iterative graph algorithm on Hadoop.

This paper focuses on providing the overview about the various recommender techniques developed or proposed till now. Various categories in which recommender systems can be classified are discussed with examples. Also various open source graph processing platforms are discussed in detail.

The rest paper is organized as follows: Section II briefs about the work done till now related to this area. Section III describes the various categories of recommender systems. Section IV describes the graph processing platforms. Two parallel processing models mainly used with open source graph processing platforms

are discussed and compared in section V. Section VI describes various recommendation algorithms in short. Section VII concludes this paper.

II. RELATED WORK

This section will briefly review the related work on social recommender systems and semantic social recommender systems.

Proposed in 1994 by Grouplens[10], Newsnet is one of the earliest collaborative filtering recommender systems. Newsnet is a user-based collaborative filtering, based on the Pearson r correlation coefficient. Ringo[11] is another collaborative filtering recommender system, which uses personalization to recommend music and artists. In 1999, IRA [1] (Intelligent Recommendation Algorithm) was proposed as a graph-based collaborative filtering recommender system, uses a breadth-first search algorithm to compute the shortest paths between graph vertices (users). Moreover, user-item bipartite graph and one-mode projection are used in a movie recommender system. In this system a recommendation graph is defined as the sum of the bipartite graph and the one-mode projection graph, then a shortest path algorithm is applied on the recommender graph in order to compute the recommendation. In trustwalker [1], a random walk algorithm is proposed to recommend items in a trust network. In this algorithm items are recommended based on ratings expressed by trusted friends, using probabilistic item selection and random walk.Other recommender systems include semantic aspects, in addition to collaborative filtering aspects. A recommendation algorithm is also introduced for collaborative URL tagging [1]. In this system, user interests are modeled according to their social ties and the vocabularies they use to tag URLs. An algorithm is also proposed in which similar tags are grouped in clusters, these tag clusters are used as intermediate sets between users and items, in fact the recommendation is based on two closeness values: closeness between users and tag clusters, and closeness between item and tag clusters.

Also an algorithm is proposed to give recommendations in a professional social network. This algorithm aims to recommend a group of ranked authors similar to a given criterion, in a social network of connected authors. In this system, recommendations are based on the social network analysis and the user profiles, these profiles are represented by vectors of keywords. In a semantic social recommender system[1], the authors proposed to represent the users by a vector of scores of interests assigned to topics taken from domain taxonomy; this taxonomy represents item categories, then a semantic similarity measure (between the users vectors and domain taxonomy) is used in a semantic-based (taxonomy) recommender system.

II. RECOMMENDER TECHNIQUES

Specifically, recommender systems have (i) background data, that the system has before the recommendation process begins, (ii) input data, that user must communicate to the system in order to generate a recommendation, and (iii) an recommendation algorithm that combines background and input data to arrive at its suggestions. Five different recommendation techniques are summarized in Table I. Assume that I is the set of items over which recommendations might be made, the set of users whose preferences are known is assumed as U, u is the user to whom items will be recommended, and i is some item for which we would like to predict u's preference. Recommender systems can be classified into various categories as follows:

A. Collaborative recommender systems [8] generate new recommendations on the basis of commonalities between users and their aggregate ratings or inter-user comparisons. In collaborative system, a typical user profile consists of a vector of items and their ratings, continuously incremented as the user interacts with the system over time. Some collaborative systems used time-based discounting of ratings to account for drift in user interests. In some cases, ratings may be real-valued indicating degree of preference or binary (like/dislike). Some of the most important systems using this technique are GroupLens/NetPerceptions, Ringo/Firefly, Tapestry and Recommender.

The greatest strength of collaborative techniques is that they are completely independent of any machine-readable representation of the objects being recommended, and these systems work well for complex objects such as music and movies where variations in taste are responsible for much of the variation in preferences.

B. Demographic recommender systems [9] aim to categorize the user based on personal attributes and make recommendations based on demographic classes. Grundy is an early example of this kind of system, that recommended books based on personal information gathered through an interaction. The user's responses were matched against a library of manually assembled user stereotypes. Krulwich, is a system, that uses demographic groups from marketing research to suggest a range of products and services. To gather the data for user categorization a short survey is used. Pazzani's model uses Winnow to extract features from users' home pages that are predictive of liking certain restaurants. Rich's system used hand-crafted attributes with numeric confidence values. The benefit of a demographic approach is that it may not require a history of user ratings of the type needed by collaborative and content-based techniques.

