International Journal of Advances in Applied Sciences (IJAAS)
Vol. 4, No. 3, September 2015, pp. 109~116
ISSN: 2252-8814

Song Recommendation System Using Maximal b-Matching

Deepa S, Varsha R, Parvathi R
School of Computing Sciences and Engineering, VIT University, India

Article Info

Article history:

Received Jun 15, 2015
Revised Aug 10, 2015
Accepted Aug 26, 2015

Keyword:

B-matching
Big data
Hadoop
MapReduce
Bipartite graph

ABSTRACT

The last decade has witnessed a fundamental paradigm shift on how information content is distributed among people. Nowadays, an increasing number of platforms allow everyone to participate both in information production and information consumption. The phenomenon has been coined as democratization of content. However, as the opportunities to find relevant information and relevant audience increases, so does the complexity of a system that would allow suppliers and consumers to meet in the most efficient way. Our motivation is building a "featured item" component for social-media applications. Such a component would provide recommendations to consumers each time they login the system. The existing system follows either collaborative filtering or content based filtering. Collaborative filtering methods are based on collecting and analyzing a large amount of information on user's behaviours, activities or preferences and predicting what users will like based on their similarity to other users. Content-based filtering methods are based on a description of the item and a profile of the user's preference. Both of these methods require input from the user in the form of ratings or other user's likes. But social content matching takes into account only the user's preferences and also the capacity constraints. For each item 't' and each user 'u', consider constraints on the maximum number of edges that t and u can participate in the matching. These capacity constraints can be estimated by the activity of each user and the relative frequency with which items need to be delivered. Here we introduce the concept called b-matching goal is to find a matching that satisfies all capacity constraints and maximizes the total weight of the edges in the matching. The result of b-matching is the set of songs that are to be recommended to the user based on his likes.

Corresponding Author:

Dr. Parvathi R,
Departement of Computing Sciences and Engineering,
VIT University,
Vandalur, Kelambakkam Road, Chennai, Tamil Nadu 600127.
Email: parvathi.r@vit.ac.in

1. INTRODUCTION

Social media in general exhibit a rich variety of information that can be both structured and unstructured from different sources. Such type of data is very difficult to process that contains the billions records of millions people information that includes the web sales, social media, audios, images and so on. The need of big data comes from the Big Companies like Yahoo, Google, and Facebook etc for the purpose of analysis of big amount of data which is in unstructured form. Google contains the large amount of information so there is the need of Big Data Analytics that is the processing of the complex and massive datasets. Big data analysis fundamentally transforms operational, financial and commercial problems that were previously unsolvable within economic and human capital constraints using discrete data sets and on premises hardware. Usually the big complaint in social enterprise is the loss of privacy. But this is the surface not the reason. It is the unintelligent way of using data, with no empathy and no deep insights.

Big data is the expression for a group of data sets so outsized and multifaceted that it becomes difficult to process using readily available database management tools or traditional data processing applications. Big data requires exceptional technologies Massively Parallel-Processing (MPP) databases, search-based applications, data-mining grids, distributed file systems, distributed databases, cloud based infrastructure (applications, storage and computing resources) and the Internet to efficiently process large quantities of data within tolerable elapsed times. Big Data provides an infrastructure for transparency in various industries, to unravel uncertainties such as inconsistent component performance and to enhance the ease of use. The most well-known technology used for Big Data is Hadoop.

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a reliable, scalable, distributed computing environment. It is a disruptive force in the traditional data management space. It is part of the Apache project sponsored by the Apache Software Foundation. Hadoop was inspired by Google's MapReduce, a software framework in which an application is broken down into numerous small parts. Any of these parts (also called fragments or blocks) can run on any node in the cluster. Apache Hadoop ecosystem consists of the Hadoop kernel, MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper.

