Breadth First Algorithm for Calculating Taste Diversity

Cil Hardianto Satriawan / 13508061
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18061@students.if.itb.ac.id

Abstract*—Recommender systems are increasingly becoming a staple of websites that track user preferences and habits. These systems generally filter vast amounts of user profile information in a process called collaborative filtering to generate a list of recommended items a guest or user may be interested in, given knowledge of the user's past taste and preferences. An extension to this concept is the possibility of calculating a user's diversity of taste, which can serve as valuable information in the actual recommending process. In this paper I will outline the design for a Breadth-First algorithm to calculate a user's diversity in taste, given a sizable amount of information regarding the user's previous habits and the existence of a statistically large database of users, similarly with their item preference history available.

***Key Words*— Recommender, Collaborative Filtering, Breadth-First Search, Diversity**

I. INTRODUCTION

With the advent and tremendous growth of the Internet in the mid-90s, people are now connected in ways never previously thought imaginable. Online, people from two opposite corners of the world can exchange information freely, demolishing geographical and spatial borders that have hindered information sharing since the beginning of human existence.

This feat is due in large part to the boom in communication and network technology achieved throughout the 70s and 80s, focused primarily in the United States and Europe. Today, the growth of the Internet is attributed primarily to the newer economical giants such as China, Korea, Vietnam, and Indonesia. The Internet will surely face unprecedented growth well into the 21st century.

The rise of The Internet brought along with it the advent of many new ways to conduct everyday activities. Most prominent among these is the advent of e-shopping, still somewhat underutilized in Indonesia and other third-world countries, e-banking, entertainment, in the form of online games, file sharing, etc., and recently, the phenomena of social networking.

Obscured somewhat by the well-documented recent boom in networking and communications, the rapid advancements of data storage and processing power, mainly in its cost/performance ratio, continues unhindered. Coupled with the growth of the Internet and the recent rise in social networking, web entrepreneurs have invented and put into practice a plethora of systems to increase customer and user satisfaction, fulfilling needs people never even knew they had before.

One of the better-known systems made possible by the increasingly large number of online users and cheap storage are recommendation systems, which are becoming increasingly common in anything from online shops to media information database websites. Particularly prominent examples of such recommender systems are the "Users who bought this item also bought…" lists on Amazon and other online shopping stores, music recommendation websites such as Last.fm, movie database IMDB (Internet Movie Database), and Facebook's "You may also know…" lists.

II. THEORY

A recommender system is a specific information filtering system that attempts to recommend information items (products, media, etc.) that the user may be interested in. These systems usually work by comparing a user's profile to a reference trait, recommending items the user previously had not considered or had knowledge of.

A. Content-Based Recommender

Recommender systems generally belong to one of two classes; the content-based approach and the collaborative filtering approach. The content-based approach seeks to recommend items based on their 'content' traits. Items the host provides are categorized based on these particular content traits, and a check is conducted against these traits to make a recommendation. For example, a company renting DVDs must keep and store users' previous rentals lists in a special database. If one such user is known to have rented titles such as *Pride and Prejudice*, *Sense and Sensibility*, and *Elizabeth*, which have been categorized as *Period Drama* and *Romance*, then said user may also be interested in *Becoming Jane*, another title with the tags *Period Drama* and *Romance*. The system will make the recommendation the next time the user comes around to make a rental or purchase.

This approach excels in its ease of implementation.

Such a system requires the addition of tags –item descriptors of varying length—and a complete database of all items and tags. The shortcoming of this approach is that it requires the provider to be aware of the contents of items beforehand, which in the case of a store supplying thousands or hundreds of thousands of titles may be difficult. The accuracy of this approach is also limited by the specifity of the tags attached. For instance, the *Drama* tag may include hundreds of titles with not much in common. Vice versa, a tag such as *Cat Warrior* may contain one or even no titles.

