Weight-Based Personalized Recommendation using Egocentric and Collaborative Filtering

Vijesh M
Student, Department of Computer Science
PES Institute of Technology
Bengaluru, India
mv.vijesh@gmail.com

Sandeep Raju P
Student, Department of Computer Science
PES Institute of Technology
Bengaluru, India
sandeep080@gmail.com

Vijay Mahantesh SM
Student, Department of Computer Science
PES Institute of Technology
Bengaluru, India
vijaym123@gmail.com

Pallavi Karanth
Researcher
Department of Ontology
PES Institute of Technology
Bengaluru, India
pallavik@pes.edu

ABSTRACT
Personalized Recommendations serve as an important ingredient for several web based systems. These systems generally house a knowledge base containing the metadata about items and users. In this paper, we present an approach for the purpose of generating personalized recommendations to users. We've developed a graph based solution that establishes relations between items and performs unsupervised dimensionality reduction on the dataset. Alongside dimensionality reduction, we perform user profiling to determine the relative importance that a particular user gives to individual attributes. We define a characteristic parameter for each user that depicts how egocenric the user is, in choosing the items to consume. We split the process of recommendation into three stages, egocentric recommendation, collaborative filtering and hybridization of the results as a weighted combination using the egocenric behavior of the user as the weight. In our experiments, we have used the MovieLens dataset in combination with the metadata about the movies from IMDB. For experimental purposes, we split the aggregated dataset into two parts: training set and the test set, comprising of 70 percent and 30 percent of the user data respectively. Finally, from the hybridized scores thus obtained, it is possible to predict the probable rating that a user might give to a movie. For the purpose of evaluation of the results, we use Mean Absolute Error as a metric.

Keywords : *unsupervised dimensionality reduction, MovieLens dataset, Graph based egocentric recommendation and collaborative recommendation*

1. INTRODUCTION
The outburst of information on the Web has necessitated the development of effective techniques to automatically filter and organize content that are relevant to the users. Efforts have been put forth to automate the filtering and organization process. In this regard, many Recommender Systems have been developed and evaluated for performance and accuracy, with inherent tradeoffs. A general purpose recommender system for a web application fetches the information about the items from the user and recommends the items that the user is likely to be interested in. Till date, hundreds of sites implement recommender systems to serve their customer bases. One of the most effective techniques used in these systems is Collaborative Filtering. The scope for research in Recommendation Systems grew in the 1990s with the advent of early papers on Collaborative Filtering [9, 15, 6].

Gradually, over the last decade, web systems have moved towards more personalized recommendations for enhanced user experience [4, 1]. In this paper, we assume a setting where we have a single item type and each item is associated with a set of attributes. Each user is expected to have prior opinions on a set of items that the user has consumed. In real world, each user gives varied levels of importance towards various attributes of an item. This varied levels of importance is a crucial decisive factor for the user in choosing the items to consume. Naturally, the users tend to consume those items having the attribute for which the user gives a high importance. Hence, during recommendation, all the attributes cannot be treated as equals.

We implement this setting as an item graph, where each node is an item and each edge represents the relation between items, described in terms of common attribute-value pairs. We describe two approaches for recommendation: egocentric and collaborative filtering. Our approach for collaborative filtering demands the clustering of users which, in turn, depends on the similarity between users. The simple approach for Jaccard similarity is ignorant towards the properties of the items themselves. This arises the need to modify the simple Jaccard similarity to consider the properties of users and items. For the sake of simplicity, we consider a homogeneous dataset in our experiments. Ideally, the approach can also be extended to a heterogeneous set of items that share common attributes.

The sectional breakup of the paper is as follows. Section 2 contains the related work. In section 3, we introduce some of the fundamental notations that we use in the subsequent sections. Section 4 explains the organization of the dataset that we have used. In section 5, we deal with the creation of reference structures. Section 5.1 explains the creation of item graph, section 5.2 explains the creation of profiles for each user and section 5.3 provides a method for unsupervised dimensionality reduction for the dataset. In sections 5.4 and 5.5, we find the similarity between users and cluster them. We define the two approaches for recommendation in sections 6.1 and 6.2, and provide a method to combine them in section 6.3. We present the results obtained in section 7 and conclude in section refsec:conclusion.

2. RELATED WORK

Recommendation systems have gained popularity in web based systems since the appearance of papers on collaborative filtering in the 1990s [9, 15, 6]. [9] explains the concept of collaborative filters. They introduce GroupLens as a system for collaborative filtering of netnews, to help people find articles they will like in the huge stream of available articles. They hypothesize that the users who have agreed on a certain aspect in the past will probably agree again. [15] describes a technique for making personalized recommendations to a user based on the similarities between the interest profile of that user and those of other users. They also test and compare four different algorithms for making recommendations using social information filtering. [6] presents an approach for collaborative recommendation where in the history of other users is used in the automation of a social method for informing choices to the user. Their results show that the communal history-of-use data can serve as a powerful resource for use in interfaces.

