Knowledge-Based Systems 24 (2011) 1310–1316

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Improving collaborative filtering recommender system results and performance using genetic algorithms

Jesus Bobadilla [*,1], Fernando Ortega, Antonio Hernando, Javier Alcalá

Universidad Politécnica de Madrid, Computer Science, Crta. De Valencia, Km 7, 28031 Madrid, Spain

ARTICLE INFO

Article history:
Received 18 October 2010
Received in revised form 30 March 2011
Accepted 7 June 2011
Available online 15 June 2011

Keywords:
Collaborative filtering
Recommender systems
Similarity measures
Metrics
Genetic algorithms
Performance

ABSTRACT

This paper presents a metric to measure similarity between users, which is applicable in collaborative filtering processes carried out in recommender systems. The proposed metric is formulated via a simple linear combination of values and weights. Values are calculated for each pair of users between which the similarity is obtained, whilst weights are only calculated once, making use of a prior stage in which a genetic algorithm extracts weightings from the recommender system which depend on the specific nature of the data from each recommender system. The results obtained present significant improvements in prediction quality, recommendation quality and performance.

1. 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 recent years, RS have played an important role in reducing the negative impact of information overload on those websites where users have the possibility of voting for their preferences on a series of articles or services. Movie recommendation websites are probably the most well-known cases to users and are without a doubt the most well studied by researchers [19,4,22], although there are many other fields in which RS have great and increasing importance, such as e-commerce [15], e-learning [9,5] and digital libraries [26,27].

Currently, the fast increase of Web 2.0 [18,23] has led to the proliferation of collaborative websites in which the number of elements that can be recommended (e.g. blogs) can increase significantly when introduced (and not only voted) by the users, which generates new challenges for researchers in the field of RS, at the same time as it increases the possibilities and importance of these information retrieval techniques.

The core of a RS is its filtering algorithms: demographic filtering [20] and content-based filtering [21] are the most basic techniques; the first is established on the assumption that individuals with certain common personal attributes (sex, age, country, etc.) will also have common preferences, whilst content-based filtering recommends items similar to the ones the user preferred in the past. Currently, collaborative filtering (CF) is the most commonly used and studied technique [12,24], it is based on the principle set out in the first paragraph of this section, in which in order to make a recommendation to a given user, it first searches for the users of the system who have voted in the most similar way to this user, to later make the recommendations by taking the items (holiday destinations in our running example) most highly valued by the majority of their similar users.

The most significant part of CF algorithms refers to the group of metrics used to determine the similitude between each pair of users [14,1,7], among which the Pearson correlation metric stands out as a reference.

Genetic algorithms (GA) have mainly been used in two aspects of RS: clustering [16,17,28] and hybrid user models [10,13,2]. A common technique to improve the features of RS consists of

* Corresponding author. Tel.: +34 913365133; fax: +34 913367522.
E-mail addresses: jesus.bobadilla@upm.es (J. Bobadilla), fortegarequena@gmail.com (F. Ortega), ahernado@eui.upm.es (A. Hernando), jalcala@eui.upm.es (J. Alcalá).
[1] Universidad Politécnica de Madrid & FilmAffinity.com research team.

initially carrying out a clustering on all of the users, in such a way that a group of classes of similar users is obtained, after this, the desired CF techniques can be applied to each of the clusters, obtaining similar results but in much shorter calculation times; these cases use common genetic clustering algorithms such as GA-based *K*-means [17].

The RS hybrid user models commonly use a combination of CF with demographic filtering or CF with content based filtering, to exploit merits of each one of these techniques. In these cases, the chromosome structure can easily contain the demographic characteristics and/or those related to content-based filtering.

The method proposed in the article uses GA, but with the advantage that it does not require the additional information provided by the hybrid user models, and therefore, it can be used in all of the current RS, simply based on CF techniques. This is due to the fact that our method only uses the rating of users (which is the least possible information in any RS).

The following section defines the proposed method (GA-method) and the prior processing required using GA, after this, we present the sections which specify the design of experiments carried out and the results obtained, finally we list the most relevant conclusions from this study.

2. Design of the new similarity method

The fundamental objective is to improve the results of CF by obtaining a metric that improves the accuracy [12,7,11,8] of CF based RS. For this purpose, after a series of experiments carried out on different metrics, different levels of sparsity [25,6] and different databases, we have obtained an original, simple and efficient approach which improves the usual results obtained in RS.

2.1. Values

We will consider a RS with a set of U users, $\{1,\ldots,U\}$, and a set of I items $\{1,\ldots,I\}$. Users rate those items they know with a discrete range of possible values $\{m,\ldots,M\}$ where m usually represents that the user keeps completely unsatisfied about an item, and a value M usually represents that the user is completely satisfied with an item. RS normally use m with value 1 and M with value 5 or 10. In this way, the range of possible ratings is usually $\{1,\ldots,5\}$ or $\{1,\ldots,10\}$.