B. Collaborative Filtering Recommender

The other commonly used approach is collaborative filtering. This approach makes predictions through the filtering of vast amounts of user taste and preferences information. Again, there are two methods of collaborative filtering. The first type is based on the premise of a rating system, where items are rated by user on a particular scale. This method proceeds as follows:

1. Look for users who share the same rating patterns with the active user (for whom the recommendation is made).
2. Use the ratings from these like-minded users to calculate a prediction for the active user.

This method of collaborative filtering is called user-based collaborative filtering, an example of which is the Nearest Neighbour Algorithm.

However, this method proves inappropriate when dealing with items that are highly subject to user taste and preference, such as books, movies, restaurants, and music. Therefore, a fundamentally different approach is required to address these types of items.

The type of collaborative filtering we are interested in in the scope of this paper follows these steps:

1. Build an item-tem matrix determining the relationship between pairs of items.
2. Infer user taste using this matrix and data regarding user taste.

Amazon's "User that brought this item also brought…" list is an example of a collaborative filtering recommender that utilizes these steps. Another example is the Last.fm music recommendation system. To better understand the process involved, let us use Last.fm's music recommender system as a case study.

When guests register as Last,fm users, they are asked of their choice media player. Users then download a plugin which attaches itself to the user's media player, collects listening data and uploads it to Last.fm servers in a process called "scrobbling". The uploaded data is derived from the metadata contained in the listener's media file, typically an IDv3 or IDv2 tag for MP3 files. Last.fm corroborates this data and displays it on the user's profile.

Although it is now possible to obtain "personalized" recommendations, Last.fm's music recommender was traditionally situated on the Artist/Musician page, where it is displayed in a list of Similar Artists, and can be thought of as a "Fans of this band also listen to…" list. For example, the similar artist list for The Beatles is John Lennon, Paul McCartney, George Harrison, Ringo Starr, Paul McCartney & Wings, Wings, and Paul & Linda McCartney. It is understandable that the solo projects of a famous band almost always appear in similar artists, as the listeners of these solo projects are almost exclusively a subset of the listeners of the main band.

Personalized recommendations, on the other hand, take the active user's listening history list in descending order, generate the similar artist list for each artist, and checks for overlaps. The artist with the most overlaps is placed at the top of the personalized recommendation list, the artist with the second most in second place, and so on and so forth. For instance, a user has Radiohead, Muse, Frederick Chopin, and The Beatles as his/her top listened to artist. Last.fm generates the similar artists for Radiohead, Muse, Frederick Chopin, and The Beatles, and counts how many times a particular similar artist is generated. The higher the count, the higher the similar artist is placed in the recommendations list generated. Last.fm usually considers the 50 top artists/bands in a user's listening history. Time period can also be considered, to account for changes in taste; Last.fm may use the 50 top artists over the last 3 months to generate the list. As a note, this method is generally applicable to any collaborative filtering method employing a "similar item" technique. Amazon's personalized recommender system and Facebook's friend recommender lists, for example, can be thought of as belonging to this category. This paper will henceforth use this model when referring to recommender systems.

As we are not concerned with the details of the collaborative filter function itself, we will only describe those functions and sets required to generate the taste diversity gauge.

The user domain is the set of items highly ranked or at the top of a user's history list.

The similarity function $f_s(x)$ generates a set of results through the collaborative filtering method described above of an item x. It produces a set $\{y_1,y_2,...,y_n\}$ of similar results.

The similarity counter function utilizes the similarity function on a set of items and counts the number of exact matches in each resultant set.

Finally, **the recommender function** arranges the resultant set of the similarity counter function in descending order and takes the first *m* occurrences, where *m* is an arbitrary integer larger than 0.

C. The Breadth First Search Algorithm (BFS)

The breadth first search algorithm (henceforth referred to as BFS) is a graph-traversal algorithm to systematically traverse the nodes of a graph. The BFS algorithm derives its name from the fact that it traverses graphs 'sideways';

it first visits the root node, after which it visits a node *v* which is its child and is situated leftmost in a tree diagram. After this it visits its siblings –nodes situated to its right in a tree diagram—after which it visits its child nodes, so on and so forth until it has visited all the nodes in the tree diagram.