[8] determines the potential predictive utility for Usenet news. They develop a specially modified news browser that accepts ratings and displays predictions on a 1-5 scale. They compute the predictions using collaborative filtering techniques and compare the results with noncollaborative approaches. [10] describes several algorithms designed for collaborative filtering, including techniques based on correlation coefficients, vector-based similarity calculations, and statistical Bayesian methods. They compare the predictive accuracy of the various methods in a set of representative problem domains.

Over the past decade, web systems have moved towards personalized recommendations for better user experience. [4] describes a tag-based system for personalized recommendation. They propose an approach which extends the basic similarity calculus with external factors such as tag popularity, tag representativeness and the affinity between user and tag. [1] investigates user modeling strategies for inferring personal interest profiles from social web interactions. They analyze individual micro-blogging activities on twitter and compare different strategies for creating user profiles based on the twitter messages. [13] presents a personalization algorithm for recommendation in folksonomies, which relies on hierarchical tag clusters. [11] explores and analyzes different item-based collaborative techniques. They look into different techiniques for computing item-item similarities and various techniques to obtain recommendations from them.

Significant developments in learning using graph data has led to recent advances in recommendation techniques. [2] presents a recommendation algorithm that includes different types of contextual information. They model the browsing process of a user on a movie database by taking random walks over the contextual graph. [5] models personalized tag recommendation as a "query and ranking" problem. They also propose a novel graph-based ranking algorithm for interrelated multi-type objects.

3. NOTATIONS AND NOTIONS

This section describes the common notations and notions that we will be using throughout the paper. Items are a set of commodities that are intended to be consumed by users. The set of items are represented by I and an individual item is represented by i. The set of users are represented by U. Each user u is associated with a set of items and their corresponding ratings, $R_u = \{(i_1, r_1), (i_2, r_2), (i_3, r_3), ...(i_n, r_n)\}$. Each item is associated with a set of properties called attributes, represented by $a_i = \{a_1, a_2, a_3...a_n\}$. An attribute a_p is associated with a set of values $v = \{v_1, v_2, v_3, ...v_n\}$. The rest of the notations that we use are explained as and when the quantities are defined.

4. THE MOVIELENS DATASET AND IMDB

To subject our algorithm to testing on real world data, we have used the dataset from MovieLens. MovieLens is a movie recommendation website, incubated at GroupLens, Department of Computer Science and Engineering at the University of Minnesota. The users are required to sign up and rate the movies in order to receive recommendations. The datasets can be downloaded from *http://www.grouplens.org/node/73*. The user datasets were created by randomly sampling users who have rated atleast 20 movies. In this paper, we have used the dataset with 1 Million ratings from 6000 users on 4000 movies. Within this dataset, we have selected enough users to obtain approximately 23000 ratings.

In order to obtain metadata about the movies within the dataset, we have used the API services from *http://imdbapi.org/* and *http://www.omdbapi.com/*. For every movie in the dataset, we issue an appropriate API call to obtain the metadata about the movie. Each movie in the aggregated movie dataset has the following attributes: MovieLens ID, IMDB ID, IMDB Rating, Genres, Language, Title, Country, Directors, Writers, Actors, Run Time, Rating Count and Year of Release. The user base constitutes a list of users, the movies that the user has rated and the corresponding rating.

In the following sections, we present a novel approach for the problem of recommendation. The task of recommending items to users is divided among two processes. The prior process involves the creation of reference structures, preprocessing of raw datasets and the deduction of characteristic parameters. The latter process involves using the reference structures and the characteristic parameters in order to recommend items to users.

5. PREPARATION OF REFERENCE STRUCTURES

In this section, we explain the creation of reference structures and deduce the characteristic parameters of users and items. Throught this section, we illustrate the concepts and procedures by taking the aggregated MovieLens Dataset as an example. Note that the figures shown assert the explained concepts. They are generated using the reduced adaptations of the actual data that we use in the implementation of the algorithm.

5.1 Building the Item Graph

An item is associated with a number of attributes. Each of those attributes can be associated with a single value or multiple values. Consider any two items i_1 and i_2. It is likely that i_1 and i_2 have the same set of attributes, but the value(s) for those attributes differ. Higher the number of common attribute-value pairs between i_1 and i_2, more similar i_1 and i_2 are.

In order to represent such relation between the items, we use the graph data structure. Every node in the graph represents an item. The attributes of the items are mapped to the attributes of the nodes. An attribute of a node can be associated with a single value or a list of values. An edge between two nodes conveys that the two items are related. The attributes of the edges are the common attributes between the end nodes. Throughout the paper, we refer to the item graph as G.

