Expert Systems With Applications 53 (2016) 204–218

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering

Edjalma Queiroz da Silva*, Celso G. Camilo-Junior, Luiz Mario L. Pascoal, Thierson C. Rosa

Instituto de Informática, Universidade Federal de Goiás, Goiás 74001-970, Brasil

A R T I C L E I N F O

Keywords:
Recommender systems
Collaborative filtering
Combining techniques
Genetic algorithms

A B S T R A C T

Recommender systems (RS) are often used as guides, helping users to discover products of their interest. Many techniques and approaches to generate an effective recommendation are available for the system designers. On the one hand, this is interesting because different application's scenarios could have a fittest solution but on the other it can also cause some complexity to select the best technique to address at each state of the database. Thus, choose the best technique for each new state becomes too difficult and frequent for manually select. One of big challenges on RS is turn the techniques more useful for real-world scenarios. Therefore, automate or help the design decision is an important task to improve the usability of RS and reduce its cost. Although many works aims to improve the performance of RS for some scenarios, just a few of them try to help the designers on selection or combination of the techniques through applications' state changes. Therefore, this work proposes an evolutionary approach, called Invenire, to automate the choice of techniques used by combining results of different recommendation techniques. This is a new approach that uses a search algorithm to optimize the techniques combination, and can inspire hybrid methods and expert systems on how automate them. To evaluate the proposal, experiments were performed with a dataset from MovieLens and different collaborative filtering approaches. The results obtained show that the Invenire outperforms all collaborative filtering approach separately in all contexts addressed. The improvement achieved varies from 3.6% to 118.99% depending on the combination encountered and the experiment executed. Thus, the proposal was able to increase the accuracy on the generated recommendations and automate the combinations of techniques.

1. Introduction

In order to eliminate doubts in situations where we have to choose among products or items we are faced with, we usually rely on recommendations passed on by others. These recommendations are given to us directly ("word of mouth") (Shardanand & Maes, 1995) or through texts and videos. Film critics, book reviewers, online social networks, and printed newspapers are examples of influencer. A recommender system helps to increase the capacity and effectiveness of transmitting and receiving suggestions, a well known process in the social relationships among human beings (Resnick & Varian, 1997). In a typical system, people provide evaluations of items they have bought or used. These evaluations are usually represented as ratings.[1] The recommender system uses these gradings from some users to suggest the best *n* items to others. These systems have big challenge to determine the best combination of user expectations and adequate item (products, services or people) to be recommended, i.e., discovering the relationship of interest and options is a major problem (Cacheda, Carneiro, Fernandez, & Formoso, 2011a).

Adomavicius and Tuzhilin (2005) classify recommender systems into three major categories regarding the approach used to generate the recommendations: (i) *content-based* approach, in which similar items to those the user showed preference in the past are recommended; (ii) *collaborative filtering*, which recommends items chosen by people with similar preferences to the user and; (iii) *hybrid* approaches that combine techniques of both previous approaches to attempt to solve some problems inherent to each of them in isolation.

Collaborative filtering (CF) is one of the most used recommendation technologies. This method calculate the similarity between users and uses this information to recommend items not yet tried by the target user (Hu & Pu, 2010). The similarity is based on past reviews of shared items. This similarity is used to generate recommendations of items that were previously evaluated by these

* Corresponding author. Tel.: +556282941869.
E-mail addresses: edjalmasilva@inf.ufg.br, edjalma@ambientinformatica.com.br (E.Q. da Silva), celso@inf.ufg.br (C.G. Camilo-Junior), luizpascoal@inf.ufg.br (L.M.L. Pascoal), thierson@inf.ufg.br (T.C. Rosa).

[1] These ratings are commonly represented as a grade in the range [1, 5] or as a number of "stars" in the same range.

http://dx.doi.org/10.1016/j.eswa.2015.12.050
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

similar users but that were not yet evaluated by the target user (Herlocker, Konstan, Borchers, & Riedl, 1999; Hu & Pu, 2010).

The CF has quickly become popular in the fields of academia and industry. Companies like Google, Amazon and Netflix make great use of this approach because of its significant competitive advantage. The CF approach basically follows four steps (Cacheda et al., 2011a):

1. Calculate the similarity of each user to the target-user (similarity metrics).
2. Select a subset of h neighbors, i.e., users with highest similarity to the target-user, in order to consider the ratings of these neighbors in the prediction.
3. Normalize ratings and compute the predictions considering the evaluations of neighbors with their weights. The weight in this case is the value of similarity between the neighbor and the target-user.
4. Sort items in decreasing order of predicted scores and present the best n items to the target-user.

