Realization of A Hybrid Recommender System

Wenxue Zhang, Chenxi He, Shengqian Chen

March 2016

Abstract

The application of recommendation systems has attracted considerable attention in recent years. The most traditional techniques are content-based and collaborative filtering, and both of them have certain disadvantages which can be solved by a combination of themselves. We therefore develop a hybrid model based on Bayesian network. The switch of two recommendation mechanism is determined by probability calculated in the Bayesian network[7]. The model is then tested using the Movie Lens 100K data sets. We choose the R package recommenderlab as our benchmark.

1 General Description

Collaborative filtering method is based on collecting and analyzing a large amount of information on users' behaviors, 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 content and therefore it is capable of accurately recommending movies without requiring an "understanding" of the movie itself. It is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past.

For content based filtering, the algorithm tries to recommend movies that are similar to those that a user liked in the past. In particular, various candidate movies are compared with movies previously rated by the user and the best-matching movies are recommended.

What if we combine the best part of the two filtering? Hybrid approaches can be implemented in several ways, here we use Bayesian network to deal with it.

Here we represents how user $u \in U = \{u_1, \cdots, u_n\}$, movie $m \in M = \{m_1, \cdots, m_l\}$ and movie features $f \in F = \{f_1, \cdots, f_m\}$ are related in the Bayesian Network model. The input data of the model should be two matrix D and R. D is a sparse binary $l \times m$ matrix, with entry $d_{i,j} = 1$ indicating that movie i can be described by feature j and zero otherwise. R is the rating matrix on a scale from 1 to 5, with entry $u_{i,j} = k$ indicating that movie j is rated k by user i.

1.1 Content-Based Part

We consider the logical implication as below: each movie is described by a set of features. There should be an arc between movie and any features related to the film in the network. So in other word the probability of relevance of the movie will depend on the relevance value of the features that describe it. In this way two movies that share common features are supposed to be dependent.[3]

1.2 Collaborative Filtering Part

This part [6]aims to relate a given user with other users with similar tastes. Those relevant users here are used to estimate the probability distributions of the given user's rating prediction. The parent set of the target user will be learned from the rating matrix R. In this paper we measure the similarity as below:

$$sim(u_i, u_j) = |PC(u_i, u_j)| \frac{|I(u_i) \bigcap I(u_j)|}{|I(u_i)|}$$

where PC is Pearson correlation and I is the set of movies rated by the user in the data set. The second part penalizes highly correlated users that are based on a few co-rated movies.

We use a fixed neighbourhood as selecting top-N most similar users, which is a traditional memory-based recommendation system method.

1.3 Hybrid Part

We use a user variable U^{CB} to denote content-based prediction. Similarly U^{CF} is the collaborative prediction. In the hybrid level we use U^H. For the target user, now we have his or her preference of target movie in two approaches U^{CB} and U^{CF}. These two preferences then are combined[1] in one node U^H to get the final prediction for user. And clearly U^{CB} and U^{CF} should be the parent node of U^H.

2 Bayesian Networks

We use Bayesian network formalism to represent the relationships among users U, movies M and features F. By using Bayesian network, for example, we can express the dependence and independence relationships of users and movies, and probability distribution which measures the strength of these relationships.A node is independent of all its ancestors given that the values of its parents are known.

In this hybrid model, this model consists of three node layers(feature, movie and user's layer). Starting from the first layer in the Bayesian network, the distributions of one layer are obtained using the posterior probabilities computed in the previous one. In the first layer it's the feature node F_k. It should be a binary random variable determined by whether the k-th feature is related to the movie we want to predict or not. It takes value from $\{f_{k,0}, f_{k,1}\}$. In Second layer it's the movie node M_j, and similarly it takes value from $\{m_{j,0}, m_{j,1}\}$, meaning that the movie is not relevant or is relevant for predicting the target movie. Then the bottom layer is the user node u_i. It takes value from rating scale $\{0, 1, ..., 5\}$ and 0. Here the probability of $u_i = 0$ indicates that user has no useful information for predicting the target item's rating.