The entire item dataset I is transformed into item graph G by creating a node for each item i and assigning the attributes a_i to the created node. Each node is checked against every other node for a list of common attribute-value pairs. An edge is drawn between the two nodes if there is atleast one common attribute-value. The common attribute-value pairs is assigned as the property of the edge. Algorithm 1 illustrates the procedure explained above.

Figures 1 and 2 illustrate how G is structured for the MovieLens dataset.

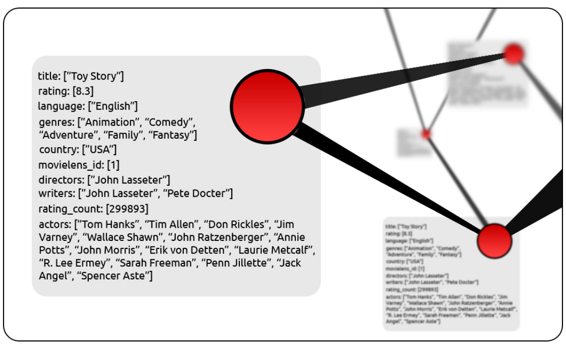

Figure 1: Each item is represented by a node. The attributes of the item form the properties of the node.

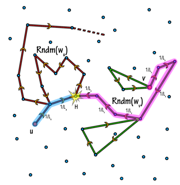

Figure 2: Items are connected to one another by edges. The attributes of the edges are the common attribute-value pairs between the end nodes.

Input: set of items I
Output: item graph $G(V, E)$
Initialize G as an empty graph
for all items i in I **do**
  create a node n_i in G
  assign the attributes of the item as the attributes of the node: $n_i \longleftarrow a_i$
  for all other nodes n_j in G **do**
    compute common attribute-value pairs *common*
    if $common \neq \phi$ **then**
      add and edge (n_i, n_j) to the graph G
      set the property of edge (n_i, n_j) to *common*
    end if
  end for
end for

Algorithm 1: Building the Item Graph

5.2 Generating User Profiles

A user is an entity who consumes an item. In this section, we build a profile for each user. The user profile consists of characteristic parameters of the user that quantifies his behaviour.

In real world, many factors influence the decision of the user to consume a particular item. These factors include the personal choices of the user and recommendation by other users. This scenario is analogous to a customer at a restaurant. The customer might place an order due to recommendations by his friends. His decision is also influenced by what he personally prefers to eat. Generally, the customer does not have equal preferences towards all properties of the food, such as sour, salt, hot and sweet. He has varied levels of liking towards various attributes. Also, the customer might choose to place the order according to his preference or others' recommendation, with a certain probability. We generalize and quantify these behaviours of the customer in order to deduce the characteristic parameters of the customer.

It is easy to see that users behave in a similar manner, regardless of the type of item. We define two characteristic parameters associated with every user. In general, a user does not have equal preference towards all the attributes of the item. The preferences can be modeled as the weight that a user gives for each attribute of the item dataset. This forms the first characteristic parameter w_{u_p}. While consuming an item, the user might decide upon the item purely based on his preferences or follow others' recommendations. We define α_{u_p} to be the probability with which the user u_p decides upon an item purely based on his preferences. α_{u_p} quantifies how *egocentric* a user is. Closer the value of α_{u_p}

to 1, more egocentric the user is. The user follows others' recommendation with a probability $1 - \alpha_{u_p}$. This forms the second characteristic parameter.

$$userProfile_{u_p} = \{w_{u_p}, \alpha_{u_p}\}$$

In order to deduce the w_{u_p} for a user $u_p \in U$, we construct an induced subgraph of the items that u_p has rated. The properties of the edge capture the attribute-value pairs that are common between the end nodes. We hypothesize that, higher the importance that a user gives for an attribute, higher the frequency of its existance as an edge property. Conversely, the frequency of occurance of attribute as a property of the edges is proportional to the importance that the user gives. In order to determine the relative importance, we divide each frequency by the sum of frequencies.

Algorithm 2 illustrates the methodology that we have developed to determine w_{u_p} for a user $u_p \in U$.

Input: set of users, items that the user has consumed and its corresponding rating, U
item graph, G
Output: set of user profiles, $userProfiles = \{userProfile_{u_1}, userProfile_{u_2}, ...userProfile_{u_n}\}$

Initialize an empty associative array $userProfiles$
for all user u in U **do**
  Initialize an empty associative array D
  to hold the attribute and the corresponding count
  Construct an induced subgraph $isg \in G$ of all the items that u is associated with
  for all edge $E \in isg$ **do**
    for all attribute $a \in E$ **do**
      if D contains a **then**
        increment $D[a]$
      else
        $D[a] = 1$
      end if
    end for
  end for
  Compute the relative frequency of attributes by normalization.
  sum = sum of values of D
  for all attribute a in D **do**
    $D[a] = D[a]/sum$
  end for
  $userProfiles[u] = \{D, 0.5\}$
end for

Algorithm 2: Creating User Profiles

Note that we have set the value of α_{u_p} to 0.5 for all the users initially. This is because we are still in the bootstrapping stage. We cannot determine how egocentric a user is at this stage. Such behavior can only be determined when the system is able to receive feedback from the user. Figure 3 shows the relative importance of attributes for users 2987, 3552 and 5997 from the MovieLens Dataset.