The collaborative filtering algorithms can be classified into two types: *memory-based* algorithms and *model-based* algorithms. They basically differ in how they process the matrix of ratings (User χ Item). The *model-based* algorithms have two distinct phases. In the first stage, the algorithm handles the matrix of ratings to generate an efficient model that represents the original matrix. In the second step, this generated model is used as the input matrix for the calculation of prediction rating for target user (Cacheda et al., 2011a; Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995). The *memory-based* collaborative filtering uses the entire matrix to calculate its prediction. First, it use any similarity measure to select users (or items) that are similar to the target-user. Then, the prediction ratings of target user are calculated from the ratings of his neighbors. Otherwise, the *memory-based* method is divided into two other algorithms. The first one is called *user-based* algorithms, where the method for obtaining neighbors is based on the user (Shardanand & Maes, 1995). And the second one is called *item-based* algorithms, where neighbors are based on item (Sarwar, Karypis, Konstan, & Riedl, 2000).

Because the algorithms are still inefficient in some cases, the development of new collaborative filtering algorithms has focused mainly on how to provide accurate recommendations (Goldberg, Nichols, Oki, & Terry, 1992). One of the analyzed points is how one can well calculate the similarity between users. Various techniques, e.g. Euclidean, Tanimoto and Pearson correlation, are presented in the literature to do this. On the one hand, the large amount of options is important because gives custom and good solutions for different domains. However, may cause doubt of which technique to choose for recommendation process. Each of these approaches have particularities and its performance depend on the context to be applied, therefore each case must be carefully analyzed before choosing which technique to adopt.

This process has high cost because the designer spends much time running experiments to decide the best technique. Furthermore, even that he has chosen a good one, when significant modifications happen in the database he should repeat the process because the algorithms' performance is highly dependent of rating matrix (created from database). Therefore, we propose the method that automatically combines some rankings of recommendations, resulting from the *memory-based* techniques, to get better result than any of them alone. Our key insight is that combining results one can get the best of each addressed technique without the complexity of choose or turn them to hybrid.

As the task of discovering a good combination manually is a difficult task, it is desired that the combination be automated. For such a matter, the work proposes a genetic algorithm(GA; Holland, 1975; Goldberg & Holland, 1988) able to automate the combination of results of different *memory-based* similarity techniques. Although many hybrid approaches were done (Adomavicius & Tuzhilin, 2005; Burke, 2002; 2007; Lu, Wu, Mao, Wang, & Zhang, 2015), the use of search algorithm for combine techniques and automate the designers' decision was not explored and has a great potential.

The GA was chosen because it is widely used in the literature. Moreover, GAs are known for their flexibility, ease of implementation, and effectiveness in performing global search in adverse environments. In this approach, the GA should be able to generate a list (L) of n items to be recommended. These items are selected from the ranking of techniques used in the combination. Therefore, the list formed by the GA depends on the performance of each technique. The techniques that achieve lower error (RMSE) will have more items among the n finals. An example of the composition of this list, in case of $\|L\| = 10$, would be: 3 items coming from the rank of technique A, 3 items coming from the rank of technique B, and 4 items arising from the rank of technique C, totaling 10 items in the final list proposed by the GA.

Four experiments was designed to test the proposal. The experiment one (Section 5.1) shows that proposal GA (specialist model) outperformed the base techniques in a minimum of 9.028% and a maximum of 48.21%. The experiment two (Section 5.2) shows the results for generalist model constructed by proposal. The idea is measure the impact of generalization in the scenarios. Although the effectiveness is worse than specialist models and then a few base techniques, the efficiency of generalist model is better than the specialist ones. So, it is an option for who wants low computational cost. The experiment three (Section 5.3) shows the performance of generalist model for different states of a database. This model outperformed six from 10 scenarios for a database ($M1$) and obtained the second best averages on all databases, very close to the best ones. Finally, the experiment four (Section 5.4) shows the results from the comparison experiments between generalist models, specialist models and base techniques on different stages of a database. The specialist model outperformed others on all scenarios followed by generalist models. The base techniques were worse in average.

Thus, this work has the following main contributions:

1. An automated and effective approach to select a good combination of recommendation techniques' results.
2. The cost reduction for recommender systems design.
3. A flexible method for different application scenarios.
4. A customizable method able to combine many techniques' results, including some from different paradigms.

The rest of the work is organized as follows: In Section 2 we briefly review some of the research literature related to our work. In Section 3 we present the main theoretical concepts needed to develop this work. In Section 4 we present our proposal. In Section 5 we present experiments and results. In Section 6, conclusions and future work are shown.