3 Probability Distribution Computation

Given the large number of users and movies, the estimation of probability distribution can be complex.It will be hard to compute and store the conditional probability table. So we propose the use of canonical model, which is helpful for a efficient inference procedure. In other word, conditional probability can be calculated using the canonical weighted sum as illustrated in part6. The following is a brief introduction for canonical weighted sum and theorem that will be used later.

3.1 Canonical Weighted Sum

Given node X_i in Bayesian network and there are j values X_i can take, let $pa(X_i)$ be the parent set of X_i, and let Y_k be the k-th parent node in the network, the conditional probability distributions can be stated as below

$$P(x_{i,j}|pa(X_i)) = \sum_{Y_k \in pa(X_i)} w(y_{k,l}, x_{i,j}),$$

where $y_{k,l}$ is the value variable Y_k takes, and $w(y_{k,l}, x_{i,j})$ is the weight measuring how Y_k taking l-th value describes the j-th state of node X_i. Obviously the weight should be non-negative values and satisfies

$$\sum_{j=1}^{r} \sum_{Y_k \in pa(X_i)} w(y_{k,l}, x_{i,j}) = 1$$

3.2 Theorem1

X_a is a node in a Bayesian network, m_{X_a} is the number of parents of X_a, Y_j is a node in $pa(X_a)$, and l_{Y_j} is the number of states taken by Y_j. The posterior probability can be expressed using the following formula:

$$p(x_{a,s}|z) = \sum_{j=1}^{m_{X_a}} \sum_{k=1}^{l_{Y_j}} w(y_{j,k}, x_{a,s}) p(y_{j,k}|z)$$

.

Here z can be the target user or movie. In content-based part, z is the movie itself. The probability $p(F_k|m_j)$ must be computed for each feature node F_k linked to the target item m_j.

4 Basic Algorithm

As we said we will use the canonical weighted sum model to model movie and user variables. We then factorize the conditional probability tables into a set of weights and use an additive criterion to combine these values.

4.1 Content-based Propagation

Here we assume all features are equally probable, $p(f_{k,1}) = \frac{1}{l}$, where l is the size of set F.

We now try to calculate $p(m_{j,1}|pa(M_j))$. Given $p(f_{k,1})$,we are able to obtain $p(f_{k,1}|m_j)$. Here the canonical weight is calculated as $w(f_{k,1}, m_j) = \frac{1}{N(m_j)} log((\frac{m}{n_k}) + 1)$, $N(m_j) = \sum_{F_k \in pa(m_j)} log((\frac{m}{n_k}) + 1)$, where n_k is the number of feature f_k is used to describe an movie, and m is the number of movies in total. It's actually called inverted document frequency, which is used to measure term's importance.Then, according to theorem 1, we now can calculate $p(m_k|m_j)$ for different movies. Again, based on theorem 1, we can propagate to U^{CB} as defined before. Generally speaking, what we do here is to calculate the probability distribution based on the parent layer, which is stated in the theorem. And the probability is calculated based on the canonical weight which is also detailed in part6.

4.2 Collaborative Propagation

In the collaborative level, the weight is supposed to reflect the contribution of each similar user in the prediction of the rating of target user. According to theorem 1, similarly, we can propagate to node U^{CF}. The calculation is detailed later in the example part of the paper.

4.3 Hybrid System

As described before, U^H has two parent nodes as U^{CF} and U^{CB}. The probability $p(U^H|U^{CF}, U^{CB})$ represent how to combine both types of information when predicting the rating for movie.