5.3 Dimensionality Reduction for Items

Dimensionality Reduction is a well studied problem in recommender systems. Several methods, such as Latent Semantic Indexing [12] and Principal Component Analysis [3] etc, have been formulated to reduce the dimensions of the datasets. In this section, we present an empirical approach to perform dimensionality reduction on the dataset. We observe the users' pattern in consuming the items and deduce the importance of an attribute.

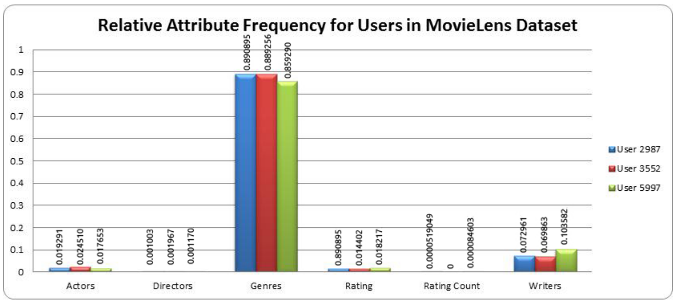

Figure 3: The bar graph indicates the weights that users 2987, 3552 and 5997 (from the MovieLens Dataset) have given to the corresponding attributes.

The importance of an attribute in the item dataset increases as number of users who give higher relative importance to that attribute increases. Hence, the importance of an attribute a in the item set I can be quantified as the mean relative importance of a, taken over all the users $u \in U$. In the previous section, we created a profile for each user. The characteristic parameter w_{u_p} consists of relative weights of attributes for u_p. The relative importance of an attribute a in the item set I can be computed as follows:

$$weight_a = \frac{\sum_{u_p \in U} w_{u_p}[a]}{Number\ of\ Users}$$

The dimensions of the dataset can then be reduced either by thresholding or selecting the top N dimensions with highest weights. Figure 4 shows the weights for various dimensions of the aggregated MovieLens Dataset.

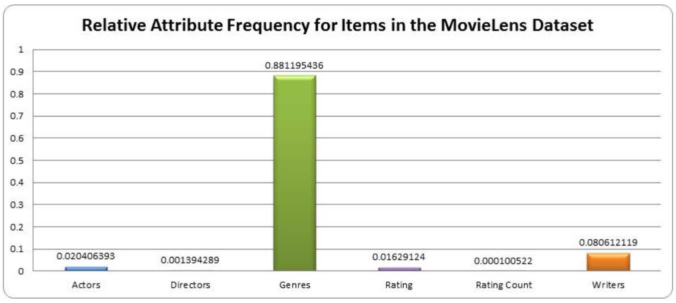

Figure 4: The figure indicates the weights for various dimensions of the aggregated MovieLens Dataset, calculated using the above formula.

5.4 Computing Similarity between Users

The techniques involved in Collaborative Filtering are based on analyzing a large amount of data on users' activities and predicting what the a given user might like. The recommendations can be based on similarity between users or similarity between items. Our algorithm uses similarity between users to perform collaborative filtering. Some of the popular similarity measures are Euclidean Distance, Cosine Similarity, Pearson Correlation and Jaccard Similarity. [14]

presents a comparative study of the impact of these similarity measures on web page clustering.

Jaccard similarity is a measure used for comparing the similarity of sample sets. Jaccard similarity between two sets A and B is mathematically defined as follows:
$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$
Each user is associated with a set of items that he has consumed. The similarity between users can be computed just by applying the simple Jaccard similarity between the sets of items for a pair of users. But there is an intrinsic drawback associated with this approach. The approach treats all the items to be equivalent. It is ignorant towards the properties of the items themselves and the relations between them. To include these considerations, we modify the simple formula for Jaccard similarity.

The idea behind the modification is as follows. To compute the similarity between two users u_p and u_q, we compute two induced subgraphs of G. One of them contains the intersecting items of the two users, and the other contains the union of items of the two users. From the two induced subgraphs, we make two aggregated lists of edge attributes, one for each subgraph. Note that the attributes in these lists can repeat. The similarity is calculated as a fraction. To compute the numerator, we iterate through the attributes in the aggregated list for item intersection subgraph. Counting the number of occurances of these attributes would implicitly mean that we are considering all the attribute as equals. But, the users give varied levels of importance to the attributes. Hence, for each attribute in the aggregated list, we sum up the corresponding weights from the users u_p and u_q and multiply the sum by the relative frequency of occurance of the attribute in the aggregated list. We follow a similar procedure to compute the denominator, with the only difference being that we consider the item union subgraph.