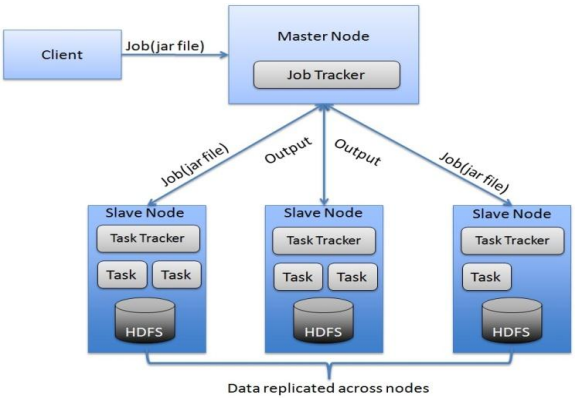

Figure 1. Hadoop Architecture

As Hadoop can be hosted on commodity hardware (usually Intel PC on Linux with one or 2 CPU and a few TB on HDD, without any RAID replication technology), it allows them to store huge quantity of data (peta-bytes or even more) at very low cost compared to the other systems. The problem that most commonly occurs in economic markets, labour markets, internet advertising, and elsewhere is the matching hitch. Matching problems are omnipresent. In this paper we focus on an application of matching in social media.

Musicians are competing for an audience among millions of others trying just as hard. And it's not the listener's fault if they miss out on something that will change their lives – these days, anyone can gain access to a library of over 15 million songs on demand for free. Recommendation technology is powering the new radio and there is a chance to make it valuable for more than just the top 5 percent of musicians.

It is assumed that music recommendation usually means:

Artist or song similarity: an anonymous list of similar items is recommended. This is seen on almost any music service. Without any context, this is just a suggestion of what other artists or songs are similar to the one you are looking at. Formally, this is not truly a recommendation as there is no user model involved.

Personalized recommendation: Given a "user model" a list of songs or artists that the service does not think the user knows about yet.

Playlist generation: This is different from the above two in a way that the user receives a list of items in some order (usually meant to be listened to at the time.) The playlist can be personalized (from a user model) or not, and it can be within catalogue. The playlist should vary artists and types of songs as it progresses, and many rely on some form of steering or feedback (thumbs up, skips, etc.).

The main amazing and useful feature of recommendation systems are that it can find things for you that you wouldn't have otherwise come across. Social music recommendations enabled by social networks are extremely valuable in music discovery as they can automatically predict anything with the help of explicit social signals.

1.1. Literature Survey

The quality of user-generated content varies drastically from excellent to abuse and spam. As the availability of such content increases, the task of identifying high-quality content in sites based on user contribution in social media sites becomes increasingly important as described by E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne in Finding high-quality content in social media, 2008. Community Question Answering (CQA) has emerged as a popular forum for users to pose questions for other users to answer. But the quality of the information retrieved from these forums varies so that a large quantity of this information cannot be used as mentioned in Learning to Recognize Reliable Users and Content in Social Media with Coupled Mutual Reinforcement, J Bian, Y Liu, D Zhou, E Agichtein, H Zha in 2009. These CQAs form the basis of the recommendation systems wherein the past information can be used as references to similar queries. The task of discovering similar objects within a given collection is common to many real world applications and machine learning problems. Many approaches can be used to find the similar objects and one such approach is similarity self-join as proposed by Baraglia, R.; De Francisci Morales, G.; Lucchese, C in Document Similarity Self-Join with MapReduce, 2010. As defined, given a collection of objects, the Similarity Self Join problem requires to discover all those pairs of objects whose similarity is above a user defined threshold. Gediminas Adomavicius, Alexander Tuzhilin, Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, 2005, have explained the current recommendation systems, their functionality and limitations. It gives clear insights into how the recommendation systems work and the things that can be done to improve the process.

2. RESEARCH METHOD

The recommendation system usually deals mainly with the song preferences of a social media user. There are hundreds of genres of songs and preferences of a user cannot be restricted to a few. So taking into account the various choices of the user our system will recommend songs that are similar. Finding out the right songs is in itself a tedious task. Here the million song dataset which consists of triplets about songs, users and play count from several websites like Last.fm is used. The Million Song Dataset was created under a grant from the National Science Foundation, project IIS-0713334. The original data was contributed by The Echo Nest, as part of NSF-sponsored GOALI collaboration. Million song dataset contains songs based on tags but then there are thousands of songs from a single tag, all of which can't be recommended. Using JSON parser the dataset will be parsed and a list on songs based on the genres will be extracted. Then using the concept of b-matching the best song would be recommended to the user.