We propose a parameter α[4] for each movie, on a scale from 0 to 1, be used to control the selecting mechanism of two previous parts. For a given user u, u_s^H means that at hybrid level he or she will rate the target movie s. Similarly we define the variable u_s^{CB} and u_s^{CF}. Here s is of rating scale from 1 to 5.

$$Pr(u_s^H|u_s^{CB}, u_s^{CF}) = 1$$
$$Pr(u_s^H|u_s^{CB}, u_s^{CF}) = \alpha \text{ if } q \neq s$$
$$Pr(u_s^H|u_s^{CB}, u_s^{CF})1 - \alpha \text{ if } t \neq s$$
$$Pr(u_s^H|u_s^{CB}, u_s^{CF}) = 0 \text{ if } t, q \neq s$$

Based on the definition above, the higher α is, the more weights is on content-based node. With $\alpha = 1$ it will behave as a content-based model, while $\alpha = 0$ it will behave as a collaborative model. So now the key question is that how we should decide the parameter α.It's a direct reflection of our confidence in collaborative recommendation, and thus it can be used to determine the extent to which we should believe in content-based and collaborative level in the hybrid procedure. Considering the higher value of α, the more weight we will put in the content-based part. In this paper we use $\alpha = p(u_0^{CF})^2$.

5 Evaluation of the Hybrid Recommendation System

5.1 Data Sets

In this paper we use the Movie Lens 100K data set. There are 1682 movies rated by 943 users on a scale of 1 to 5 and there are 10000 ratings in total. Besides, all 1682 movies are classified into 19 genres (Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, Unknown). And thus it allows us to perform content based recommendation.

5.2 Evaluation Measures

In order to measure how accurate the result of this model is, we use the mean absolute value to quantify how much different the predicted rate is from the user's true rate. We can also choose R library "recommenderlab" as the benchmark of the MAE.[5]

$$MAE = \frac{\sum_{i=1}^{N} abs(p_i - r_i)}{N},$$

where N is the number of cases, p_i is the predicted rate and r_i is the real rate. Other than that, we can also measure the percentage of times the system correctly predict the rating denoted by C, which as calculated as below:

$$C = \frac{\sum_{i=1}^{N} Id(p_i, r_i)}{N},$$

where id(x,y) returns 1 if x=y and 0 otherwise.

6 Algorithms with Example

Here I will illustrate how our algorithms works step by step.
For example:

Table1	Feature1	Feature2	Feature3	Feature4
Movie1	1	0	0	1
Movie2	0	1	1	1
Movie3	1	1	0	1
Movie4	1	1	1	1
Movie5	1	0	0	1

Table2	Movie1	Movie2	Movie3	Movie4	Movie5
User1	0	3	5	1	4
User2	2	2	4	1	0
User3	1	5	1	4	4

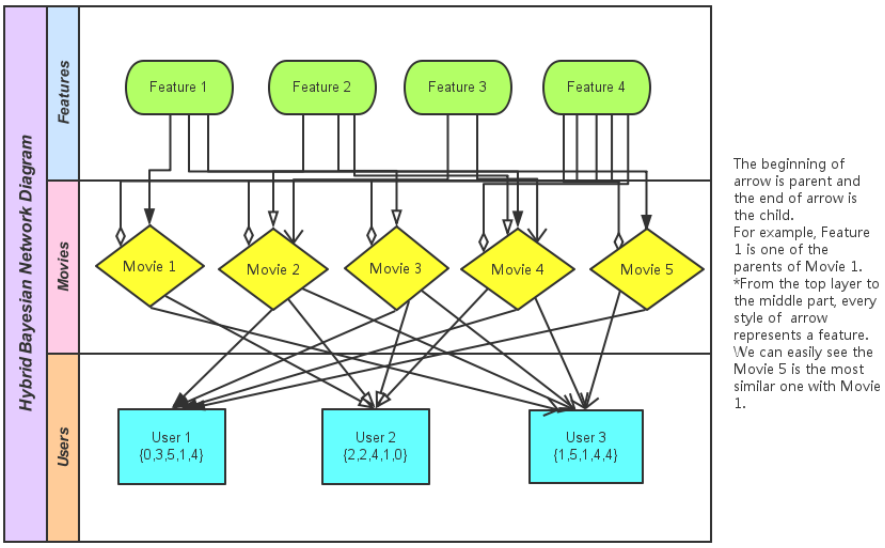

Figure 1: Hybrid Bayesian Network

From Table2, we can see that User1 didn't rate Movie1. Thus, how the User1 will rate Movie1 is our interest.

