FilmAffinity Recommender System Database and its Java Framework

ABSTRACT

The research process in the area regarding the collaborative filtering of recommender systems requires public databases to be available with which to experiment test equations and algorithms and compare results. In this paper, we present a new public database: *FilmAffinity*, which will help to open up the possibilities for researchers to carry out comparative studies with the results of their experiments.

In order to reinforce and facilitate understanding and use of the database we also provide a series of basic statistical results and common measurements of collaborative filtering which are used to compare this database with *MovieLens* and *NetFlix*.

We also publish a Java-based framework which will enable researchers to obtain results on recommender systems by extending their own metrics and user processes with the ease provided by object-oriented programming.

INTRODUCTION

The basic principle of recommender systems (RS) is the expectation that the group of users similar to one given user, (i.e. those that have rated an important number of elements in a similar way to the user) can be used to adequately predict that individual's ratings on products the user has no knowledge of. This way, a trip to Senegal could be recommended to an individual who has rated different destinations in the Caribbean very highly, based on the positive ratings about the holiday destination of "Senegal" of an important number of individuals who also rated destinations in the Caribbean very highly. This suggestion (recommendation) will often provide the user of the service with inspiring information from the collective knowledge of all other users of the service.

In short, the internal operating core of RS is based on carrying out collaborative filtering (CF) [1,2] starting from the ratings expressed by a group of users about a group of items, the aim is to select users who have the most similar ratings or tastes to those of the individual who is using the system at any one time. In general, the objective is to suggest a series of elements to the individual, on which this individual have not shown a preference but which have been very highly rated by an important proportion of the group of users with similar preferences to the individual.

The quality of the results offered by a RS greatly depends on the quality of the results provided by its CF [1] phase; i.e. it is essential to be capable of adequately selecting the group of users most similar to a given individual.

CF memory-based methods [3,4,5] use metrics [2,5] that are directly applied to the data matrix that contains the ratings made by the set of users of the system on the set of items available. The current RS for commercial use employ memory-based methods due to their robustness, predictability and efficiency.

Most of the process for research into the CF of RS requires the use of real databases with a sufficient size in order to test the new equations, methods and algorithms that the scientific community develops, and which could not be validated or perfected without the use of these databases.

An important aspect of scientific production on new methods of CF lies in the fact that the databases used are public with the aim of making them accessible to all researchers, in order to develop new recommendation algorithms and to be able to validate and extend those that emerge.

There are currently a very small number of databases used regularly by researchers. Table 1 shows a summary of their essential characteristics.

	items	users	ratings
MovieLens	10681	71567	10000000
Book-Crossing	271379	278858	1149780
NetFlix	17770	480189	100480507
Jester	100	19986	1810455

Table 1. Most widely used RS databases

MovieLens [6] is historically the most widely used database in research; it comes from EachMovie. As of October, 2004, HP retired the EachMovie dataset, it is no longer available for download. MovieLens is also available in the following sizes: 100,000 ratings for 1682 movies by 943 users, 1 million ratings for 3900 movies by 6040 users.

Book-Crossing [7] was collected by Cai-Nicolas Ziegler (2004) from the Book-Crossing community.

NetFlix [8] is by far the largest database, which is an asset as regards validating research results, although its large size means it requires excessive calculation times in the process of developing and fine-tuning new CF algorithms.

The Jester online joke recommender system [9] presents the disadvantage of relying on too small a number of items, but it is useful when working with data matrices that are not very sparse.

In addition to the availability of public databases, which is essential, it is convenient for frameworks to exist on which the scientific community can base its research on the CF of RS without the need for each person or research group to develop their own.

Representative frameworks in the area of RS:

- COFE (COllaborative Filtering Engine) [10], a free Java recommendation engine for collaborative filtering. This free server (including source code) allows anyone to easily set up a recommendation system.
- Duine [11] is a collection of software libraries that allows developers to create prediction engines for their own application. Duine has been developed by the Telematica Instituut/Novay.
- Small C++ framework [12] by Benjamin Meyer, created to be applied on NetFlix.

In the following sections of the paper we present the most important characteristics of the FilmAffinity public database and its associated framework, which we hope can be used to reinforce those that already exist and, in any case, to facilitate the experiments in which comparative results are displayed using the same algorithm on different databases.

THE FILMAFFINITY PUBLIC DATABASE

FilmAffinity.com [13] is an on-line film recommendation company that currently has around 120,000 users and 30 million votes. Its CF core has been developed at the UPM [14] by the AICU group [15], where the database and the associated Java framework can be downloaded.

Due to the company's requirements, the database we provide for research currently contains 5 million votes, and no user information is provided. Even so, its size and technical characteristics are suitable for the research process in RS.

Its most relevant characteristics are summarized in Table 2.