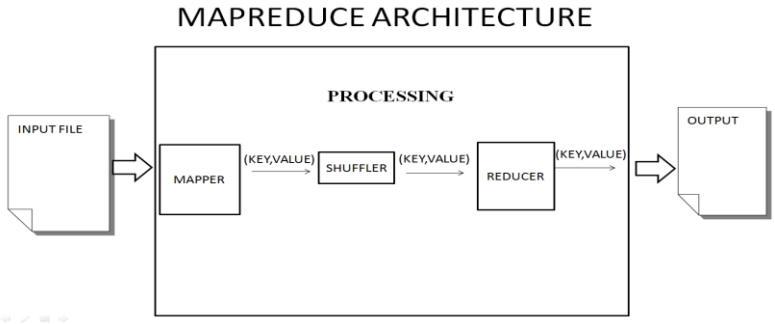

Figure 2. MapReduce architecture

Recommender systems typically produce a list of recommendations in one of two ways - through collaborative or content-based filtering. Collaborative filtering approaches building a model from a user's past behaviour (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users; then use that model to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties.

Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviours, activities or preferences and predicting what users will like based on their similarity to other users. Collaborative Filtering are based on the assumption that people who agree in past will agree in future too and that they will like the similar kinds of items as they liked in the past.

Content-based filtering methods are based on a description of the item and a profile of the user's preference. In particular, various candidate items are compared with items previously rated by the user and

the best-matching items are recommended. A key issue with content-based filtering is whether the system is able to learn user preferences from user's actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended.

Both of these methods require input from the user in the form of ratings or other user's likes. But social content matching takes into account only the user's preferences and also the capacity constraints. For each item 't' and each user 'u', consider constraints on the maximum number of edges that t and u can participate in the matching. These capacity constraints can be estimated by the activity of each user and the relative frequency with which items need to be delivered. Here we introduce the concept called b-matching whose goal is to find a matching that satisfies all capacity constraints and maximizes the total weight of the edges in the matching.

Our goal is to implement songs recommender system for the social media content. We are given a set of song ids S= {s1, . . . , sn}, which are to be delivered to a set of consumers C= {c1, . . . , cm}. For each cj and si, we assume we are able to measure the interest of consumer cj in song si with a positive weight w (cj, si). The distribution of the songs S to the consumers C can be clearly seen as a matching problem on the bipartite graph with nodes S and C, and edge weights w (si,cj). In order to avoid that each consumer cj receive too many song recommendations, we enforce a capacity constraint b (cj) on the number of items that are matched to cj. Similarly, we would like to avoid the scenario when only a few songs (e.g. the most popular ones) participate in the matching. To this end, we introduce a capacity constraint b (si) on the number of consumers that each item si is matched to. This variant of the matching problem is well known in the theoretical computer science community as the b−matching problem. We wish to find a b-matching of maximum weight.

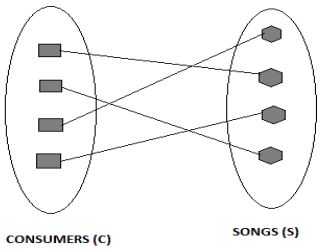

Figure 3. Sample bipartite graph matching between consumers (C) and songs (S)

2.1. b-Matching Problem

Given a graph G = (V, E), a matching M in G is a set of pair wise non-adjacent edges; that is, no two edges share a common vertex. A vertex is matched (or saturated) if it is an endpoint of one of the edges in the matching. Otherwise the vertex is unmatched.