$Intrscn = [a \mid a \in edge\ attribute\ of\ subgraph(items\ of\ u_p \cap items\ of\ u_q, G)]$

$Union = [a \mid a \in edge\ attribute\ of\ subgraph(items\ of\ u_p \cup items\ of\ u_q, G)]$

$$sim(u_p,u_q) = \frac{\sum(w_{u_p}[a]\ +\ w_{u_q}[a])*relFreq(a,Intrscn)\ \forall\ a\ \in\ set(Intrscn))}{\sum(w_{u_p}[a]\ +\ w_{u_q}[a])*relFreq(a,Union)\forall\ a\ \in\ set(Union))}$$

where $relFreq(attrib, list)$ returns the relative frequency of occurance of $attrib$ in $list$.

Algorithm 3 illustrates the computation of similarity between all pairs of users $(u_p, u_q) \in U$. Note that U_{sim} is symmetric in nature. Hence, the amount computation can be reduced approximately by half to make the implementation more efficient.

Input: set of users, items that the user has consumed and its corresponding rating, U
item graph, G
Output: User similarity matrix, U_{sim}
Number of users $N = |U|$
Initialize an empty matrix U_{sim} of size $N \times N$, filled with zeros.
for all user $u_p \in U$ **do**
  for all user $u_q \in U$ **do**
    $Intersecting\_Items$ = set of intersecting items for u_p and u_q
    $Intersecting\_Subgraph = subgraph(Intersecting_Items, G)$
    $Intersecting\_Attribs = [\ ]$
    for all edge $E \in Intersecting\_Subgraph$ **do**
      append the attributes of E to $Intersecting\_Attribs$
    end for
    $numerator = 0.0$
    for all attribute a in set($Intersecting\_Attribs$) **do**
      $numerator += (\ w_{u_p}[a]\ +\ w_{u_q}[a]\ )*$ relative frequency of a in $Intersecting\_Attribs$
    end for

    $Union\_Items$ = set of union of items for u_p and u_q
    $Union\_Subgraph = subgraph(Union_Items, G)$
    $Union\_Attribs = [\ ]$
    for all edge $E \in Union\_Subgraph$ **do**
      append the attributes of E to $Union\_Attribs$
    end for
    $denominator = 0.0$
    for all attribute a in set($Union\_Attribs$) **do**
      $denominator += (\ w_{u_p}[a]\ +\ w_{u_q}[a]\ )*$ relative frequency of a in $Union\_Attribs$
    end for
    $U_{sim}[u_p][u_q] = numerator/denominator$
  end for
end for

Algorithm 3: Computing User Similarity

5.5 Clustering the Users for better Recommendation

6. RECOMMENDATION ALGORITHM

In section 5, we created a set of reference structures that will aid us in recommending items to users. In the subsequent sections, we will use these reference structures to recommend new items to users. We present two approaches of recommending items to users. The Egocentric approach considers only the history of the user in order to recommend new items. The Collaborative Filtering approach considers the clusters to which the user belongs and his similarity with the other users in the cluster to provide recommendation. The recommendations from both the approaches are then combined to produce a hybrid recommendation.

6.1 Egocentric Recommendation

The egocentric algorithm works as follows. We initialize the similarity scores of all the items to zero. Given a user u, we analyze the list of items that u has previously consumed. If a user has consumed an item i, it is very likely that the user might be interested to consume the neighbors of i, but not to equal extents. The extent to which the user might be interested to consume a neighbor not only depends upon the similarity between the items, but also upon the weight that the user gives to the common features between the items. The common features between the items are captured as the property of the edge between them. Hence, we increment the score of each neighbor by the weight that the user gives to each property of the edge. After updating all the nodes, we create an associative array of items and their score. Note that the associative array does not contain the items that are already consumed by the user. We then normalize the scores by dividing all the scores by the maximum score. The pseudocode for the algorithm is presented in Algorithm 4.