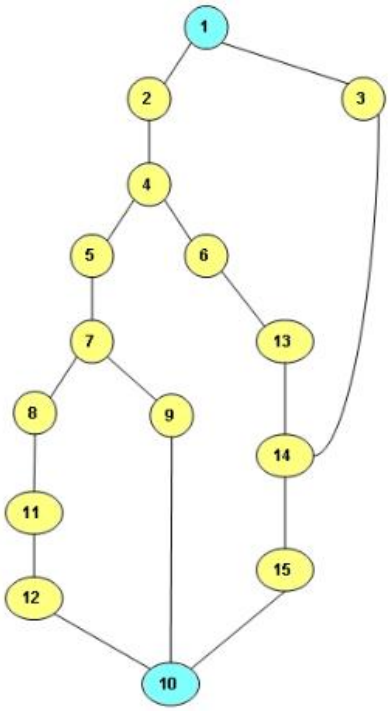

Diagram 1: Node traversal in BFS

This diagram shows the order in which nodes are transversed in a BFS algorithm, 1 being the root node and beginning of the algorithm.

For a data structure that builds nodes dynamically, as we will be using, the algorithm can be represented as a function that executes a function and enqueues the results into a queue, executing the function with the result at the head of the queue as a parameter while end conditions are not met.

III. Application of Algorithm

The following section will outline the design for a BFS algorithm that calculates the diversity of a user's tastes and preferences given a collaborative filtering environment.

A. Basic Concept

By generating the list of *m* similar items for the top *n* items in a user's profile, and subsequently counting the number of *unique* items in the list, we can gauge the diversity of the user's taste. The upper limit to this count is *m* x *n.* Calculating this value is a relatively trivial task, and is limited mostly by the site's (or database's) network bandwidth.

For a small *n* or a small *m* however, it is more difficult to obtain accurate results. These limits may realistically occur if the size of the collaborative filter's database is relatively small or if the amount of the active user's user preference data is small.

A way to overcome this is by conducting a similar artist search on the results of the first, and returning the depth at which a *non-unique* similar artist is found. A maximum depth value must be passed at the beginning of the calculation, to ensure that the function does not loop ad infinitum.

Utilizing the similarity function described in section II.B, we first determine the size of the domain set, the resultant set, and the maximum depth of the search. Other parameters may be added relevant to the specific recommender system involved, such as time period considered.

B. Implementation

The algorithm proceeds as follows:

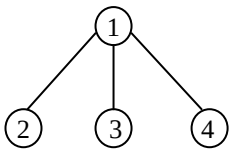

Diagram 2: Step 1

The first step is to initialize the algorithm by executing a function f_D(n, m, u) that sets the number of top items considered, the number of similar items to generate, and maximum search depth in n, m, and u, respectively. In this case, n = 3, m = 3, and u = 3. The three top items are immediately sent to the checking array, where we will conduct the checking to determine whether or not an item has been encountered before.

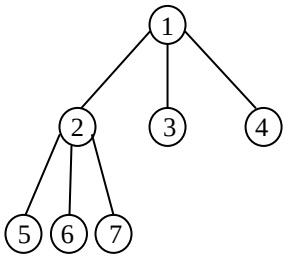

Diagram 3: Step 2

Next, we traverse to level 1 of the tree and execute the similarity function f_S(x, m) for each node in the level, where x is the node name (item identifier) and m is the number of similar items to generate. In this case, x = 2 and m = 3. Each of these nodes are immediately checked against the checking array with a function f_C(x) where x is the node number, in this case 5, and returns a Boolean. If during checking the name of the item at node 5 is found to be unique, the node is enqueued and awaits processing. The same process is repeated for the other 3 nodes created

during the same f_S function. However, if the item name contained in node 5 is not unique, the loop terminates and the depth of the tree is returned, along with the item name.