The ratings made by a particular user x can be represented by a vector, $\boldsymbol{r}_x = \left(r_x^{(1)}, r_x^{(2)} \ldots, r_x^{(I)}\right)$ with dimension I (the number of items in the RS) in such way that r_x^i represents the rating that the user x has made over the item i. Obviously, a user may not rate all the items in the RS. We will use the symbol • to represent that a user has not rated an item. Consequently, we use the expression $r_x^{(i)} = \bullet$ to state that the user x has not rated the item i yet.

Next, we will consider an example. We will consider that $m = 1$ and $M = 5$ (that is to say the possible set of ratings is $\{1,\ldots,5\}$), there are nine items, $I = 9$, and there are two users whose ratings are represented by the following $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ vectors:

$$\boldsymbol{r}_1 = (4,5,\bullet,3,2,\bullet,1,1,4)$$
$$\boldsymbol{r}_2 = (4,3,1,2,\bullet,3,4,\bullet,2)$$

As may be seen, while the user 1 has rated the item 5, $r_1^5 = 2$, the user 2 has not rated the item 5 yet: $r_2^5 = \bullet$.

In order to compare both vectors, $\boldsymbol{r}_x$, $\boldsymbol{r}_y$, we can consider another vector $\boldsymbol{v}_{x,y} = \left(v_{x,y}^{(0)},\ldots,v_{x,y}^{(M-m)}\right)$ whose dimension is the number of the possible ratings that a user can make over an item. Each component $\boldsymbol{v}_{x,y}^{(i)}$ of the vector $\boldsymbol{v}_{x,y}$, represents the ratio of items, j, rated by both users (that is to say, $r_x^{(j)} \neq \bullet$ and $r_y^{(j)} \neq \bullet$) and over which the absolute difference between the ratings of both users is $i\left(\left|r_x^j - r_y^j\right| = i\right)$, to the number of items rated by both users. That is to say, $\boldsymbol{v}_{x,y}^{(i)} = a/b$ where b is the number of items rated by both users, and a is the number of items rated by both users over which the absolute difference in the ratings of both users is i.

In this way, the vector $\boldsymbol{v}_{1,2}$ for the previous example is the following (just observe that there are only 5 items rated by both users 1 and 2):

$$\boldsymbol{v}_{1,2} = (1/5, 1/5, 2/5, 1/5, 0)$$

All components of the vector $\boldsymbol{v}_{1,2}$ are divided into 5 since there are 5 items rated by both users. The component $v_{x,y}^{(0)}$ represents the ratio of items which both users rated identically to the number of items rated by both users. In the previous example, $v_{1,2}^{(0)} = 1/5$, since there is only one item (the item 1) which both users have rated identically (that is to say, $\left|r_1^{(1)} - r_2^{(1)}\right| = 0$). In the same way, $v_{x,y}^{(M-m)}$ represents the ratio of items rated in opposite way by both users, to the number of items rated by both users. In the previous example, $v_{1,2}^{(M-m)} = v_{1,2}^4 = 0/5 = 0$, since there are no items rated with values 5 and 1 (the highest difference in ratings) by the users 1 and 2. In the example above, $v_{x,y}^{(2)} = 2/5$, since there are exactly two items, item 2 and item 9, such that $\left|r_1^{(2)} - r_2^{(2)}\right| = \left|r_1^{(9)} - r_2^{(9)}\right| = 2$.

2.2. Similarity

We will consider a family of similarity functions. For each vector $\boldsymbol{w} = (w^{(0)},\ldots,w^{(M-m)})$ whose components lie in the range $[-1, 1]$ (that is to say, $w^{(i)} \in [-1,1]$), we will consider the following similarity function:

$$sim_w(x,y) = \frac{1}{M-m+1}\sum_{i=0}^{M-m} w^{(i)}\, v_{x,y}^{(i)} \qquad (1)$$

Consequently, for each vector, $\boldsymbol{w}$, there is a similarity function whose component, $w^{(i)}$, represents the importance of the component $v_{x,y}^{(i)}$ for calculating the similarity between two users (according to this similarity function). In this way, the similarity function associated to the vector $\boldsymbol{w} = (1, 0.5, 0, -0.5, -1)$ represents a similarity function such that:

- Since $w^{(0)} = 1$, this similarity function evaluates highly positively the number of items rated identically.
- Since $w^{(4)} = -1$, this similarity function evaluates highly negatively the number of items which have been rated in opposite way ($w_x^{(0)} = -1$).
- Since $w^{(2)} = 0$, this similarity function takes no consideration of those items over which the difference between the ratings made by both users is 2.
- Since $w^{(1)} = 0.5$, this similarity function evaluates with a positive intermediate value those items over which the difference between the ratings made by both users is 1.
- Since $w^{(3)} = -0.5$, this similarity function evaluates with a negative intermediate value those items over which the difference between the ratings made by both users is 3.