Input: user u
item graph G
user base U
Output: Items and their corresponding egocentric scores, *scores_ego*

Initialize the similarity scores of all the nodes in G to 0.
for all item $i \in U[u]$ **do**
  for all neighbors n_i of i in G **do**
    for all attribute a of edge (n_i, i) **do**
      Increment the score of n_i by $w_u[a]$
    end for
  end for
end for
Initialize an empty associative array, *scores_ego* $= \{\}$
for all node $i \in G$ and $i \notin U[u]$ **do**
  scores_ego$[i]$ = similarity score of the node i
end for
Normalize the scores by dividing all the scores by the maximum score
Sort *scores_ego* in the decreasing order of scores

Algorithm 4: Egocentric Recommendation

6.2 Collaborative Filtering based on User Similarity

Collaborative Filtering(CF) is a collection of techniques that are used in recommending items to users based on the ratings of other users. CF techniques are by far the most successful techniques in recommending items to users. The core idea behind CF techniques is the concept of similarity. The recommender systems generally use two forms of similarities: user-based similarities[quote something] and item-based similarities[quote something]. These techniques generally use a standard metric, such as Euclidean Distance, Cosine Similarity, Pearson Correlation or Jaccard Similarity[quote something]. Once a similarity matrix is build, these systems use various algorithms to generate recommendations.

In this paper, we have used an extended version of Jaccard similarity, as described in section 5.4. The algorithm that we have developed uses user-based similarity. Our algorithm begins by iterating through all the clusters. A cluster houses many users. Each user is associated with a list of items that he has consumed. For each user u_p in each cluster c, we increment the score of each item i consumed by u_p, by the similarity between u and u_p. Note that the clusters we obtained are overlapping. That is, a single user can belong to multiple clusters. If the given user u_1 is very similar to another user u_2, then u_2 will appear in majority of the clusters where u_1 is present. This implies that the score for the items consumed by u_2 is naturally boosted, since u_2 is presented in most of u_1's clusters and the similarity between u_1 and u_2 is high. The pseudocode for the algorithm is presented in Algorithm 5.

Input: user u
item graph G
user base U
similarity matrix U_{sim}
clusters C
Output: Items and their corresponding CF scores, *scores_collab*

Initialize the similarity scores of all the nodes in G to 0.
for all cluster $c \in C$ **do**
  for all user $u_p \in c$ **do**
    for all item $i \in U[u_q]$ **do**
      Increment the score of i by $U_{sim}[u][u_p]$
    end for
  end for
end for
Initialize an empty associative array, *scores_collab* $= \{\}$
for all node $i \in G$ and $i \notin U[u]$ **do**
  scores_collab$[i]$ = similarity score of the node i
end for
Normalize the scores by dividing all the scores by the maximum score
Sort *scores_collab* in the decreasing order of scores

Algorithm 5: Collaborative Recommendation based in User Similarity

On sorting *scores_ego* and *scores_collab* in the decreasing order, we get a pure egocenric recommendation list and a pure collaborative recommendation list respectively. In the next subsection, we describe a method to combine *scores_ego* and *scores_collab* to get a hybrid recommendation.

6.3 Hybrid Recommendation

Sections 6.1 and 6.2 dealt with generating egocentric and collaborative recommendations for the user. Both the algorithms associated a real number between 0 and 1 with each item that a user has not consumed. Hybrid Recommendation involves combining *scores_ego* and *scores_collab* into a single list *scores_hybrid*, by means of weighted combination of corresponding items from both the lists.

Since we are merging the lists that signify egocentric behavior and group behavior, we choose α_{u_p} as the weight. In section 5.2, we had defined α_{u_p} as the probability with which the user u_p decides upon an item purely based on his preferences. α_{u_p} quantifies how *egocentric* a user is. Hence, assign a weight α_{u_p} to *scores_ego* and $1 - \alpha_{u_p}$ to *scores_collab*.

We propose two methods by which the lists can be combined:

1. Rank based weighted average:
   In this approach, we take a weighted average of ranks of each item and sort the items in the decreasing order of their composite ranks. As mentioned earlier, we assign a weight α_{u_p} to *scores_ego* and $1 - \alpha_{u_p}$ to *scores_collab*. The following formula mathematically describes the operation:
   $compositeRank_i = \alpha_{u_p} * rank(i, scores\_ego) + (1 - \alpha_{u_p}) * rank(i, scores\_collab)$

2. Score based weighted average:
   In this approach, we take a weighted average of score of each item and sort the items in the decreasing order of their composite scores. As mentioned earlier, we assign a weight α_{u_p} to *scores_ego* and $1 - \alpha_{u_p}$ to *scores_collab*. The following formula mathematically describes the operation:
   $compositeScore_i = \alpha_{u_p} * scores\_ego[i] + (1 - \alpha_{u_p}) * scores\_collab[i]$

Rank based weighted average method can be used to provide recommendations to the user in the order of their ranks. But predicting the rating that the user might give to an item tends to be relatively hard. Consequently, evaluating the accuracy of our algorithm also tends to be relatively hard. Hence, in our experiments, we use the score based weighted average method to predict the rating that a user might give to a new item.

