# Graph-search based Recommendation system

This is project is about building a recommendation system using graph search methodologies. We will be comparing these different approaches and closely observe the limitations of each.

## Abstract:

Implemented a movie recommendation system using the movielens dataset from the grouplens site. This dataset is transformed to a bipartite graph which allowed to address the problem using graph based traversal algorithms instead of usual approaches that are used by recommendation systems. The goal is to implement collaborative filtering technique as well as content based recommendation using the graph traversal algorithms. We will evaluate the advantages and shortcomings and then also discuss how we can improve on this approach.

## Introduction:

The amount of content that is being generated by social media sites, movies, tv shows etc is increasing tremendously and its very hard for a user or person to choose from such a huge pool of content. There are endless choices. Hence we need to filter out most of these content and give suggestions to user. Recommendation systems are designed to solve this very problem to give users best suggestions based on existing data and the user preferences.

Recommendation systems are widely used in e-commerce sites such as Netflix to suggest movies , amazon to suggest products, music application such as iTunes and spotify to suggest next songs that the user may like to hear. It can be applied to even domains such as social networking. Facebook uses it for suggesting friends.

In this project we implement a collaberative filtering recommendation system that uses existing data to give better suggestions. We will be building a bipartite graph from the data set to support graph traversal for collaborative filtering system.

## Data:

The data set is obtained from http://grouplens.org/ . They have a collection of ratings of movies from MovieLens website . This data set covers 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 671 users. For implementing the collaberative filter system we will be using all the data except for tags. The data set has mainly two files : movies.csv, ratings.csv.

Number of users: 671
Number of movies: >9000
Number of ratings: 100,000

Files: movies.csv, ratings.csv

## Representation of data:

To facilitate graph traversal techniques and collaborative filtering , the data is transformed into bipartite graph representation. In a bipartite graph, nodes are divided into two distinctive sets. Links between pairs of nodes from different node sets are admissible, while links between nodes from the same node set are not allowedIn our case the information is about weather or not a person(customer) has watched the movie (product) and the how much rating the customer has given for the movie. Such an information can be easily represented as show in the below table:

| customer/movie | movie 1 | movie 2 | movie 3 | movie 4 |
|---|---|---|---|---|
| customer1 | 0 | 1 | 0 | 1 |
| customer2 | 0 | 1 | 1 | 1 |
| customer3 | 1 | 0 | 1 | 0 |

The zeros in the above table represent weather a customer has watched a movie or not. The nonzero's represent that a customer has watched the movie and the numeric value represents the rating he/she has given for that movie. You can traverse from customer to movie but you cannot traverse from customer to customer directly . Likewise you cannot directly traverse form movie to movie either.

Biparte matrix translation to graph:

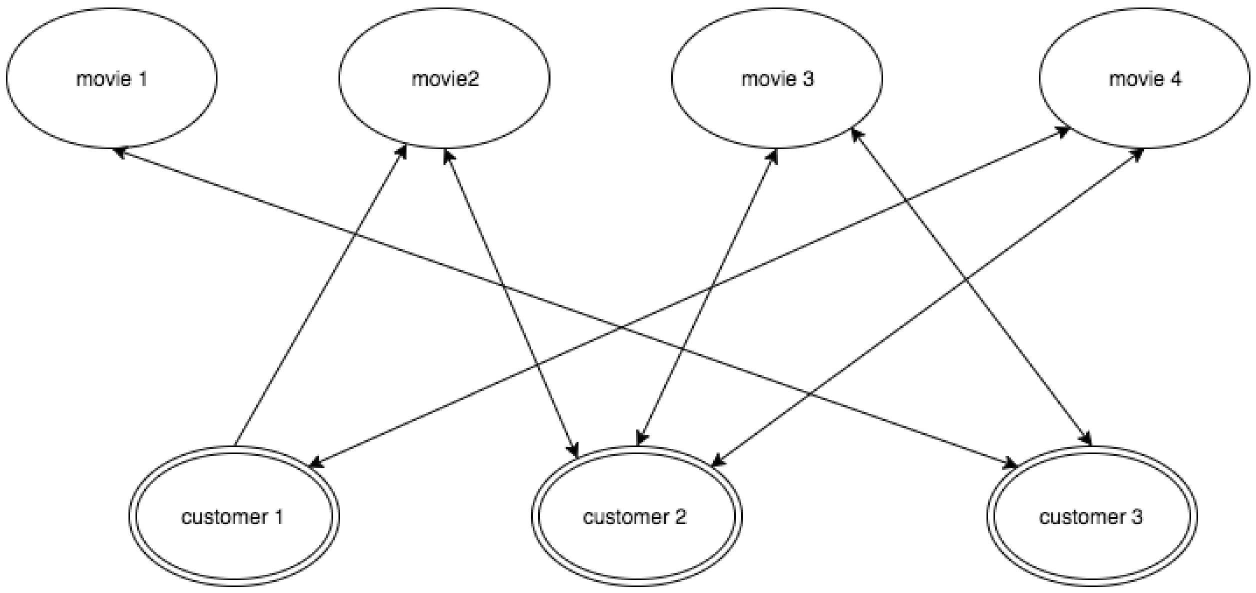

Figure 1:

The actual data set has 9123 movies and 671 customer. So the biparte graph matrix is of size (9123 * 671).