The question related to obtain the most suitable similarity function represented by a vector $\boldsymbol{w}$ for a recommender system depends on the nature of the data in this recommender system. We will try to find, among all the possible vectors $\boldsymbol{w}$, one representing a similarity function which provides a minimal Mean Absolute Error (MAE) in the recommender system. In order to obtain this similarity function, we will use a genetic algorithm.

3. Genetic algorithm

In order to find an optimal similarity function, $sim_{\boldsymbol{w}}$, we use a genetic algorithm to find the vector $\boldsymbol{w}$ associated to the optimal similarity function $sim_{\boldsymbol{w}}$.

We use a supervised learning task (Goldberg, 1989) whose fitness function is the Mean Absolute Error MAE of the RS. In this way, the population of our genetic algorithm is the set of different vectors of weights, $\boldsymbol{w}$. Each individual, that is to say each vector of weights, represents a possible similarity measure (see (1)), for which we will evaluate the MAE of the RS using this similarity measure. While running our genetic algorithm, the successive population generations tend to improve the MAE in the RS. Our genetic algorithm stops generating populations when MAE in the RS for a vector of weight is lower than a threshold, γ (which has been previously established).

As usual, we use only a part of the whole recommender system (training users and training items) in order to obtain an optimal similarity function. Once obtained this similarity function, we carry out our experiments to evaluate the similarity function obtained by using only the test users and the test items (that is to say, those users and items which have not been used in the GA algorithm).

3.1. Genetic representation

As usual, individuals of populations are represented in binary form as strings of 0s and 1s. As we have previously stated, in our genetic algorithm, each vector of weights, $\boldsymbol{w}$, representing a similarity vector is a possible individual of the population. Each component $w^{(i)}$ in the vector of $\boldsymbol{w}$ will be represented by 10 bits. Consequently, the vector $\boldsymbol{w} = (w^{(0)}, \ldots, w^{(M-m)})$ will be represented by the following string of 0s and 1s (with a length of $2^{10(M-m+1)}$ bits):

$$b^9_{M-m} \ldots b^1_{M-m} b^0_{M-m} \quad b^9_{M-m-1} \ldots b^1_{M-m-1} b^0_{M-m-1} \ldots b^9_1 \ldots b^1_1 b^0_1$$
$$b^9_0 \ldots b^1_0 b_1 b^0_1$$

where each component of the vector $w_i \in [-1, 1]$ can be obtained through the following expression:

$$w_i = \frac{2\sum_{j=10}^{9} 2^j b_i^j}{2^{10} - 1} - 1 \tag{2}$$

3.2. Initial population

In order to choose the population size, we have considered the criterion of using a number of individuals in the population which is the double of the number of bits used to represent each individual [3]. Consequently, when using Movielens and Netflix the initial population will be of 100 individuals, in view that in both recommender systems $M = 5$ and $m = 1$, but when using FilmAffinity (where $M = 10$ and $m = 1$) the initial population would be of 200 individuals. Table 1 shows the precise information used in our experiments.

In order to generate the initial population we have considered the following criteria:

- The 50% of the initial population is obtained using random values.

Table 1
Descriptive information of the databases used in the experiments.

	Movielens	Film affinity	Netflix
#Users	6040	26,447	480,189
#Movies	3706	21,128	17,770
#Ratings	1,000,209	19,126,278	100,480,507
Min and max values	1–5	1–10	1–5

- The 50% of the initial population is seeded in areas where optimal vectors $\boldsymbol{w}$ may lie.
  In this way, the value w_0 measures how the similarity function weights the number of items rated identically by two users. Consequently we can reasonably expect that an optimal similarity function will have a high positive value of $w^{(0)}$. In this way, the value $w^{(0)}$ of individuals will be chosen randomly to lie within a high positive range of values. In the same way, the value $w^{(M-m)}$ ($w^{(4)}$ for Movielens and $w^{(9)}$ for FilmAffinity) measures how the similarity function weights the number of items rated in a totally opposite way by two users. Consequently we can reasonably expect that an optimal similarity function will have a high negative value of $w^{(M-m)}$. In this way, the value $w^{(M-m)}$ of individuals will be chosen randomly to lie within a high negative range of values.
  Next, we will show the ranges used to seed the 50% of the initial population.
  With the Movielens and Netflix Recommender System ($m = 1$ and $M = 5$):

$$w_0 \in [1, 0.6], \quad w_1 \in [0.6, 0.2], \quad w_2 \in [0.2, -0.2],$$
$$w_3 \in [-0.2, -0.6], \quad w_4 \in [-0.6, -1]$$

With the FilmAffinity Recommender System ($m = 1$ and $M = 10$):

$$w_0 \in [1, 0.8], \quad w_1 \in [0.8, 0.6] \quad w_2 \in [0.6, 0.4],$$
$$w_3 \in [0.4, 0.2], \quad w_4 \in [0.2, 0], w_5 \in [0, -0.2],$$
$$w_6 \in [-0.2, -0.4], \quad w_7 \in [-0.4, -0.6],$$
$$w_8 \in [-0.6, -0.8], \quad w_9 \in [-0.8, -1]$$

3.3. Fitness function

The fitness function of a genetic algorithm is used to prescribe the optimality of a similarity function $sim_{\boldsymbol{w}}$ (for the individual $\boldsymbol{w}$). In our genetic algorithm, the fitness function will be the Mean Absolute Error (MAE) of the RS (indeed we only use the training users and the training items since we are trying to find an optimal similarity function) for a particular similarity function $sim_{\boldsymbol{w}}$ (described in Eq. (1)).