Number of users	9257
Number of films	19970
Number of ratings	5333988
Min. and max. values of the ratings	1..10
Mean of the ratings	6.65
Typical deviation of the ratings	0.78
Minimum number of ratings (all the users)	51
Mean of the users ratings	576

Table 2. Most representative values of the FilmAffinity database for public use

The distribution of the votes is provided in the graph in Figure 1.

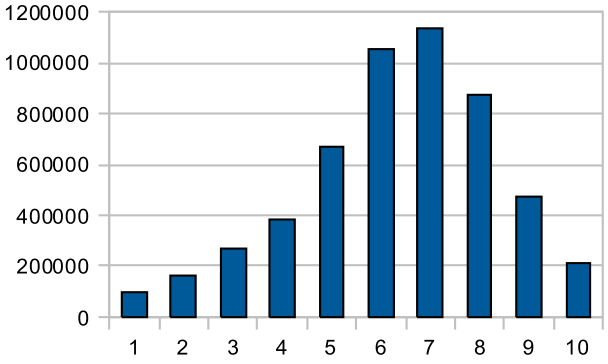


Figure 1. Number of existing votes in each of their possible values [1..10]

Figure 2 shows a comparative of the values of accuracy obtained with FilmAffinity using Pearson correlation compared to other film databases: MovieLens and NetFlix. It is very interesting to observe that FilmAffinity offers the best results of accuracy (fewer MAE errors) [1,2,5], and therefore, it is particularly interesting in order to test equations, methods and algorithms that aim to improve this highly important parameter of CF in RS.

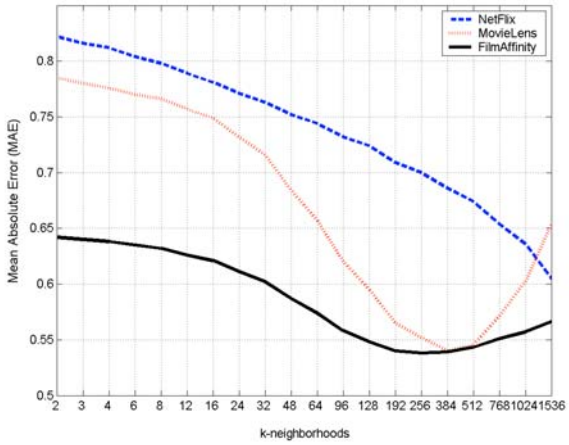


Figure 2. Results of accuracy obtained for different values of k-neighborhoods (x axis) in different RS databases. In the case of FilmAffinity, its ratings have been previously normalized in the range [1..5]

Figure 3 shows us the estimation capability achieved using Pearson correlation in NetFlix, MovieLens and FilmAffinity. The estimation capability refers to the percentage of predictions that the k-neighborhoods can make.

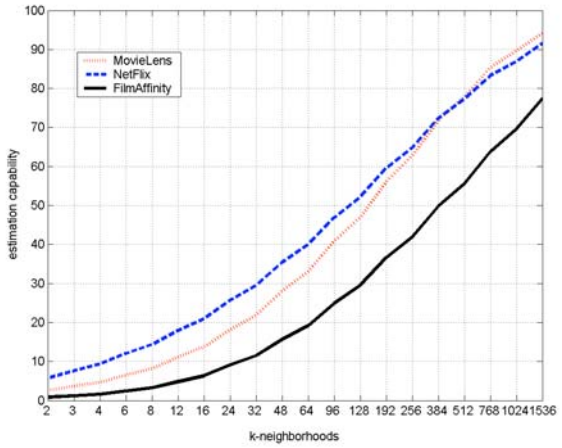


Figure 3. Percentage of estimation capability of FilmAffinity, MovieLens and NetFlix obtained for different values of k-neighborhoods (x axis)

In figure 3 we can observe that when the number *k* of neighbors is small the estimation capacity is low (it is more improbable that one of the few k-neighborhoods has voted for a given film) and therefore as we take larger values for *k* the estimation capacity grows as it becomes more probable that one of the many neighbors has voted for a given film.

The lower prediction capability of FilmAffinity is due to logical behavior related with the fact that using this database is possible to find more similar neighbors (which improve the measure of accuracy); the more similar the neighbors are to the test user, in general, not only will they have more similar vote values, but also they will have a greater tendency to vote for the same subset of the total films rated (the same genres, in the same years, etc.).

JAVA FRAMEWORK FOR RECOMMENDER SYSTEMS

Besides the database, in [15] a framework programmed in Java is provided which enables results to be obtained using the FilmAffinity database or any other equivalent RS. Indeed, FilmAffinity is provided in text mode, with the same format as MovieLens.

This framework makes intensive use of the object-oriented characteristics of Java, enabling the developers to extend interfaces to create new metrics and their own user processes. Additionally, it has the advantage of single-handedly recognizing the characteristics of the database and of the physical system (hardware) with the aim of adapting itself to the most effective processing mode, by automatically balancing the use of memory and disk space and deciding on the ideal number of threads to minimize the process time.