C. In a ***content-based system, [2]*** the objects of interest are defined by their associated features. Text recommendation systems like the newsgroup filtering system NewsWeeder uses the words of their texts as features. A content-based recommender system learns a profile of the user's interests based on the features present in objects the user has rated. The type of user profile derived by a content-based recommender depends on the learning method employed. Neural nets, decision trees, and vector-based representations have all been used.

D. ***Utility-based recommenders*** [9] make suggestions based on a computation of the utility of each object for the user. The central problem is how to create a utility function for each user. The e-commerce site PersonaLogic2 and Tête-à-Tête each have different techniques for arriving at a user-specific utility function and applying it to the objects under consideration. The user profile therefore is the utility function that the system has derived, and the system employs constraint satisfaction techniques to locate the best match. The advantage of utility-based recommendation is that it can factor non-product attributes, such as product availability and vendor reliability, into the utility computation, making it possible for example to trade off price against delivery schedule for a user who has an immediate need.

E. ***Knowledge-based recommendation*** [9] attempts to suggest objects based on inferences about a user's preferences and needs. In some sense, all recommendation techniques could be described as doing some kind of inference. Knowledge-based approaches are distinguished in that they have functional knowledge: they have knowledge about how an item meets an individual's need, and can therefore reason about the relationship between a need and a possible recommendation. The user profile can be any knowledge structure that supports this inference.

The knowledge used by a knowledge-based recommender can also take many forms. Entree uses knowledge of cuisines to infer similarity between restaurants. Google uses information about the links between web pages to infer popularity and authoritative value. Utility-based approaches calculate a utility value for objects to be recommended, and in principle, such calculations may be based on functional knowledge.

Table I: Recommendation Techniques

Technique	Background	Input	Process
Collaborative	Ratings from U of items in I.	Ratings from u of items in I.	Identify users in U similar to u, and extrapolate from their ratings of i.
Content-based	Features of items in I	u's ratings of items in I	Generate a classifier that fits u's rating behavior and use it on i.
Demographic	Demographic information about U and their ratings of items in I.	Demographic information about u.	Identify users that are demographically similar to u, and extrapolate from their ratings of i.
Utility-based	Features of items in I.	A utility function over items in I that describes u's preferences.	Apply the function to the items and determine i's rank.
Knowledge-based	Features of items in I.	Knowledge of how these items meet a user's needs.	A description of u's needs or interests. Infer a match between i and u's need.

III. GRAPH PROCESSING PLATFORMS

Hadoop, YARN, Stratosphere, Giraph, GraphLab, and Neo4j are six popular platforms and may be used for graph processing. We introduce each platform in the following, in turn.

A. ***Hadoop [3]*** is an open-source, generic platform for big data analytics. Hadoop is based on the MapReduce programming model. Hadoop has been widely used in many areas and applications, such as log analysis, user interests' prediction, advertisement, search engine optimization, etc. It is becoming the de-facto platform for batch data processing. Its programming model may have low performance and high resource consumption for iterative graph algorithms, because of the structure of its map-reduce cycle. For example, for iterative graph traversal algorithms Hadoop would often need to store and load the entire graph structure during each iteration, to transfer data between the map and reduce processes through the disk-intensive HDFS, and to run an convergence-checking iteration as an additional job.

B. ***YARN*** is the next generation of Hadoop. YARN [3] can seamlessly support old MapReduce jobs, but was designed to facilitate multiple programming models, rather than just MapReduce programming model. A major contribution of YARN is to separate functionally resource management and job management; the latter is done in YARN by a per-application manager. The original Apache Hadoop MapReduce framework has been modified to run MapReduce jobs as an YARN application manager.

C. ***Stratosphere*** is an open-source platform for large-scale data processing. Stratosphere [3] consists of two key components: Nephele and PACT. Nephele is the scalable parallel engine for the execution of data flows. In Nephele, jobs are represented as directed acyclic graphs (DAG), a job model similar for example to that of the generic distributed platform Dryad. For each edge (from task to task) of the DAG, Nephele offers three different types of channels for transporting data, through the network, in-memory, and through files. PACT is a data- intensive programming model that extends the MapReduce [7] model with three more second-order functions (Match, Cross, and CogGroup). PACT supports several user code annotations, which can inform the PACT compiler of the expected behavior of the second-order functions. By using this information, the PACT compiler can produce execution plans that avoid high cost operations such as data sorting and shipping, and data spilling to the disk. Compiled PACT programs are converted into Nephele DAGs and executed by the Nephele data flow engine. Hadoop Distributed File System (HDFS) is used for Stratosphere as the storage engine.