The MAE is obtained by comparing the real ratings with the predicted ratings made according to the similarity function. In order to calculate the MAE of the RS for a particular similarity function, $sim_{\boldsymbol{w}}$, we need to follow the next steps for each user x:

1. Obtaining the set of *K neighbors of* x, K_x (the K most similar users to a given user x), through the similarity function $sim_{\boldsymbol{w}}$ defined in Eq. (1).
2. Calculating the prediction of the user x and an item i, p_x^i. This is obtained through the Deviation From Mean (DFM) as aggregation approach:

$$P_x^i = \bar{r}_x + \frac{\sum_{n \in K_x} [sim_w(x, n) \times (r_n^i - \bar{r}_n)]}{\sum_{n \in K_x} sim_w(x, n)} \tag{3}$$

where $\bar{r}_x$ is the average of ratings made by the user x.

Once every possible prediction according to the similarity function $sim_{\boldsymbol{w}}$ is calculated, we obtain the MAE of the RS as follows:

$$fitness = MAE = \frac{1}{\#U} \sum_{u \in U} \frac{\sum_{i \in I_u} |p_u^i - r_u^i|}{\#I_u} \tag{4}$$

When running the genetic algorithm, $\#U$ and $\#I_u$ represent respectively the number of training users and the number of training items rated by the user u.

3.4. Genetic operators

As usual, our genetic algorithm uses the common operators in genetic algorithms: selection, crossover (recombination) and mutation. Since we have obtained quick satisfactory results using these three classical operators, we have not used other possible operators like migration, regrouping or colonization-extinction. The features of our genetic operators are:

- Selection. The chosen method is the fitness proportional selection. That is to say, the selection probability of an individual depends on its fitness level.
- Crossover. We use the one-point crossover technique. The crossover probability is 0.8.
- Mutation. We use a single point mutation technique in order to introduce diversity. The mutation probability is 0.02.

3.5. Reproduction and termination

When using the reproduction operator, 1000 individuals are generated in each generation using random crossover between the best-fit parent individuals.

The number of individuals keeps constant through every generation. As we have stated above, the population is 100 when using Movielens and Netflix as recommender systems and 200 when using FilmAffinity as RS. We only keep the 5% of the best individuals from each generation to obtain the next one (elitist selection).

The genetic algorithm stops when there is an individual in the population with a fitness value lower than a constant γ. We use $\gamma = 0.78$ for the Movielens and Netflix RS (in these databases, we have that $m = 1$ and $M = 5$) and we use $\gamma = 1.15$ for the RS FilmAffinity (in this database, we have that $m = 1$ and $M = 10$). We have used these values according to the optimal values found in [7].

4. Experiments

4.1. Prediction and recommendation quality measures

In this section, we show different experiments made in order to prove that the method proposed works for any different RS. In this way, we have used in the experiments the following three different databases (Table 1) of real RS:

- Movielens: This database is a reference in research or RS over the last years. In this database, we can compare the results obtained using our method with the results previously obtained using other methods.
- FilmAffinity: This database contains a large amount of data. This database provides an experiment to prove that our GA method is useful for databases with a large amount of data.
- Netflix: This database offers us a very large database on which metrics, algorithms, programming and systems are put to the test.

Since our method provides high values in the quality measures applied on the three databases, we reduce the danger that the proposed metric may only work for a specific RS. As usual, we will use the typical quality measures: Mean Absolute Error (MAE), coverage, precision and recall.

The results of our proposed genetic algorithm are compared with the ones obtained using traditional metrics on RS collaborative filtering: Pearson correlation, cosine and Mean Squared Differences. These baseline metrics are used to compare our results obtained from the genetic algorithm with the ones obtained from these metrics.

The genetic algorithm described previously is run 20 times. Each time the genetic algorithm runs, it provides a similarity function $sim_{\boldsymbol{w}}$ associated to the vector $\boldsymbol{w}$ (see Equation (1)). Associated to this similarity function, $sim_{\boldsymbol{w}}$, we calculate the typical quality measures: MAE, coverage, precision and recall. For each quality measure we show graphics depicting the best (labeled 'BestGA') and worse value (labeled 'WorstGA') obtained through the 20 runnings. The area between them is displayed in grey color. Each graphic is described with the results obtained with the metrics: Pearson correlation (COR), cosine (COS) and Mean Squared Differences (MSD).