In our example, there exists 5 movies $\mathscr{M} = \{M_1, M_2, ...M_5\}$, which are described by 4 features $\mathscr{F} = \{F_1, F_2, ...F_4\}$, and each movie is specified by one or several. There are also 3 users $\mathscr{U} = \{U_1, U_2, U_3\}$ and for each user, a set of ratings about certain observed movies in $\mathscr{M}$.

Step 1 Content-based propagation:

- $ev_{cb} = m_{1,1}$ set $Pr(m_{1,1} \mid m_{1,1}) = 1$
- Compute $Pr(F_k \mid ev_{cb})$ for k=1,2,3,4
  Here $Pa(M_1) = \{F_1, F_4\}$

$$Pr(f_{k,1} \mid m_{1,1}) = \begin{cases} Pr(f_{k,1}) & \text{if } F_k \notin Pa(M_1) \\ Pr(f_{k,1}) + \frac{w(f_{k,1},m_{1,1})Pr(f_{k,1})(1-Pr(f_{k,1}))}{Pr(m_{1,1})} & \text{if } F_k \in Pa(M_1) \end{cases}$$

where $Pr(f_{k,1}) = \frac{1}{4}$ since there are 4 features in total.
and the denominator

$$\begin{aligned} Pr(m_{1,1}) &= \sum_{F_k \in Pa(M_1)} w(f_{k,1}, m_{1,1}) Pr(f_{k,1}) \\ &= w(f_{1,1}, m_{1,1}) Pr(f_{1,1}) + w(f_{4,1}, m_{1,1}) Pr(f_{4,1}) \end{aligned}$$

The canonical weights are calculated as $w(f_{k,1}, m_{1,1}) = \frac{1}{M(M_1)} log((\frac{m}{n_k}) + 1) = \frac{1}{M(M_1)} log((\frac{5}{n_k}) + 1)$, n_k:the number of times that feature F_k has been used to describe an item, and $w(f_{k,0}, m_{1,1}) = 0$ for future use. $M(M_1) = \sum_{F_k \in Pa(M_1)} log((\frac{5}{n_k}) + 1)$

In our example:

Probability of relevance	Feature1	Feature2	Feature3	Feature4
Item1	0.654	0.250	0.250	0.596

- Propagate to movies.

$$\begin{aligned} Pr(m_{k,1} \mid m_{1,1}) &= \sum_{j=1}^{m_{m_k}} \sum_{s=0}^{1} w(f_{j,s}, m_{k,1}) Pr(f_{j,s} \mid m_{1,1}) \\ &= \sum_{j=1}^{m_{m_k}} w(f_{j,1}, m_{k,1}) Pr(f_{j,s} \mid m_{1,1}) \end{aligned}$$

where m_{m_k} be the number of parents of M_k and F_j be a node in $Pa(M_k)$
Note we have set $Pr(m_{1,1} \mid m_{1,1}) = 1$ before. Then, we got:

Probability of relevance	Movie1	Movie2	Movie3	Movie4	Movie5
Item1	1	0.332	0.478	0.402	0.627

- Propagate to U_1.

$$Pr(u_{1,r} \mid m_{1,1}) = \sum_{k=1}^{m_{u_1}} \sum_{s=0}^{1} w(m_{k,s}, u_{1,r}) Pr(m_{k,s} \mid m_{1,1})$$

where m_{u_1} be the number of parents of U_1 and M_k be a node in $Pa(U_1)$

The canonical weights are calculated as

$$\begin{aligned} w(m_{k,1}, u_{1,r^\star}) &= \frac{1}{4} \text{ since } U_1 \text{ has rated 4 movies. } r^\star \text{ is the rate we have} \\ w(m_{k,1}, u_{1,r}) &= 0 \text{ if } r \neq r^\star \\ w(m_{k,0}, u_{1,0}) &= \frac{1}{4} \\ w(m_{k,0}, u_{1,r}) &= 0 \end{aligned}$$

Note: $Pr(m_{k,0} \mid m_{1,1}) = 1 - Pr(m_{k,1} \mid m_{1,1})$. Then we got:

Probability of rate(CB)	0	1	2	3	4	5
User1	0.540	0.100	0.000	0.083	0.157	0.120

- Let's see why the result is reasonable. From Table1, it's obvious that Movie5 is the most similar one with Movie1, and User1 rates 4 for Movie5 from Table2. Thus our Content-based Method shows that 4 is the most probable rating except 0.

