**HED-X PROJECT REPORT**

TEAM NAME : **CHRYSALIS**

TEAM MEMBERS:

* SRINIDHI.M.P
* AISHWARYA.R
* AKSHATHA.R
* PRAVEEN.S
* MYTHRI.N.R

PROJECT TITLE: **MOVIE RECOMMENDATION ENGINE**

**SKILLS/COMPETENCIES REQUIRED**: 1. R Programming

2. Data Mining

3. Clustering

4. Collaborative Filtering

**PROBLEM STATEMENT**: Recommendation Systems have changed the way people find products, information, and even other people. They study patterns of behavior to know what someone will prefer from among a collection of things he has never experienced. The technology behind recommendation systems has evolved into a rich collection of tools that enable the practitioner or researcher to develop effective recommendation .In this project you need to recommend the top 5 movies the user can watch.

**PROJECT DESCRIPTION**: Recommender systems are a subclass of information filtering system that seek to predict the ‘rating’ or ‘preference’ that a user would give to an item. In this project ,recommend the top 5 movies the user can watch. The dataset includes user details ,movie details and user ratings.

**SOFTWARE /HARDWARE REQUIREMENTS** to implement the project:

1 .Hardware:

* Operating System-Windows/Linux(Ubuntu)
* Minimum 4GB RAM
* 3.500 GB Hard disk

2. Software:

* R(<https://cran.r-project.org/>)
* RStudio (<https://www.rstudio.com/products/download/>)

**PROJECT DURATION**: 2 WEEKS

**RESOURCES**: Download the data set from [http://blog.revolutionanalytics.com/2012/04/simple-tools-for-building-a -recommendation-engine.html](http://blog.revolutionanalytics.com/2012/04/simple-tools-for-building-a%20-recommendation-engine.html)>

**ABSTRACT**: Recommender systems are a subclass of information filtering system to predict the rating or preference that a user would give to an item. Recommender systems have become extremely common in recent years, and are utilized in areas such as movies , music, books, research articles ,social tags and products in general. Our Movie Recommendation engine aims to provide the top 5 recommendations to a user based on his ratings for a particular movie. We are using item based collaborative filtering method.

**DESIGN**: The aim of our movie recommendation engine is to recommend the top 5 movies to a user based on his previous ratings or preferences. There are two types of Recommendation algorithms:

1. CONTENT BASED ALGORITHM

2. COLLABORATIVE FILTERING ALGORITHM:

IDEA: If a person A likes item 1,2,3 and B likes 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.

In our project, we are using COLLABORATIVE FILLTERING ALGORITHM.

Again Collaborative Filtering is of 2 types:

1. User-User Collaborative Filtering: Here we find look alike customers (based on similarity) and offer products which first customer’s look alike has chosen in the past .

2. Item-Item Collaborative Filtering: It is similar to the previous algorithm ,but instead of finding customer look alike ,we find user item look alike.

We have used User based collaborative filtering approach which groups users according to prior usage behavior or according to their preferences, and then recommends an item that a similar user in the same group viewed or liked .

For example: If user 1 liked movies A,B and C ,and if user 2 liked movies A and B ,then movie C might make a good recommendation to user 2.

We are using the recommenderlab package available in R.

W are working with 1 million row movie dataset from [www.grouplens.org](http://www.grouplens.org) Three zipped up .dat files comprises this data set. The first file ,ratings.dat, contains User-ID, Movies ID, Rating and Timestamp.

The second file ,users.dat, contains the UserID ,Gender ,Age ,Occupation and Zip-code for each user.

The third file ,movies.dat, contains the Movie-ID, Title and Genre associated with each movie.

IMPLEMENTING USER BASED COLLABORATIVE FILTERING:

This involves 2 steps:

STEP-1.Calculate the similarity score for people helps us to identify similar people. We use Cosine based Similarity function to calculate the similarity between the users .