As we have stated above, we only use a part of the whole RS (training users and training items) in order to obtain an optimal similarity function. Once obtained this similarity function, we carry out our experiments using only the test users (the users which are not used in the GA algorithm) and the test items (the items which are not used in the GA algorithm).

When using Movielens and FilmAffinity, we use 20% of users (randomly chosen) as test users and 20% of items (randomly chosen) as test items. We use the rest 80% of users and items as training users and training items. When using Netflix, we also use 20% of items as test items but we only use 5% of the users as test users due to the huge number of users in the database.

The constant K related to the number of neighbors for each user, varies between 50 and 800. These values enable us to view the trends on the graphics. The precision and recall recommendation quality results have been calculated using different values of N (number of recommendations made) from 2 to 20. When using Movielens and Netflix the relevant threshold value $= 5$, and $= 9$ when using FilmAffinity (as stated above, while the possible value in the ranking of item is $\{1, \ldots, 5\}$ in Netflix and Movielens, it is $\{1, \ldots, 10\}$ in FilmAffinity). Table 2 shows the numerical parameters used in the experiments.

4.2. Performance experiment

In this section, we describe an experiment which proves that predictions and recommendations can be made much faster when using the similarity function obtained with our GA-method (see Eq. (1)) than when using the traditional metrics (Pearson Correlation, cosine, Mean Squared Differences). Indeed, since the similarity functions in our GA method (see Eq. (1)) are much simpler than the formulae involved in the traditionalmetrics, we can obtain the k-neighbors for a user much faster (in this way, predictions and recommendations can be obtained much faster).

The experiment designed compares the performance of different similarity measures by calculating the similarity between users

Table 2
Main parameters used in the experiments.

	K (MAE, coverage)		Precision/recall			Test users (%)	Test items (%)	Genetic runs
	Range	Step	K	N	θ			
Movielens 1 M	{50, ..., 800}	50	100	{2, ..., 20}	5	20	20	20
Film affinity	{50, ..., 800}	50	150	{2, ..., 20}	5	20	20	20
Netflix	{50, ..., 800}	50	200	{2, ..., 20}	9	5	20	20

in Movielens. We compare the following similarity measures: Pearson correlation (COR), cosine (COS), Mean Squared Differences (MSD) and the similarity measure obtained with our GA-method.

In this experiment, we have first divided the users in Movielens into two subsets: a subset of 1204 (20% of users which are regarded as test users in Section 4.1); and the subset of the rest of users (80% users, 4832 users). Once divided the database, we have calculated the similarity measure between each user in the first subset of 20% users and each user in the second subset of 80% users.

5. Results

5.1. Prediction and recommendation results

In this section we show the results obtained using the databases specified in Table 1. Figs. 1–3 show respectively the results obtained with Movielens, FilmAffinity and Netflix. As may be seen in these figures, the results obtained for all the quality measures (MAE, coverage, precision and recall) with our genetic algorithm are better that the ones obtained with the traditional metrics.

Next, we will analyze the results according to the three databases:

- **Movielens**: Graphic 1a informs about the MAE error obtained for Movielens when applying Pearson correlation (COR), cosine (COS), Mean Squared Differences (MSD) and the 20 runs of the GA-method. The GA-metric leads to fewer errors, particularly in the most used values of K. The black dashed and continuous lines represent respectively the worst and the best result obtained in the 20 times the GA method is run. The grey area contains the area between these lines.
  Graphic 1b informs about the coverage obtained. As may be seen, our GA-method can improve the coverage for any value of K (the number of neighbors for each user) in relation to any traditional metric used. This fact must be emphasized since the new latter similarity metrics which are usually proposed improve the MAE while resulting in a worse coverage [7]. While Graphic 1a shows that the best results in MAE with the GA-method are obtained when using a small value in K, Graphic 1b shows that the best results with the GA-method are obtained in coverage using medium values in K. In this way, we should use intermediate values in K ($K \in \{150, \ldots, 200\}$) for obtaining the most satisfactory results both in MAE and in coverage.
  Graphics 1c and 1d inform respectively about the precision and recall. These quality measures are improved for any value used in the number of recommendations, N. Consequently, the GA-method improves not only the accuracy and the coverage, but also provides better recommendations.
- **FilmAffinity**: The results obtained with this database provide similar results to the one obtained for the database Movielens. As illustrated in Graphic 2a, we can see that our GA-method does not provide so good results in MAE with low K for this database as for MovieLens. However, it provides better results when using high values in K. The reader must observe that the ranking values used in FilmAffinity are larger ($\{1, \ldots, 10\}$) than the ones used in Movielens.
  As may be seen in Graphic 2b, our GA-method provides best levels in coverage. In the same way, the results obtained for precision and recall (see Graphics 2c and 2d) in this database are similar to the ones obtained for the MovieLens database.
- **Netflix**: The results obtained with this database may be used to state that our method can be used for a RS with a large amount of data (see Table 1). Our method provides similar results for this database to the ones obtained for previous databases in both quality prediction measures (MAE and coverage) and quality recommendation measures (precision and recall). Nevertheless, we must emphasize that regarding both MAE (graphic 3a) and precision (graphic 3c) our GA method shows improvements when used in connection to this database.