Figure 4 shows a general view of the system architecture, detailing the dependencies between its most representative objects.

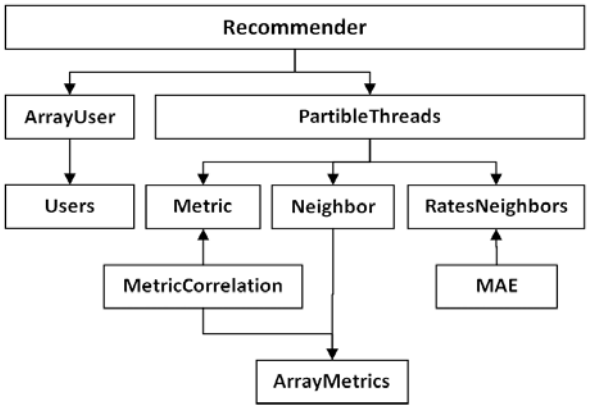
Figure 4. RS Java framework software architecture

Table 3 provides a brief description of the utility of each of the most representative objects of the framework.

Java Object	Description
User	Contains each user data (ratings, etc.)
ArrayUsers	Reads from file and stores the users
ArrayMetrics	Compute and store the similarities results for each couple of users
Recommender	The main class.
PartibleThreads	Automatically manages all the threads.
Partible	Java interface to implement user processes
Metric	Abstract class to implement your own metrics.
MetricCorrelation	Implements Pearson correlation
Neighbors	Obtains the users k-neighborhoods (from ArrayMetrics)
MAE	Computes the MAE

Table 3. Description of the fundamental objects of the framework.

Finally, as an example, we include a program which, by using the framework, shows the way to obtain the MAE of



the FilmAffinity database using 20% (0.2) as test users, the Pearson correlation metric and the k-neighborhoods values 2,3,4,6, etc.

```
import recommender.*;
public class Demo {
public static void main(String[] args) {
ArrayUsers arrayUsers; Recommender recommender;

arrayUsers=new ArrayUsers("affinity.txt",0.2);

System.out.println("Number of users: " +
arrayUsers.getNumberOfUsers());//...

recommender = new Recommender(arrayUsers);
recommender.process(new MetricCorrelation());
```

```
int ks[] = {2,3,4,6,8,12,16,24,32,48,64,96};
int kMax = ks[ks.length - 1];

recommender.process(new Neighbors(kMax));

for (int i = 0; i < ks.length; i++) {
        recommender.process(MAE(ks[i]));
        System.out.println("MAE:"+(Double)
        arrayUsers.get("MAE"));
}}}
```

CONCLUSIONS

The availability of public databases is essential for the research process in the area of recommender systems, whilst free access to programming frameworks facilitates and accelerates the research and development of collaborative filtering cores. The scientific community now has a new database (FilmAffinity), as well as its associated programming framework.

The most outstanding characteristic of the FilmAffinity database is that it is a real commercial database that presents a lower level of accuracy than its better known references (MovieLens and NetFlix), making it ideal for a large number of the objectives of collaborative filtering research.

The framework provided allows metrics and user processes to be extended with the simplicity that characterizes object-oriented programming.

ACKNOWLEDGMENTS

Our acknowledgement to the GroupLens Research Group, and to the FilmAffinity and NetFlix companies.

REFERENCES

1.Adomavicius, Tuzhilin, A. Toward the Next Generation of Recommender Systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Enginnering, vol. 17, no. 6, (June 2005), 734-749

2.Herlocker, J. L., Konstan, J.A., Riedl, J.T., Terveen, L.G. Evaluating Collaborative Filtering Recommender Systems, ACM Transactions on Information Systems, vol. 22, no. 1, (January 2004), 5-53

3.Breese, J.S., Heckerman, D., Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering, in Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1998, 43-52

4.Kong, F., Sun, X., Ye, S. A Comparison of Several Algorithms for Collaborative Filtering in Startup Stage, In Proceedings of the IEEE networking, sensing and control, (March 2005), 25-28

5.Sanchez, J.L., Serradilla F., Martinez E. & Bobadilla, J. Choice of Metrics used in Collaborative Filtering and their Impact on Recommender Systems, in Proceedings of the IEEE International Conference on Digital Ecosystems and Technologies DEST, (February 2008) , 432 – 436, Digital Object Identifier 10.1109/DEST. 2008.4635147.

6.http://www.movielens.org

7.http://www.informatik.uni-freiburg.de/~cziegler/BX/

8.www.netflixprize.com

9.http://www.ieor.berkeley.edu/~goldberg/jester-data/

10.http://eecs.oregonstate.edu/iis/CoFE/

11.http://duineframework.org/

12.http://github.com/icefox/ netflixrecommenderframework/tree/master

13.http://www.filmaffinity.com

14.http://www.upm.es

15.http://aicu.eui.upm.es/aicu/doku.php?id=english