## Related work:

In general recommendation systems are implemented in three ways:

### 1. Content based approach:

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on a description of the item or product and a profile of the user's preference

### 2. Collaborative filtering:

Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an “understanding” of the item itself ### 3. A hybrid of collaborative and content based approach: In this approach we combine the both the collaborative and content based approaches to come with recommendations.

Our main focus in this project is to implement the collaborative filtering as well as the content based filtering.

## Recommendation Algorithms:

### 1. The content based Filtering:

The idea behind content based filtering is when a user likes/watches certain movies, using the meta information of the movies that the user watched we will suggest similar movies which may have the same properties. For example the following meta information such as the genre about a movie can be used to suggest similar movies that belong to the same genre. We can also couple this with ratings that the user has given to these movies earlier.
meta-properties: genre
user-given-properties: rating

Here is a simple algorithm

```
Algorithm:
    Step1: choose  all the movies that the user watched.
    Step2: obtain genre of all the movies that user watched

    Step 3: sum all the ratings for each genre that is given by the target user

    Step 4: Divide the cumulative rating of each genre with
    the number of movies in that genre.

    Step 4: Now pick the top three genres using the above computation
    and recommend movies that belong to that genre.
```

This may not be the best approach but this takes into consideration that may be a user likes a particular genre and he is trying to find a good movie in that genre. He may not have found a good movie so far. Or it can also be that the user in general likes movies from certain genres more than other genre.

### 2. Collaborative Filtering:

Collaborative filtering can be implemented in two ways . User based collaborative filtering and item based collaborative filtering. In this project we will be focussing on the user-user collaborative filtering. In user-user collaborative filtering when try to recommend a user , we try to find other similar users who have watched almost the same movies as our current user. We use similarity metrics such as euclidian distance, manhattan distance , pearson correlation etc to find such similar users. In this project implemented euclidian distance to find similar users. We will be taking the example of the tabl1 and try to recommend movies for customer 1 in the table.

#### Euclidian distance Similarity:

From the table 1 let’s assume we are trying to recommend movies for customer 1. Our goal is to find similar users. In order to do this we try to find the euclidian distance of customer 1 to all other customers respectively.

The euclidian distance for any two vectors, p = (p1, p2,. . . , pn) and q = (q1, q2,. . . , qn) are two points in Euclidean n-space, is the distance (d) from p to q, or from q to p and is given by the Pythagorean formula: $\sqrt[2]{\sum_{i=1}^{n}(q_i - p_i)^2}$

Here vecotors are nothing but the rows of the biparte matrix shown in table 1. i.e customer 1 row is vector p and any other customer such as customer 2 row is vector q. The distance between customer 1 and customer 2 is computed as follows:

customer 1, p = (0,1,0,1)
customer 2, q = (0,1,1,1)

$\sqrt[2]{((0-0)^2 + (1-1)^2 + (0-1)^2 + (1-1)^2)^2} = 1$

As you can see the distance is one. We obtain distances of customer 1 w.r.t all other users. We choose the first few users( atleast three) who are closer to the customer 1. The most important thing is we ignore all those users who have a euclidian distance of zero w.r.t customer 1. This is becuase those users have watched the same set of movies as customer 1 and have no other information to provide that would be helpfull in recommending customer 1.

We compute a distance metric as follows:

| d | customer1 | customer 2 | customer 3 |
|---|---|---|---|
| distance from customer1 | 0 | 1 | 2 |

So customer 3 not very similar to customer 1 . But customer 2 is similar and may have some intresting information that we can use to suggest movies to customer 1.

**graph based recommendation :**

Now that we have the list of users (neighborhood users) similar to the target users( whom we are recommending)we will use the biparte graph matrix to search for movies that can be recommended to the target user.

The similarity between customer 1 and customer 2 is obvious becuase they have both watched movies "move1" and "movie4". As a result, "movie3" is recommended to customer 1 because customer 2 has watched it too. From the distance metrics we know that customer 1 and customer 3 are not very similar. Therefore, customer 1, which has been purchased by customer 3, will not be recommended to customer 1.