Cosine Similarity:

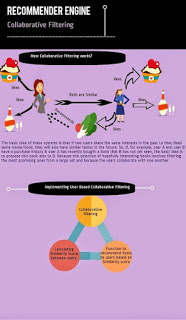
Cosine similarity is a measure of similarity between two vectors of an [inner product space](https://en.wikipedia.org/wiki/Inner_product_space) that measures the [cosine](https://en.wikipedia.org/wiki/Cosine) of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

In R we have a cosine function readily available:  
user\_sim = cosine(as.matrix(t(x)))

STEP2: For recommending movies for a particular user ,using the above similarity matrix, we need to first fill the N/A where he has not rated. As first step, separate the non-rated movies by him and a weighted matrix is created by multiplying user similarity score (user\_sim[,user no) with ratings given by other users.

Next step is to sum up all the columns of the weight matrix, then divide by the sum of all the similarities for critics that reviewed that movie. The result calculation gives what the user might rate this movie.

THIS PROCEDURE CAN BE ILLUSTRATED AS:



**SOURCE CODE FOR THE MOVIE RECOMMENDATION ENGINE**:

library(lsa)

critics = read.csv("movieratings.csv")

#calculate the euclidian distance

#EUD = dist(critics[,2:7])

#cosine similarity calculation

x = critics[,2:7]

x[is.na(x)] = 0

user\_sim = cosine(as.matrix(t(x))) #user similarity

#Recommending items

#create weightge matrix

weight\_mat = user\_sim[,7]\*critics[,2:7]

rec\_itm\_for\_user = function(userNo)

{

#calcualte column wise sum

col\_sums= list()

rat\_user = critics[userNo,2:ncol(critics)]

x=1

tot = list()

z=1

for(i in 1:ncol(rat\_user)){

if(is.na(rat\_user[1,i]))

{

col\_sums[x] = sum(weight\_mat[,i],na.rm=TRUE)

x=x+1

temp = as.data.frame(weight\_mat[,i])

sum\_temp=0

for(j in 1:nrow(temp)){

if(!is.na(temp[j,1])){

sum\_temp = sum\_temp+user\_sim[j,ncol(rat\_user)]

}

}

tot[z] = sum\_temp

z=z+1

}

}

z=NULL

z=1

for(i in 1:ncol(rat\_user)){

if(is.na(rat\_user[1,i]))

{

rat\_user[1,i] = col\_sums[[z]]/tot[[z]]

z=z+1

}

}

return(rat\_user)

}

#to get N recommendations:

rec\_itm\_for\_user(1) #first person recommendations

**CONCLUSION AND CHALLENGES**

Collaborative recommendation algorithm can be powerful engine. In this project we create an engine for recommending top five movies a user is preferred to watch. The algorithm used here uses large data set and finds the optimal set of movies for recommendation using collaborative technique and cosine similarity matrix.

* Since the R programming language and concepts of data science are new to us, we faced difficulties in the initial stages.
* Our first approach to the problem was wrong so we had to come up with a new algorithm in limited time.

ADVANTAGES OF COLLABORATIVE FILTERING

* CF systems do not require content information about neither users or items to be machine-recognizable. Pure CF methods utilize only ratings and do not require any additional information about users or items.
* These systems can make an assessment of quality, style or viewpoint by consideration of other people’s experience.
* The notable advantage is that CF systems can produce personalized recommendations, because they consider other people’s experience and recommendations are based on that experience.
* Another notable advantage is that the CF recommender systems can suggest serendipitous items by observing similar-minded people’s behavior.

DISADVANTAGES OF COLLABORATIVE FILTERING

* The principal disadvantage is that CF systems cannot produce recommendations if there are no ratings available.
* They demonstrate poor accuracy when there is little data about users’ ratings. This and previous disadvantage are called the above-mentioned Cold-Start problem.
* Another principal disadvantage is that CF systems are not content aware meaning that information about items are not considered when they produce recommendations.
* Many of existing CF algorithms work slowly on a huge amount of data. Several techniques such as clustering [Pham 2011] and parallelization [Recht 2011] were discovered to overcome the problem.
* Lack of heterophilious diffusion, where individuals seek recommendations from more advanced peers unlike them.

**REFERENCES**

* <http://www.r-bloggers.com>
* <http://bigdata-madesimple.com>
* <http://www.stackoverflow.com>
* <http://www.muffynomster.wordpress.com>