Step 2 Collaborative propagation:

- For each $U_k \in U_1^+$, set $Pr(u_{k,r^\star}) = 1$, i.e. $Pr(u_{2,2}) = 1$, $Pr(u_{3,1}) = 1$
- Propagate to U_1 using Canonical Weighted Sum.

$$Pr(u_{1,r} \mid Pa(U_1)) = \sum_{U_k \in Pa(U_1)} w(u_{k,s}, u_{1,r})$$

Here we use a fixed neighbourhood (selecting the top-N most similar users) as the parents for our active user:User1. Let's see N=1.

Similarity between users are calculated as

$$Sim(U_i, U_j) = abs(PC(U_i, U_j)) \times \frac{\mid M(U_i) \cap M(U_j) \mid}{\mid M(U_i) \mid}$$

where PC denotes Person correlation coefficient and M is the set of movies rated by the user in the Table2. In this way, we are measuring how the sets of ratings overlap.

In our example:

Similarity with User1	1	2	3
User1	1	0.219	0.064

Thus, in our example, the unique parent for User1 is User2.

The canonical weights are calculated as:

$$\begin{aligned} w(u_{i,t}, u_{1,s}) &= Rsim(U_i, U_1) \times Pr(U_1 = s \mid U_i = t) \\ w(u_{i,t}, u_{1,0}) &= 0 \\ w(u_{i,0}, u_{1,0}) &= Rsim(U_i, U_1) \\ w(u_{i,0}, u_{1,s}) &= 0 \end{aligned}$$

where $Rsim(U_i, U_1) = \frac{Sim(U_i, U_1)}{\sum_{j \in Pa(U_1)} Sim(U_j, U_1)}$ and $Pr(U_1 = s \mid U_i = t) = \frac{N(u_{i,t}, u_{1,s}) + 1/5}{N(u_{i,t}) + 1}$ where $N(u_{i,t}, u_{1,s})$ is the number of times from $M(U_i) \cap M(U_1)$ which have been rated t by U_i and also s by the active user:User1.

Applying the process above, we got:

Probability of rate(CF)	0	1	2	3	4	5
User1	0	0.1	0.1	0.6	0.1	0.1

- The probability of rating 0 is 0 because the all the parents for User1, User2, has rated the Movie1.
- The probability of rating 3 is the largest, which is exactly what we are expected because User2 rates 2 for Movie 1 and the only Movie from $M(U_1) \cap M(U_2)$ which has been rated 2 by U_2, Movie2, has been rated 3 by the active user: User1.

Step 3 Combine content-based and collaborative likelihoods at hybrid node U_1:

- We propose a parameter α_1, $0 \leq \alpha_1 \leq 1$, be used to control the contributions of each method.

$Pr(u_{1,s}|u^{CB}_{1,s}, u^{CF}_{1,s}) = 1$

$Pr(u_{1,s}|u^{CB}_{1,s}, u^{CF}_{1,q}) = \alpha_1$ if $q \neq s$

$Pr(u_{1,s}|u^{CB}_{1,t}, u^{CF}_{1,s}) = 1 - \alpha_1$ if $t \neq s$

$Pr(u_{1,s}|u^{CB}_{1,t}, u^{CF}_{1,q}) = 1$ if $t, q \neq s$

Considering this equation, the higher the value of α_1, the greater the weights of the content-based nodes. Our initial hypothesis is that the parameter α_1 might depend on our confidence on the results obtained in the collaborative method. Let's see: $\alpha_1 = Pr(u^{CF}_{1,0})^2$.