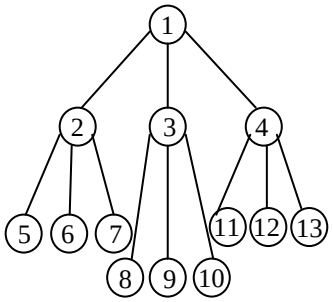

Diagram 4: Step 3

When the last level 1 node has been traversed and its item name checked, we reach the end of the level. Subsequently, f_S is executed with the object at the head of the queue as the parameter. In this case, x = 5, and the tree depth level variable is incremented. This process continues as long as no non-unique items have been found and the maximum tree depth has not been reached. As u = 3 in this example, the process will stop once it finds a non-unique or when it has just enqueued node 49.

If by the end of the process no non-unique items have been found, the user can be said to have a large diversity of taste relative to the size of the parameters used. If a match is found, then the depth of the tree at the solution will be compared to the size of parameters *n* and *m* to obtain a normalized value representing the user's taste diversity. The scale of the value obtained is arbitrary.

With some careful analysis, certain knowledge of user's tastes and habits can be gleaned. For example, a solution found at level 1 suggests a very narrow preference range, whereas a solution involving a large *n* and *m* value at a large depth suggests a user with high taste diversity. However, at such small datasets, one should be careful not to make conclusive remarks regarding user tastes.

IV. Analysis

The use of the BFS algorithm to produce accurate results in the above conditions is highly subject to the size of the parameters used. Unfortunately, physical testing was not conducted, and therefore further research is required to determine the upper and lower limits the parameters must satisfy in order to produce accurate results.

Accurate results of taste diversity calculations highly depend on a large database of available user data. It can be shown that results become accurate only when the amount of data analyzed is in the thousands or tens of thousands of records. In the case of the above algorithm, we assume that the database used for the execution of the similarity function is suitably large, and hence results produced should be reasonably accurate.

There are two methods mentioned in the algorithm, the first not involving the use of the BFS algorithm. However, for suitably large datasets the previous method is considerably lighter on the servers involved. Unless a future method is found to lighten the load on servers during node traversals of this type, it would be unfeasible to implement this algorithm on a commercial website.

Finally, the application of taste diversity algorithms is still highly underrated. During the process of writing, the author has only managed to discover such algorithms on music recommendation websites, such as Last.fm. Diversity or 'eclecticity' values are used mainly as a means of showing musical randomness and as a point of pride. However, careful analysis of such data should theoretically lead to telling information regarding a user's taste. For instance, application of such 'taste' diversity algorithms on Facebook's "you may also know…" lists may result in knowledge of the user's social circles; the amount of disconnected social circles he/she has, friending habits, etc.

Ultimately, the purpose of taste diversity algorithms should be to make better recommendations for the users involved. For example, users who have high taste diversity may appreciate recommendation that do not appear to be similar to their top items, and would welcome such suggestions as they diversify their taste.

V. Conclusion

The subject of collaborative filtering and recommender systems is one with huge potential in the future. Applications include online medical treatment, book and file sharing, and other subjects unimaginable at the moment.

Similarly, the idea of calculating user taste diversity has many applications, and may be applied to certain subjects in novel ways. In conclusion, this algorithm is a small contribution on my part to the vast and growing methods of taste diversity calculation, and may, hopefully, be useful in the future.

VI. Acknowledgment

I would like to thank my mentor, Ir. Rinaldi Munir, M.T., as it was due to his guidance and his textbook that this paper was written and completed.

References

[1] http://en.wikipedia.org/wiki/Breadth-first_search accessed: 6 Dec 2010 19:13

[2] K. Bradley and B. Smith, *Improving Recommendation Diversity*,Smart Media Institute

[3] http://en.wikipedia.org/wiki/Collaborative_filtering accessed: 6 Dec 2010 20:08

[4] http://en.wikipedia.org/wiki/Recommender_systems accessed: 6 Dec 2010 20:17

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

ttd

Nama dan NIM