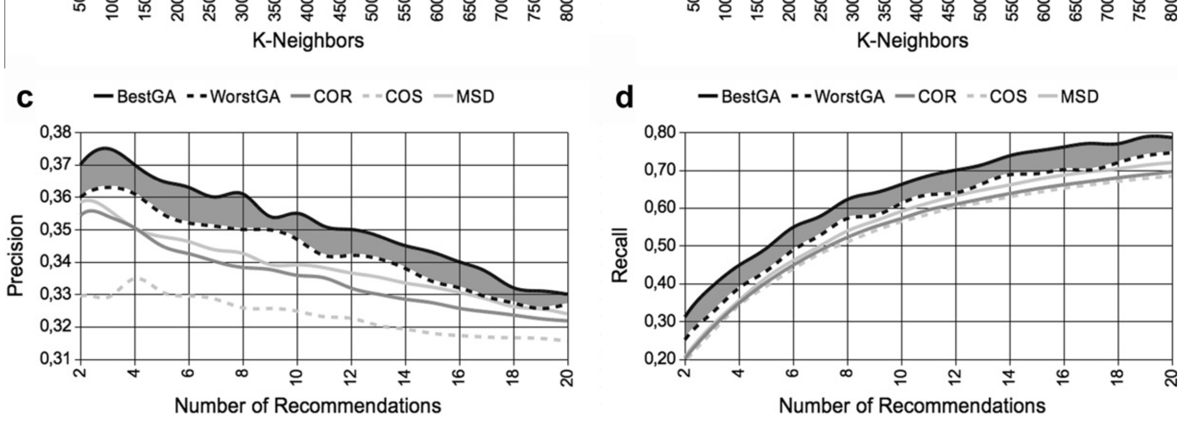

Fig. 1. Traditional metrics and proposed genetic similarity method comparative results using Movielens: (a) accuracy (Mean Absolute Error), (b) coverage, (c) precision, (d) recall.

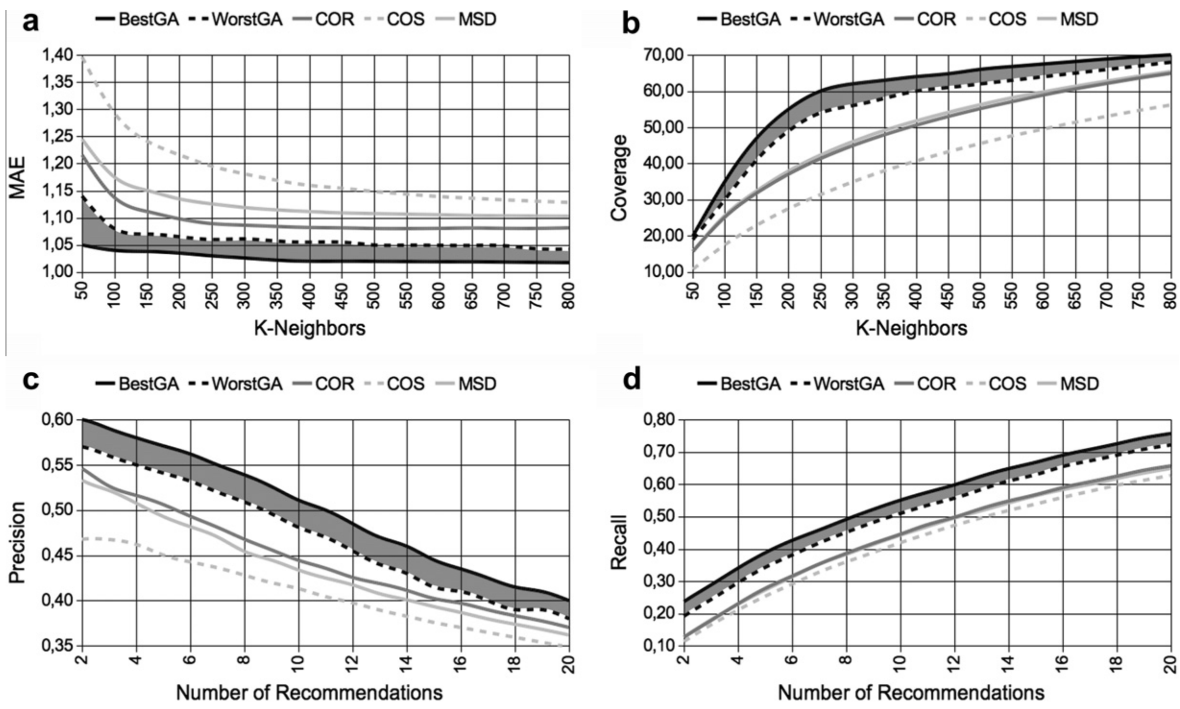

Fig. 2. Traditional metrics and proposed genetic similarity method comparative results using FilmAffinity: (a) accuracy (Mean Absolute Error), (b) coverage, (c) precision, (d) recall.