In our example:

Probability of rate(Hybrid)	0	1	2	3	4	5
User1	0	0	0	1	0	0

Step 4 Selecting the predicted rating:

- For User1, we transform the posterior probability into a new one in the domain $\{1, 2, ...5\}$ using the following expression:

$$Pr(U_1 = s) = \frac{Pr((U_1 = s)}{1 - Pr(U_1 = 0)}, \forall s \in \{1, 2, ...5\}$$

Thus,

Probability of rate(Hybrid)	1	2	3	4	5
User1	0	0	1	0	0

Finally, we choose the most probable rating which is 3 in this case.

Step 5 Evaluation:

-

$$MAE = \frac{\sum_{i=1}^{N} abs(p_i - r_i)}{N}$$

By applying a loop on items, we got 5 estimate ratings for User 1:

User1	Movie1	Movie2	Movie3	Movie4	Movie5
True ratings	0	3	5	1	4
Estimated ratings	3	3	5	1	1

$$MAE = 0.75$$

$$C = 75\%$$

7 CHTC

- The most advantage of CHTC would be running thousands of jobs at the same time. It is fast and efficient. So we need to break our large computational task into many smaller tasks. So for our algorithm, it's naturally to break the loop into predictions for only one user and treat it be a small job.
- For each job, there are three basic files we need to create. The first file is the R script of a small job. The second file is the executable file with extension .sh that specifies the parameters, R script and the output file name. The last file with extension .sub submits jobs to the condor and tells the executable file, input files that needs to be transferred (R script, installed R with packages, data sets that will be read).
- Parameters can change along with the submission queue or can be fixed as you specify.
- For our movieLens data set, we submit 943 jobs to condor and each job is predicted ratings for one user.

8 Result for the hybrid model

This section of the paper presents some basic performance measurement of the model on the Movie Lens 100k data set. We use two measurements MAE and correct prediction percentage as described before in model evaluation part. The predicting ratings of 943 users for 1682 movies are obtained using the algorithm shown in previous example, which is

Method	MAE	Correct Prediction Percentage
Hybrid model	1.24	36%
Rlab	0.78	44%

Here Rlab is the R package "recommenderlab". It's calculated using user-based collaborative filtering. The predicted rating includes decimal part so the MAE is significantly smaller than the hybrid model. We round up the rating and obtain the accurate prediction is about 44 percent.

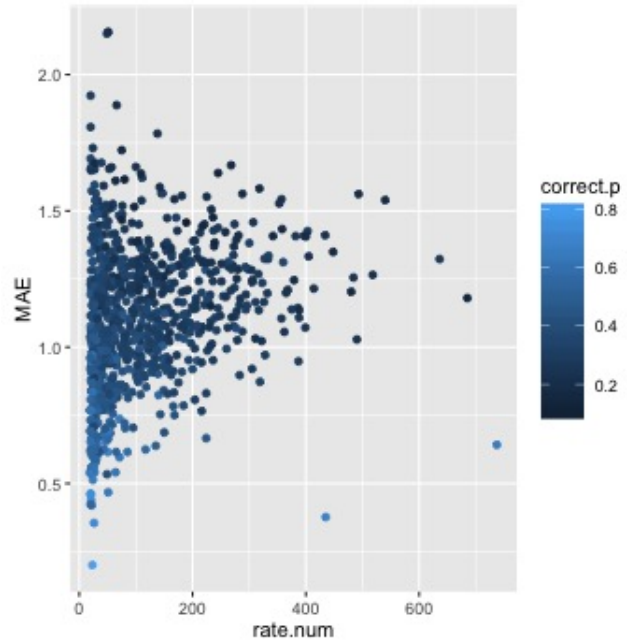

Figure 2: Overall Performance

The x axis of Figure 2 is the number of movies rated by each user. It shows the overall performance of our hybrid approach, from it we can see the system comes stable as the number of rated movies goes larger. The problem of cold-start will exist since the CB part lack of features.

9 Conclusion and Future Work

Here we propose a hybrid Bayesian network model for movie recommendation system. The probabilities are quantified by canonical models, which allows us to design an efficient inference procedure. Content-based and collaborative recommendation is automatically selected during the procedure according to how uncertain we are at the collaborative level.