A maximal matching is a matching M of a graph G with the property that if any edge not in M is added to M, it is no longer a matching, that is, M is maximal if it is not a proper subset of any other matching in graph G. In other words, a matching M of a graph G is maximal if every edge in G has a non-empty intersection with at least one edge in M.

A maximal matching can be found with a simple greedy algorithm. A maximal matching with k edges is an edge dominating set with k edges. Conversely, if we are given a minimum edge dominating set with k edges, we can construct a maximal matching with k edges in polynomial time. Therefore the problem of finding a minimum maximal matching is essentially equal to the problem of finding a minimum edge dominating set.

For the song recommendation system, b-matching is used to find the list of songs for each consumer based on the weight of the similarity of the match. The capacity constraint is also taken into account wherein each consumer cannot have more than ‘n’ recommendations and each song cannot be recommended to more than ‘n’ consumers. From the millions of songs that are available, the b-matching problem chooses the songs with the maximum matching weight. Further eliminations are done using the δ(e) function that reduces the number of weakly covered edges. The stackMR algorithm is inclusive of both the b-matching the removing of weak edges by pushing and popping the edges into the stack.

2.2. Stack-MR Algorithm

1: /* Pushing Stage */
2: **while** E is non empty **do**
3: Compute a maximal $\lceil \epsilon b \rceil$-matching M (each vertex v has capacity $\lceil \epsilon b(v) \rceil$), using the procedure in [12];
4: Push all edges of M on the distributed stack (M becomes a layer of the stack);
5: **for** all $e \in M$ in parallel **do**
6: Let $\delta(e) = (w(e) - y_u/b(u) - y_v/b(v))/2$;
7: increase y_u and y_v by $\delta(e)$;
8: **end for**
9: Update E by eliminating all edges that have become weakly covered
10: **end while**
11: /* Popping Stage */
12: **while** the distributed stack is nonempty **do**
13: Pop a layer M out of the distributed stack.
14: In parallel include all edges of M in the solution.
15: For each vertex v: let $\bar{b}(v)$ be the set of edges in M that are incident to v; update $b(v) \leftarrow b(v) - \bar{b}(v)$; if $b(v) \leq 0$ then remove all edges incident to v.
16: **end while**

Figure 4. Stack MR algorithm

3. RESULTS AND ANALYSIS

The result generated by the recommendation system is the name of the user and the list of songs that can be recommended to that user. The stack MR algorithm has high performance results. It performs well under a large number of edges. The b-matching algorithm works better with more edges and produces less additional overhead. This behaviour is expected, as the number of edges increase the algorithms have more flexibility. Since we add edges by lowering the edge-similarity threshold, the gain in the b-matching value tends to saturate and so it requires less number of map reduce steps. The stack MR algorithm proposes to its neighbour edges chosen uniformly at random and so the number of edges that are chosen based on similarity is less leading to easier processing.

The bipartite graphs can be represented in the form of adjacency list where each node in one side of the graph contains a list of connected nodes in the other side of the graph.

Figure 5 represents the user and song information in the form of adjacency list. It depicts the list of user who have listened to a particular song.

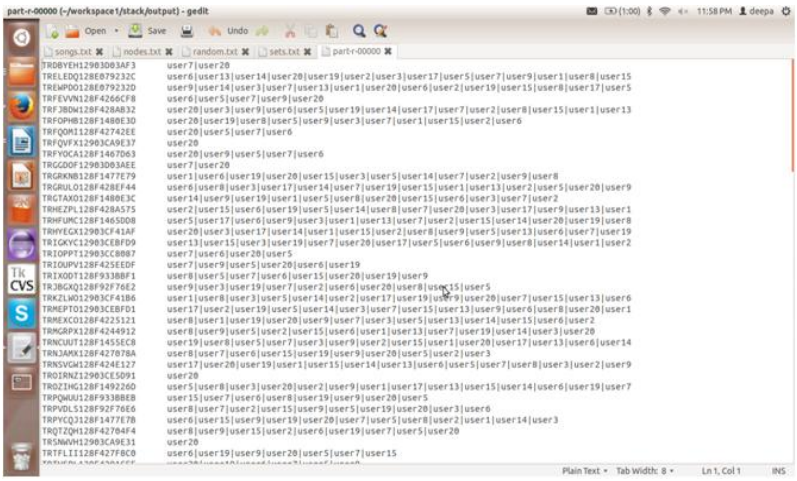

Figure 5. Adjacency list of songs and users

Output:

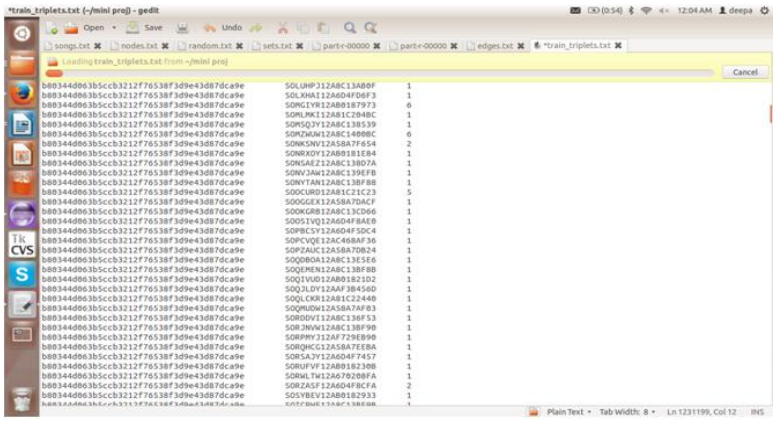

Figure 6. The data obtained from MillionSong dataset

Figure 6 contains the data obtained from the million song dataset. The million song dataset has information about the list of songs, users, the play count, artist information and album information.

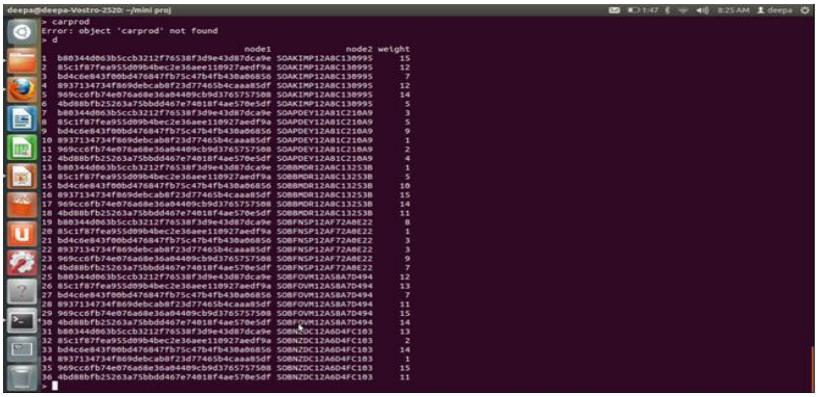

Figure 7. Weighted user-song data

Figure 7 shows the user, songs and the associated weight information obtained from the million song dataset.

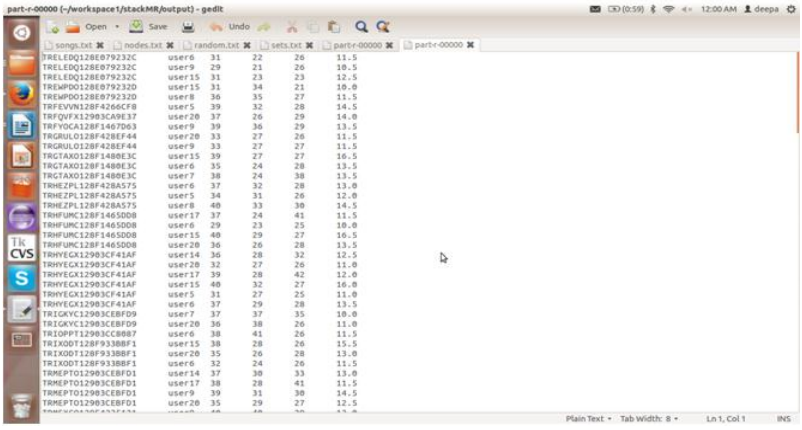

Figure 8. User, songs and the weighted average

Figure 8 depicts the user, song, the weighted average and the δ(e) that is used to remove the weakly covered edges.