7. RESULTS AND DISCUSSIONS

7.1 Dataset

As mentioned in section 4, we have used the aggregated MovieLens Dataset in our experiments. We have used the dataset with 1 Million ratings from 6000 users on 4000 movies, and retrieved the metadata about the movies from *http://imdbapi.org/* and *http://www.omdbapi.com/*. We split the user dataset into two parts, training data and test data. The training data constitutes 70 percent of the dataset. The users in the training set were chosen randomly. We test the accuracy of our algorithm by predicting the user ratings in the remaining 30 percent of the dataset.

7.2 Evaluation Metric

The performance of recommender systems have been extensively studied in research literature since 1994 [9]. [7] broadly classifies recommendation accuracy metrics into three classes: predictive accuracy metrics, classification accuracy metrics and rank accuracy metrics. In this paper, we have chosen to evaluate our algorithm using a type of predictive accuracy measure called *Mean Absolute Error* (MAE). It measures the average absolute error in the predicted rating and the user's true rating. [10, 15] present some of the cases where MAE has been used to evaluate recommender systems. MAE is mathematically defined as follows:

$|E| = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N}$

7.3 Experimental Results

In this section, we present the results obtained on applying our algorithm to the aggregated MovieLens Dataset. Section 5.1 dealt with the creation of item graph G. Figure **??** is a snapshot of the item graph G. Given two movies from the dataset, just a single common attribute-value pair will result in an edge formation. Hence, the graph that we obtain is fairly dense. After transforming the aggregated MovieLens dataset, we found that G was a very dense graph, with a density of 0.47435, with an average degree of 1842.41.

In section 5.2, we created the profiles for each user, $userProfile_{u_p}$. We also defined two characteristic parameters, w_{u_p} and α_{u_p}, for each user. We provided a methodology to determine w_{u_p} in Algorithm 2.

Figure 5 statistically illustrates the weights for the user 2987 of the MovieLens dataset, for various attributes. In general, it is very likely that the users watch the movies from the same country that they reside in. As far as the dynamics of shooting movies go, the filming locations of a movie are likely to be in the same country that the movie is being produced. In general, users who are familiar with a particular language tend to watch the movies from the same language. Naturally, the weights for country, filming locations and language are relatively high. The fact that there are comparitively limited number of values for a particular also boosts up the relative score. The ratio between relative scores indicates the number of times that a particular attribute is important, relative to the other. From the figure, we can deduce that the user 2987 statistically gives 37.3 times more importance to actors, compared to directors. On similar lines, user 2987 gives 3.5 times more importance to genre, compared to actors.

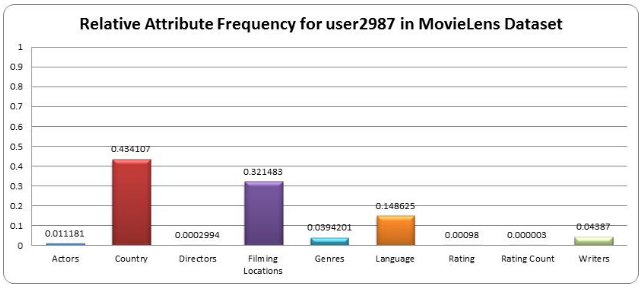

Figure 5: Relative Attribute Importance for the user 2987 of the MovieLens dataset

In section 5.3, we reduced the dimensionality of the dataset based on the relative importances that each user gives to a particular attribute. We apply the same techinique to the aggregated MovieLens dataset. Figure 6 shows the relative attribute importance for the dataset.

We analyze the data for the entire aggregated MovieLens dataset in the same way as we did for user 2987. On an average, we find that the users statistically give 24.2 times more importance to actors in comparison with directors. On similar lines, users give 5.8 times more importance to genres, compared to actors.

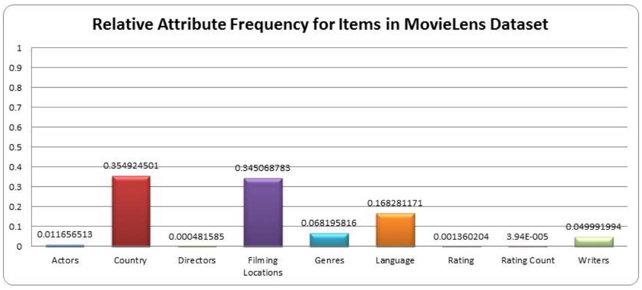

Figure 6: Relative Attribute Importance of the items in the MovieLens dataset

To evaluate the accuracy of our algorithm, we use MAE as the evaluation metric. We use α_{u_p} as a parameter in our experiments. For varying values of α_{u_p}, we predict each user's ratings for all the movies that the user has not rated in the training set. We compare these predicted ratings with the actual ratings present in the test set and compute the MAE. The variation of MAE for various values of α_{u_p} is shown in Figure 7.