For further research, we think that the data of movie genre should include more information such as director, leading actor/actress and release date. It helps the contend-based recommendation to do a better job with more information. In this paper there are only 18 genres for almost 2000 movies, which obviously is not sufficient and lower the accuracy of the prediction.

Furthermore, there are also other hybrid methods in recommendation systems. Burke[2] classified them into seven categories, weighted, switching, mixed, feature combination, feature augmentation, cascade, and meta-level. This paper should be considered as feature augmentation. We can try other method during the hybrid step of out algorithm, and hopefully we can obtain a better result. Other than that, the algorithm takes about fifteen minutes to predict one user's rating vector for all 1682 movies. Perhaps we can work on the computation efficiency part in the future.

References

[1] R. Burke. Hybrid recommender systems: survey and experiments. *User Modeling and User-Adapted Iteraction*, 12:331–370, 2002.

[2] R.D. Burke. Hybrid web recommender systems. *Lecture Notes in Computer Science*, 4321:377–408, 2007.

[3] A. Tuzhilin G. Adomavicius. Toward the next generation of recommender systems: a survey of the stat-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, 2005.

[4] P.Caravelas G. Lekakos. A hybrid approach for movie recommendation. *Multimedia Tools and Applications*, 36:55–70, 2008.

[5] L.G. Terveen J.L. Herlocker. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

[6] D. Heckerman J.S Breese. Empirical analysis of predictive algorithms for collaborative filtering. *pp*, pages 43–52, 1998.

[7] Juan M. Luis M. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International journal of Approximate Reasoning*, 51:785–799, 2010.