D. ***Giraph*** is an open-source, graph-specific distributed platform. Giraph uses the Pregel [5] programming model, which is a vertex-centric programming abstraction that adapts the Bulk Synchronous Parallel (BSP) [6] model. A BSP computation proceeds in a series of global supersteps. Within each superstep, active vertices execute the same user-defined function, and create and deliver inter-vertex messages. Barriers ensure synchronization between vertex computations: for the current superstep, all vertices complete their computation and all messages are sent before the next superstep starts execution. Giraph utilizes the design of Hadoop, from which it leverages only the Map phase except reduce phase. For fault-tolerance, Giraph uses periodic checkpoints; to coordinate superstep execution, it uses ZooKeeper. Giraph supports in-memory executions, which can speed-up job execution, but, crashes may occur in case of large amounts of messages or big datasets, due to lack of memory.

E. ***GraphLab*** is an open-source, graph-specific distributed computation platform implemented in C++. Besides graph processing, it also supports various machine learning algorithms. GraphLab[3] stores the entire graph and all program state in memory. For improved performance, GraphLab implements several mechanisms such as: supporting asynchronous graph computation to alleviate the waiting time for barrier synchronization, using prioritized scheduling for quick convergence of iterative algorithms, and efficient graph data structures and data placement. We run all our GraphLab experiments in a synchronized mode for matching the execution mode of the other platforms.

F. ***Neo4j*** is one of the popular open-source graph databases. Neo4j [3] stores data in graphs rather than in tables. Every stored graph in Neo4j consists of relationships and vertices annotated with properties. Neo4j can execute graph-processing algorithms efficiently on just a single machine, because of its optimization techniques that favor response time. Neo4j uses a two-level, main-memory caching mechanism to improve its performance. The file buffer caches the storage file data in the same format as it is stored on the durable storage media. The object buffer caches vertices and relationships (and their properties) in a format that is optimized for high traversal speeds and transactional writes. Table II summarizes the above discussion.

Table II Graph processing platforms

Platform	Version	Type	Release date
Hadoop	**hadoop-0.20.203.0**	**Generic, Distributed**	**2011-05**
YARN	**hadoop-2.0.3-alpha**	**Generic, Distributed**	**2013-02**
Stratosphere	**Stratosphere-0.2**	**Generic, Distributed**	**2012-08**
Giraph	**Giraph 0.2 (revision 1336743)**	**Graph, Distributed**	**2012-05**
GraphLab	**GraphLab version 2.1.4434**	**Graph, Distributed**	**2012-10**
Neo4j	**Neo4j version 1.5**	**Graph, Non-distributed**	**2011-10**

IV. PARALLEL PROGRAMMING MODELS

A. MapReduce

Map Reduce [4] is a distributed programming model derived from functional paradigm. It is dedicated for complex and distributed computations. MapReduce divides the processing into two consecutive stages: the map and the reduce phase. A data input to first stage is split into chunks. Each chunk is processed by one mapper. Mappers can emit < key,value > pairs to the reduce phase. Before the Reduce phase the pairs are sorted and collected according to key, therefore each reducer gets <key,list(value) > format of data. Output of reduce phase is saved into distributed file system. While processing iterative graph algorithms by means of MapReduce, in a single iteration, the graph structure and other static data, must be transferred over the network of computational nodes from Mappers to Reducers so that they can serve as an input for the next iteration. This causes a great network overhead and seems to be the biggest pitfall of graphs processing in MapReduce. The stateless, two-phased computational nature of MapReduce does not allow vertices to reside on the same machine throughout the whole computation. After every phase (map/reduce) the entire data must be written to a global memory in order to be consumed in the next phase.

Since the distributed file system serves as the global memory in the cloud environment, the whole computation is very disk-intensive. Additionally, the map and reduce workers reside on different physical machines and for that reason, the data is constantly transferred over the network, which is the scarce resource in cloud environment.