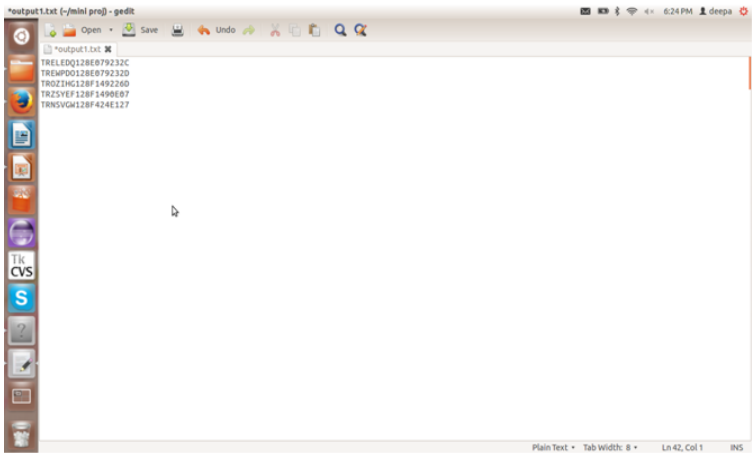

Figure 9. Recommended Songs for a particular user

Figure 9 depicts the final recommendation based on each individual user.

4. CONCLUSION

The problem of social content matching is investigated using MapReduce algorithm. Although all the algorithms can deal with any undirected graph, we focus on bipartite graphs which are relevant in our application scenarios. Furthermore using the data from the social networking sites makes the end result more refined. Also some of the problems in the recommendation systems like cross domain recommendations, constraint based recommendations, group recommendations, and context aware recommendations are not dealt with.

Cross-Domain Recommendation: Current systems are really good at learning preferences in one domain (Say movies), but the same algorithms do not work as well in other domains. So if a user likes a particular genre of songs what does it have to say about his taste in movies? These cross-domain solutions can be dealt with in the future.

Constraint-Domain Recommendation: Most of the research has focussed on the virtual goods such as movies and music, where an item can be recommended unlimited number of times. In the real world, that's often not the case. Consider restaurants. If a great restaurant is recommended to more people than it can handle, it struggles to cope up with the load and its services decline. How can recommendations be done in domains where the items are limited? Here the recommendation becomes a more relaxed version of the classic matching problem.

Group Recommendation: Here, the basic premise is to recommend an item to a group of people (e.g.) going to a movie together. Typical models compute individual recommendations and then use a smart way to combine them. But often there will be disagreements, and different groups may have different dynamics. There is a vast literature on consensus and voting strategies and mechanisms that could be explored, as well as multiple-staged user-interaction paradigms.

Context-aware Recommendation: With increasing use of mobile devices, user self-disclose a lot of contextual information (e.g.) location, current activity etc. These data sources give real-time information, which are both an opportunity and challenge for a recommender system. How to incorporate such fine grained information in recommender models?

These are some of the challenges that are faced by the recommendation systems that are yet to be overcome. The solutions to these challenges can help improve the business across the world.

REFERENCES

[1] Gianmarco De Francisci Morales, Aristides Gionis and Mauro Sozio, "Social Content Matching in MapReduce", *IEEE trans.Comp*, Vol. 4, pp. 460-469, April 2011.
[2] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", *Communications of the CACM*, Vol. 51, No. 1, pp. 107–113, 2008.
[3] R. Baraglia, G. De Francisci Morales, and C. Lucchese, "Document similarity self-join with MapReduce", *In ICDM*, pp. 731–736, 2010.
[4] Oreilly, "Hadoop The Definitive Guide", 3rd Edition, McGraw-Hill, 2012.
[5] Chuck Lam, "Hadoop in action", DreamTech press, 2011.