Appendix

```
# hello-chtc.sub
# My very first HTCondor submit file
#
# Specify the HTCondor Universe (vanilla is the default and is used
#  for almost all jobs), the desired name of the HTCondor log file,
#  and the desired name of the standard error file.
#  Wherever you see $(Cluster), HTCondor will insert the queue number
#  assigned to this set of jobs at the time of submission.
universe = vanilla
log = movie_$(Cluster).log
error = movie_$(Cluster)_$(Process).err
#
# Specify your executable (single binary or a script that runs several
#  commands), arguments, and a files for HTCondor to store standard
#  output (or "screen output").
#  $(Process) will be a integer number for each job, starting with "0"
#  and increasing for the relevant number of jobs.
executable = movie.sh
output = movieout/movie_$(Cluster)_$(Process).out
#
# Specify that HTCondor should transfer files to and from the
#  computer where each job runs. The last of these lines *would* be
#  used if there were any other files needed for the executable to run.
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = movie.R, R.tar.gz, ml-100k.zip
#
# Tell HTCondor what amount of compute resources
#  each job will need on the computer where it runs.
request_cpus = 1
request_memory = 1GB
request_disk = 1GB
#
# Tell HTCondor to run 3 instances of our job:
arguments = $(Process)
queue 943

##First read in the arguments listed at the command line
args=(commandArgs(TRUE))

##args is now a list of character vectors
## First check to see if arguments are passed.
## Then cycle through each element of the list and evaluate the expressions.
if(length(args)==0){
   print("No arguments supplied.")
   ##supply default values
```

```
    a = 1
    b = c(1,1,1)
}else{
    for(i in 1:length(args)){
      eval(parse(text=args[[i]]))
    }
}

library(Hmisc)
  u=row
  N=20
  u.data<-read.delim("./ml-100k/u.data",head=F)
  names(u.data)<-c("userid","itemid","rating","timestamp")
  u.info<-read.delim("./ml-100k/u.info",head=F,sep="")
  u.item<-read.delim("./ml-100k/u.item",sep="|",head=F)
  names(u.item)<-
c("moveid","movietitle","releasedate","videoreleasedate","IMDbURL","unknown","Action","Adven
ture","Animation","Children's","Comedy","Crime","Documentary","Drama","Fantasy","Film-
Noir","Horror","Musical","Mystery","Romance","Sci-Fi","Thriller","War","Western")
  u.genre<-read.delim("./ml-100k/u.genre",sep="|",head=F)
  u.user<-read.delim("./ml-100k/u.user",sep="|",head=F)
  names(u.user)<-c("userid","age","gender","occupation","zipcode")
  item<-dim(u.item)[1]
  user<-dim(u.user)[1]
  feature<-dim(u.genre)[1]
  table1<-as.matrix(u.item)[,6:(5+feature)]
  class(table1)<-"numeric"
  table2<-matrix(,ncol=item,nrow=user)
  for(k in 1:u.info[3,1]){
    i<-u.data$userid[k]
    j<-u.data$itemid[k]
    table2[i,j]<-u.data$rating[k]
  }
  table2[is.na(table2)]<-0
  nk=colSums(table1!= 0)
  item.num=dim(table1)[1]
  feature.num=dim(table1)[2]
  pre_r=rep(0,item.num) #run by here 6:12pm
  for(i in 1:item.num){
    if(table2[u,i]!=0){
    M.vector=table1 %*% (as.matrix(log(item.num/nk+1)))
    table3= t(t(table1) * (log(item.num/nk+1))) * as.vector((1/M.vector))
    p.itemgivenev=rep(0,item.num)
    p.f=1/feature.num
    p.item=sum(table3[i,]*p.f)
    p.featuregivenev=rep(p.f,feature.num)+table3[i,]*p.f*(1-p.f)/p.item
    p.itemgivenev=table3 %*% as.matrix(p.featuregivenev)
    p.itemgivenev[i,1]=1
    p.itemgivenev_n=1-p.itemgivenev
```

```
    pa<-as.numeric(table2[u,]!=0)
    p0<-t(as.matrix(pa)) %*% (p.itemgivenev_n/sum(pa))
    w<-matrix(0,nrow=5,ncol=item.num)
    aa<-cbind(table2[u,],1:item.num)
    w[aa]=1/sum(pa)
    p.cb<-c(p0,w %*% p.itemgivenev)
    pc=rcorr(t(table2), type="pearson")$r
    index<-t(table2[u,]*t(table2)!=0)
    c<-rowSums(index)
    sim<-c/c[u]*(abs(pc)[u,])
    neighbor<-order(sim,decreasing=T)[2:(1+N)]
    rsim<-sim/sum(sim[neighbor])
    N_ua<-matrix(,nrow=N,ncol=6)
    p.a<-matrix(,nrow=N,ncol=6)
    for (n in 1:N){
      j<-neighbor[n]
      if (table2[j,i]!=0){
        for (s in 1:5){
          N_ua[n,s]=sum(table2[j,index[j,]]==table2[j,i] & table2[u,index[j,]]==s)
        }
        N_ua[n,6]=sum(table2[j,index[j,]]==table2[j,i])
        p.a[n,]<-c(0,((N_ua[n,]+1/5)/(N_ua[n,6]+1))[-6]) #p(a=0|u=r)=0
      }
      if (table2[j,i]==0){
        p.a[n,]<-c(1,rep(0,5))
      }
    }
    p.coll<-rsim[neighbor] %*% as.matrix(p.a)
    alpha<-vector()
    alpha<-p.coll[1]^2
    cb<-(order(p.cb,decreasing=T)-1)[1]
    cf<-(order(p.coll,decreasing=T)-1)[1]
    p<-rep(0,6)
    if (cb==cf){
      p[cb+1]=1
    }
    if (cb!=cf){
      p[cb+1]=alpha
      p[cf+1]=1-alpha
    }
    pp=p[-1]/(1-p[1])
    pre_r[i]=order(pp,decreasing=T)[1]
    }
  }
pre_r
name = paste0(u, N, ".csv")
write.csv(pre_r,file = name)

#!/bin/bash
```

```
# untar your R installation
unzip ml-100k.zip
tar -xzvf R.tar.gz

# make sure the script will use your R installation
export PATH=$(pwd)/R/bin:$PATH

# run R, with the R script name as an argument to this bash script
R CMD BATCH '--args row='$1' ' movie.R movie$1_20.Rout
```