B. Bulk Synchronous Parallel

Having in mind the inefficiency of MapReduce for iterative graph processing, Google has created Pregel – a distributed, fault-tolerant system for graph processing, based on Bulk Synchronous Parallel (BSP) processing model. Although Pregel is a proprietary system, it has inspired the creation of several open-source systems which adopt the BSP [4] model like Apache Hama or Apache Giraph. In BSP, the computation consists of a series of supersteps. In every superstep, a user defined function is executed in parallel on every item from the dataset acting as an agent. Pregel and Pregel-inspired graph analytics systems are vertex-centric: a single agent computation has a graph representation in BSP. It consists of node identifiers, its current value and/or state, and a list of nodes outgoing edges. Before the computation, all vertices are partitioned and loaded into local memories of machines and remain there throughout the entire computation, so that all processing is carried out on the local memory. In every superstep, a vertex receives messages from other vertices and executes a user defined function compute(), which performs local processing and sends messages to some or all of its neighbours. Once a vertex finishes the local computation, it deactivates itself and waits for all other vertices to finish. The barrier of synchronization mechanism allows the next superstep to begin when all vertices complete processing in the current superstep. Then only the vertices that have received a message are activated.

A stateful nature of the BSP model facilitates efficient graph computing since graph's structure does not need to be send over the network. Only specific messages necessary for the algorithm execution are exchanged. There is no network overhead related to graph structure passing as in the MapReduce model. Moreover, storing the whole graph structure in local memory of workers allows in-memory computation. The need of disk write-reads as well as objects serialization between iterations is eliminated. However, it is possible only if the graph structure fits in memory of all workers. Otherwise, the spilling-to-disk techniques must be implemented. Although such techniques are claimed to be implemented in Pregel, to our best knowledge, they are not yet supported in its open-source followers. This might be a drawback of choosing BSP model for graph processing, as we later present.

V. RECOMMENDATION ALGORITHMS

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 item-to-item collaborative filtering — focus on finding similar items, not similar customers. The algorithm attempts to find similar items for each of the user's purchased and rated items,. It then aggregates the similar items and recommends them.

VI. CONCLUSION

This paper focuses on providing the overview about the various recommendation techniques developed or proposed till now. Various categories in which recommendation algorithms can be classified are discussed. Also various open source graph processing platforms are discussed in detail.

ACKNOWLEDGEMENT

It is my pleasure to get this opportunity to thank my beloved and respected Guide Prof. C.O. Banchhor who imparted his valuable knowledge specifically related to hadoop and giraph. We are grateful to department of Information Technology SCOE, Pune for providing us infrastructure facilities and moral support.

REFERENCES

[1]. Dalia Sulieman, Maria Malek, Hubert Kadima, et. Al. ,"Graph searching algorithm for semantic social recommendation", IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2012, pp. 733-738

[2]. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," Knowledge and Data Engineering, IEEE Transactions on, vol. 17, no. 6, pp. 734–749, 2005.

[3]. Yong Guo, Marcin Biczak, et al.,"How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis"

[4]. Tomasz Kajdanowicz, Wojciech Indyk, et al. "Comparison of the Efficiency of MapReduce and Bulk Synchronous Parallel Approaches to Large Network Processing", IEEE 12th International Conference on Data Mining Workshops,2012

[5]. Grzegorz Malewicz, Matthew H. Austern,et al.,"Pregel: A System for Large-Scale Graph Processing", ACM 978-1-4503-0032-2/10/06, SIGMOD'10, June 6–11, 2010

[6]. Zhigang Wang, Yubin Bao,et al., "A BSP-based Parallel Iterative Processing System with Multiple Partition Strategies for Big Graphs, IEEE International Congress on Big Data,2013

[7]. Lina Li, Cuiping Li,et al., "Map Reduce-Based SimRank Computation and Its Application in Social Recommender System", IEEE International Congress on Big Data,2013

[8]. Zhi-Dan Zhao, Ming-Sheng Shang, "User-based Collaborative-Filtering Recommendation Algorithms on Hadoop",IEEE, Third International Conference on Knowledge Discovery and Data Mining,2010

[9]. Robin Burke, "Hybrid Recommender Systems: Survey and Experiments",

[10]. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grou-plens: an open architecture for collaborative filtering of netnews," in CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work. New York, NY, USA: ACM, 1994, pp.175–186.

[11]. U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating "Word of Mouth"," in Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems, vol. 1, 1995, pp. 210–217.