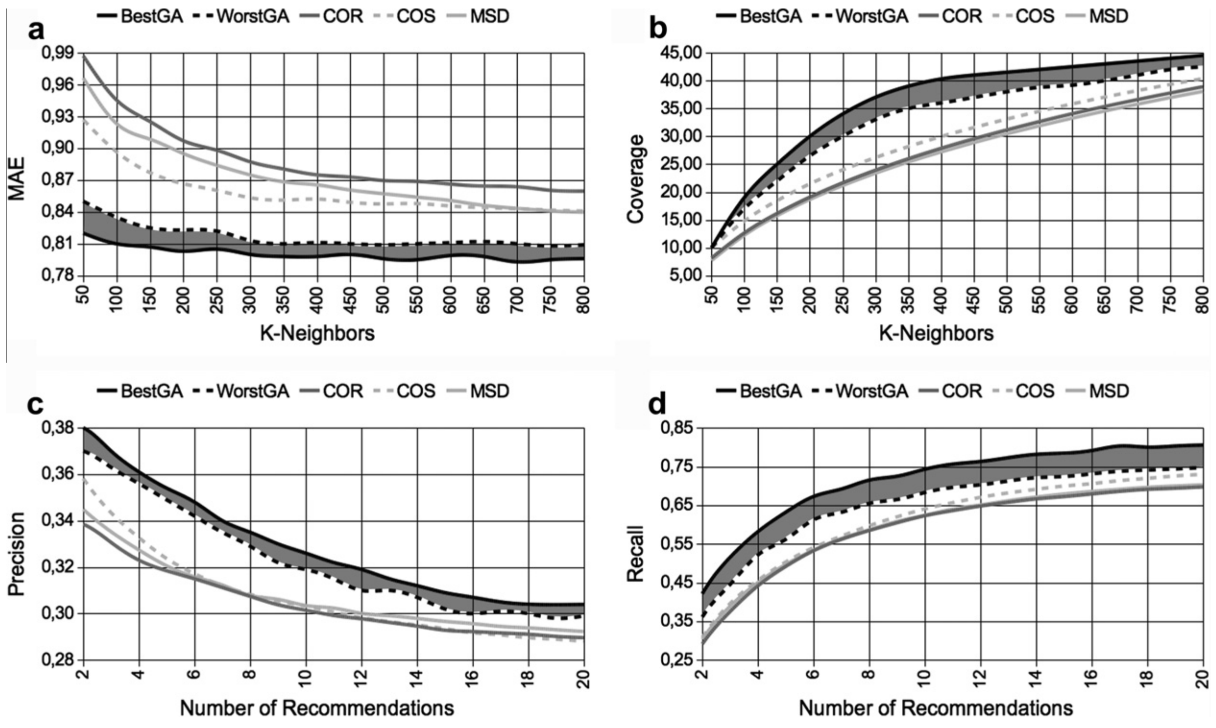

Fig. 3. Traditional metrics and proposed genetic similarity method comparative results using Netflix: (a) accuracy (Mean Absolute Error), (b) coverage, (c) precision, (d) recall.

Table 3
Performance results obtained using: correlation, cosine, MSD and the proponed GA-method.

Metric	Correlation	Cosine	MSD	GA-method
Time (s)	28, 56	29, 19	25, 56	16, 47

5.2. Performance results

Table 3 shows the results obtained in the experiments described in Section 4.2. As may be seen, the time needed to make predictions when using our GA method is 42.44% lower than the time needed when using Pearson correlation.

Since obtaining the k-neighbors for a user takes nearly the whole time for finding the recommending items, the simplicity of the formula used for calculating the similarity function (used for obtaining k-neighbors) is crucial so as to accelerate the process of finding recommendations. Since the formula used in the similarity function in our GA method (see Eq. (1)) is much simpler than the formulae involved in the classical metrics, our similarity functions based on the GA method let us make recommendations faster than the one used for traditional metrics.

6. Conclusions

In this paper we have presented a genetic algorithm method for obtaining optimal similarity functions. The similarity functions obtained provide better quality and quicker results than the ones provided by the traditional metrics. Improvements can be seen in the system's accuracy (MAE), in the coverage and in the precision & recall recommendation quality measures.

The proposed use of GAs applied to the RS is a novel approach and has the main advantage that it can be used in all CF-based RS, without the need to use hybrid models which often cannot be applied, as in many cases no reliable demographic information or content-based filtering information is available.

The GA-metric runs 42% as fast as the correlation, which is a great advantage in RS, which are highly dependent on equipment overloads when many different recommendation requests are made simultaneously (user to user) or on costly off-line running processes (item to item).