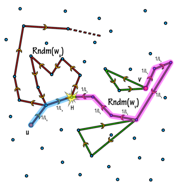

Figure 7: The plot of variation of Mean Absolute Error with varying values of α_{u_p}

Explain what happens at extreme points, i.e, at $\alpha_{u_p} = 0$ and $\alpha_{u_p} = 1$

8. CONCLUSION

In this paper, we demonstrated a novel approach for recommending items to users and performed experiments on the aggregated MovieLens dataset to test our algorithm. We use graph as the core reference structure. We form a graph of items and the relations between items are manifested as properties of the corresponding edges. Based on the users' history of item consumption, we profile each user and deduce the weight that the user gives to the attributes. We use the profiles to statistically determine the importance of each attribute in the dataset, in turn performing dimensionality reduction. To find out the similarity between users, we modified the simple Jaccard similarity to consider the properties of users and items. Based on a similarity threshold, we form overlapping clusters of users. To provide recommendation for a particular user, we follow two distinct approaches: egocentric and collaborative filtering. These recommendations are then combined to form hybrid recommendation.

Our results show that the importance of each dimension of the dataset can be weighted in an unsupervised manner, in accordance with the pattern that the users follow to consume the items. We deduce the relative importance that each user gives to each of the attributes. The recommendations that we provide to a user is personalized based on the corresponding relative importances of the corresponding user. We also show the variation in the Mean Absolute Error of our rating predictions as we increase the egocentric behavior of the user.

9. FUTURE WORK

Our experiments were performed on the MovieLens dataset of an intermediate size, with a restricted number of ratings and users. The tests must be performed on larger datasets, with a wide variety of attributes. The characteristic parameter α_{u_p} represents how egocentric a user is. Determining the value of this parameter calls for a feedback mechanism from the users. At the time of the writing, we are designing the methods to obtain this parameter.

In this paper, we have presented the empirical results that we have obtained on applying our algorithm on the dataset. The theoritical bounds of the algorithm is yet to be formulated. The approach presented in this paper does not enforce all the items to have the same set of attributes, i.e, the attributes need not be homogeneous across all the items. In our experiments, we have chosen a homogeneous dataset. The tests must be performed on heterogeneous datasets, where it is not necessary that all items have the same set of attributes.

10. ACKNOWLEDGEMENTS

The work was conducted at PES Institute of Technology, West Campus, Bengaluru, India. We would like to thank Ms. Pallavi Karanth, for her guidance. We would also like to thank Dr. Kavi Mahesh, for strengthening our understanding of the subject. We would also like to thank Prof. Nitin V Pujari, HOD, Department of Computer Science, PESIT and Dr. KNB Murthy, Principal, PESIT for providing us with an opportunity to carry out the research work. We also thank anonymous reviewers for their valuable comments.

11. REFERENCES

[1] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing temporal dynamics in twitter profiles for personalized recommendations in the social web. In *ACM WebSci'11*, pages 1–8, 2011.

[2] Toine Bogers. Movie recommendation using random walks over the contextual graph. In *CARS'10: Proceedings of the 2nd workshop on context-aware recommender systems*, 2010.

[3] E.E. Cureton and R. B D'Agostino. Factor analysis: An applied approach. Lawrence Erlbaum associate pubs. Hillsdale, NJ, 1983.

[4] Frederico Durao and Peter Dolog. Personalized tag-based recommendation in social web systems. In *CEUR Workshop Proceedings*, volume 485, pages 40–49, 2009.

[5] Ziyu Guan, Jiajun Bu, Qiaozhu Mei, Chun Chen, and Can Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. ACM*, 2009.

[6] W. Hill, Stead L., Rosenstein M., and Furnas G. Recommending and evaluating choices in a virtual

community of use. In *Proceedings of CHI, '95*, 1995.

[7] Jonathan Herlocker J, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. In *ACM Transactions on Information Systems (TOIS) 22.1*, pages 5–53, 2004.

[8] Konstan J., Miller B., Maltz. D., Herlocker J., Gordon L., and Reidl J. Grouplens: Applying collaborative filtering to usenet news. Communications of the ACM, 40(3), pages 77–87, 1997.

[9] Resnick P., Lacovou N., Suchak M., Bergstrom P., and Reidl J. Grouplens: An open architecture for collaborative filtering of netnews. In *proceedings of CSCW '94, Chapel Hill, NC*, 1994.

[10] Breese J S, Heckerman D, , and Kadie C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of 14th Conference on Uncertainty in Aritficial Intelligence*, pages 43–52, 1998.

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *10th international conference on World Wide Web*, pages 285–295, 2001.

[12] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. 2002.

[13] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *ACM conference on Recommender systems*, pages 259–266, 2008.

[14] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. 2000.

[15] Shardanand U. and Maes P. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of CHI '95. Denver CO.*, 1995.