The above recommendation approach can be easily implemented in a graph-based model by computing the associations between movie nodes and customer nodes. In this context, the association between two nodes is determined by the existence and length of the path(s) connecting them. Standard collaborative filtering approaches, including both the user-based and item-based approaches, consider only paths with length equal to 3. For instance, the association between customer 1 and movie3 is determined by all paths of length 3 connecting customer 1 and movie3. It is easy to see from Figure 1 that there exist two paths connecting customer1 and movie3:
customer1—movie1—customer2-movie3
and customer1—movie4—customer2—movie3.

This strong association leads to the recommendation of movie3 to customer1. Intuitively, the higher the number of distinctive paths connecting a product node to a consumer node, the higher the association between these two nodes. The product therefore is more likely to be recommended to the consumer. Extending the above approach to explore and incorporate transitive associations is straightforward in a graph-based model. By considering paths whose length exceeds 3, the model will be able to explore transitive associations.

So we can formalize this as follows:
If there are n paths between (customer i , movie i) then the the wieight of each path is computed as follows:

Aglorithm:

```
Take constant alpha= (0,1)
weights = 0
```

```
For each path between (customer i and movie i):
    compute the depth of the path.
    weights = weights + $(alpha)^3$
```

$weights(customer1, movie3) = (0.5)^3 + (0.5)^3 = 0.25, and weihts(cusotmer1, movie1) = 0$

It's zero becuase there is no path to movie1. Hence we will recommend movie 3 to customer 1.

## Experimental Evaluation:

One of the ways to evaluate the content based and collaberative based techniques is to use the similarity metrics as follows:

1.) Compute the euclidian distances of the all users with respect to the target user(who is to be recommended). Obtain all the similar users i.e users whose distance is less w.r.t target user.

2.) Now compute the recommendation filters and recommend the movie to the target user. Update the movie list of the target user.

3.) Now compute the euclidian distances of all the previous similar users w.r.t the updated target user. And see how much has the cummulative distace varied.

4.) We compute this change in cummulative distance for both collaberative as well as content based recommendation. The better algorithm is the one whose cummulative distance has reduced drastically.

Lets say the collabertive algorithm recommends "movie3" to customer 1 And the content based algorithm recommends "movie1" to customer 1

So now updated vectors are:

**For collabertive:**

Target User: customer 1, p = (0,1,1,1)
Similar Users: customer 2, q = (0,1,1,1)

Total distance = $\sqrt[2]{((0-0)^2 + (1-1)^2 + (1-1)^2 + (1-1)^2)^2} = 0$

**For content based:**

Target User: customer 1, p = (1,1,0,1)
Similar Users: customer 2, q = (0,1,1,1)

Total distance = $\sqrt[2]{((1-0)^2 + (1-1)^2 + (0-1)^2 + (1-1)^2)^2} = \sqrt[2]{2}$

It seems like the content based has not fared well as it failed to give the most similar recommendation. But this is epxected behaviour. This approach is more helpfull when we are trying to compare one collabertive filtering algorithm with another collaberative filtering algorithm.

In this program we have implemented only for one movie recommendation to the target user customer 1. So apparently the euclidian distances are not good enough to compare the algorithms. This is because the program took a lot of time just to execute for one target user.

```
The content Based recommendatin is:
284      318  Shawshank Redemption, The (1994)  Crime|Drama

The collaberative based recommendatoin is :
15       16   Casino (1995)  Crime
```

## Conclusion:

1.) There are many ways to implement collaberative as well as content based filtering. What we have implemented in this project is not the best approach. It can be improved a lot more.

2.) In this experiment though we have used the Euclidian distance to get similar users , there are far more better approaches to get similar / neighbouring users such as the manhattan distance , the pearson corellation which is considered one of the best for the collabertive filtering problem.

3.) In the project we have implemented BFS to obtain the paths from customer to movie in biparte graph. But as the number of movies increases this might be problem as we have to compute more number of paths. We have to explore other ways such as implementing greedy algorithms which are faster and take less computation but may not give an optimal solution. Or we can do iterative deepening search to limit depth of the graph search. This is something that would very efficient and intresting to try.

4.) As for the testing it would be more realistic to have data where we have information about what movies did the customer/user choose after the recommendation. This way we can implement precission and recall metircs easily.

## References:

1.) https://en.wikipedia.org/wiki/Recommender_system
2.) https://en.wikipedia.org/wiki/Euclidean_distance
3.) GRAPH-BASED ANALYSIS FOR E-COMMERCE RECOMMENDATION[http://arizona.openrepository.com/arizona/bitstream/10150/196109/1/azu_etd_1167_sip1_m.pdf]
4.) Collaborative Filtering using Weighted BiPartite Graph Projection [http://snap.stanford.edu/class/cs224w-2013/projects2013/cs224w-038-final.pdf]

5.)Movie Recommendation based on graph traversal Algorithms[http://www2.fiit.stuba.sk/~bielik/publ/abstracts/2013/televido-dexa2013.pdf]
6.) http://grouplens.org/blog/