Acknowledgements

Our acknowledgement to the *MovieLens* Group and the *FilmAffinity.com* and *Netflix* companies.

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.
[2] M.Y. Al-Shamri, K.K. Bharadwaj, Fuzzy-genetic approach to recommender systems based on a novel hybrid user model, Exp. Syst. Appl. (2008) 1386–1399.
[3] J.T. Alander, On optimal population size of genetic algorithms, Comput. Syst. SoftwareEng. (1992) 65–70.
[4] N. Antonopoulus, J. Salter, Cinema screen recommender agent: combining collaborative and content-based filtering, IEEE Intell. Syst. (2006) 35–41.
[5] J. Bobadilla, F. Serradilla, A. Hernando, Collaborative filtering adapted to recommender systems of e-learning, Knowl. Based Syst. 22 (2009) 261–265.
[6] J. Bobadilla, F. Serradilla, The incidence of sparsity on collaborative filtering metrics, Australian Database Conf. ADC 92 (2009) 9–18.
[7] J. Bobadilla, F. Serradilla, J. Bernal, A new collaborative filtering metric that improves the behavior of recommender systems, Knowl. Based Syst. 23 (6) (2010) 520–528.
[8] J. Bobadilla, F. Ortega, A. Hernando, A collaborative filtering similarity measure based on singularities, Inf. Proc. Manage. (2011), doi:10.1016/j.ipm.2011.03.007.
[9] H. Denis. Managing collaborative learning processes, e-learning applications. 29th Int. Conf. Inf. Tech. Interfaces, (2007) 345–350.
[10] L.Q. Gao, C. Li, Hybrid personalizad recommended model based on genetic algorithm, Int. Conf. Wireless Comm. Netw. Mobile Comput. (2008) 9215–9218.
[11] G.M. Giaglis, Lekakos, Improving the prediction accuracy of recommendation algorithms: approaches anchored on human factors, Interact. Comp. 18 (3) (2006) 410–431.
[12] J.L. Herlocker, J.A. Konstan, J.T. Riedl, L.G. Terveen, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (1) (2004) 5–53.
[13] Y. Ho, S. Fong, Z. Yan, A hybrid ga-based collaborative filtering model for online recommenders, Int. Conf. e-Business (2007) 200–203.
[14] H. Ingoo, J.O. Kyong, H.R. Tae, The collaborative filtering recommendation based on SOM cluster-indexing CBR, Exp. Syst. Appl. 25 (2003) 413–423.
[15] H. Jinghua, W. Kangning, F. Shaohong, A survey of e-commerce recommender systems, Int. Conf. Service Syst. Service Manage. (2007) 1–5.
[16] K. Kim, H. Ahn, Using a clustering genetic algorithm to support customer segmentation for personalizad recommender systems, Artif. Intell. Simulat. 3397 (2004) 409–415.
[17] K. Kim, H. Ahn, A recommender system using GA K-means clustering in an online Shopping market, Expert Syst. with Appl. 34 (2) (2008) 1200–1209.
[18] M. Knights. Web 2.0. IET Comm. Eng. (2007) 30–35.
[19] J.A. Konstan, B.N. Miller, J. Riedl, PocketLens: toward a personal recommender system, ACM Trans. Inf. Syst. 22 (3) (2004) 437–476.
[20] B. Krulwich, Lifestyle finder: intelligent user profiling using large-scale demographic data, Artif. Intell. Magaz. 18 (2) (1997) 37–45.
[21] K. Lang. NewsWeeder: Learning to filter netnews, 12th Int. Conf. Machine Learning, (1995) 331–339.
[22] P. Li, S. Yamada, A movie recommender system based on inductive learning, IEEE Conf. Cyber. Intell. Syst. 1 (2004) 318–323.
[23] K.J. Lin. Building Web 2.0. Computer, (2007) 101–102.
[24] Y. Manolopoulus, A. Nanopoulus, A.N. Papadopoulus, P. Symeonidis, Collaborative recommender systems: combining effectiveness and efficiency, Exp. Syst. Appl. 34 (4) (2008) 2995–3013.
[25] M. Papagelis, D. Plexousakis, T. Kutsuras, Alleviating the sparsity problem of collaborative filtering using trust inferences, LNCS 3477 (2005) 224–239.
[26] C. Porcel, E. Herrera-Viedma, Dealing with incomplete information in a fuzzy linguistic recommender system to diseminate information in university digital libraries, Knowl. Based Syst. 23 (1) (2010) 32–39.
[27] C. Porcel, J.M. Moreno, E. Herrera-Viedma, A multi-disciplinar recommender system to advice research resources in university digital libraries, Exp. Syst. Appl. 36 (2009) 12520–12528.
[28] F. Zhang, H.Y. Chang, A collaborative filtering algorithm employing genetic clustering to ameliorate the scalability issue, IEEE Int. Conf. e-Business Eng. (2006) 331–338.