2. Related work

The first system created using the collaborative filtering (CF) approach was the *Tapestry* (Goldberg et al., 1992; Resnick & Varian, 1997), which was a system with complete capabilities of filtering electronic documents. For instance, a user can create a filtering rules for *e-mail* such as “Show me all documents answered by other members of my research group”. However, this system required the users to determine the relevant predictive relationships. Thus, it were only valuable in small closed communities where everyone was aware of the interests and duties of other users.

From this, many others works were done to improve the CF systems. There are studies that make comparisons between traditional

CF methods and proposal of hybrid schemes (Fong, Ho, & Hang, 2008). Dellarocas (2000) suggests the use of CF techniques combined with basic (field value) and advanced (frequency domain) mechanisms to predict personalized reputation. Herlocker (2000) proposes adjustments in the CF technique based on application of weight on the similarity coefficient given to each rating, based on the amount of items in common previously evaluated with the target-user. Other work (Qingshui & Meiyu, 2010), the authors propose a hybrid model to improve CF algorithms. The proposed algorithm makes use of a *content-based* approach to get users interested in the collection of a multi-dimensional vector model, and then CF is applied to find their target customers interested in the most similar “neighbors”.

Furthermore, the authors (Lampropoulos, Sotiropoulos, & Tsihrintzis, 2012) propose a cascade hybrid recommendation to the combination of One-Class classification and CF. The article breaks down the problem of recommendation into a cascade recommendation schema of two levels. Thus, it obtains the benefits of the CF methodologies and *content-based* approach. The first level makes use of the *content-based* approach by applying the One-Class classification paradigm to incorporate the user preferences individually (subjective) in the recommendation process approach. The second level has the purpose of assigning specific scores to items to classify them.

On a recent work, Lu et al. (2015) present a survey with a large review of up-to-date application developments of recommender systems. This paper clustering the applications into different categories like e-government, e-business, e-commerce/e-shopping, e-library, e-learning, e-tourism, e-resource services and e-group activities. Moreover, analyze the most commonly used, for each application categories, recommendation techniques, like traditional methods CF, content-based, knowledge-based, and hybrid methods. The authors emphasize that recommender systems (RS) has being used on many real-world applications and it is thus vital improve the quality of recommendation techniques and its applicability. Therefore, how be useful and effective for real-world scenarios is a big challenge for RS and a hot topic.

Some researchers have sought to combine different areas of computer science with the recommender systems to address big challenges. One of these areas is information retrieval, as seen in the work of Liu, Koutrika, and Wu (2015) that addresses a common problem nowadays of reading different documents on-line. The authors report that while reading, people may have difficulty understanding a passage or wish to learn more about the topics covered by it, hence they may naturally seek additional or supplementary resources for these specifics topics. These resources should be close to the passage both in terms of the subject matter and the reading level. However, using a search engine to find such resources interrupts the reading flow and, it is also an inefficient, because most of web search engines and recommender systems do not support large queries or do not understand semantic topics. To address these challenges, Liu et al. present a system that enables on-line reading material to be smoothly enriched with additional resources that can supplement or explain any passage from the original material for a reader on demand. The system facilitates the learning process by recommending learning resources (documents, videos, etc.) for selected text passages of any length. The recommended resources are ranked based on two criteria (a) how they match the different topics covered within the selected passage, and (b) the reading level of the original text where the selected passage comes from.

Another field that has proven to be a great ally to recommender systems, especially in the movie recommendation domain, is Semantic Web. Among the papers regarding this area, the work of Colombo-Mendoza, Valencia-García, Rodríguez-González, Alor-Hernández, and Samper-Zapater (2015) stands out by presenting a context-aware mobile recommender system based on Semantic Web technologies. The system uses a domain ontology primarily serving a semantic similarity metric adjusted to the concept of “packages of single items”, in addition to also use location, crowd and time to compose a mobile system named RecomMetz. It has unique features: (1) the items to be recommended have a composite structure (movie theater + movie + showtime), (2) the integration of the time and crowd factors into a context-aware model, (3) the implementation of an ontology-based context modeling approach and (4) the development of a multi-platform native mobile user interface intended to leverage the hardware capabilities (sensors) of mobile devices.

Other great ally field is the evolutionary computation (EC). Many works were done based on great optimization capacity of evolutionary algorithms. For instance, Hwang, Su, and Tseng (2010) use genetic algorithm to optimize the recommendations based on personal preferences. The work of Salehi, Pourzaferani, and Razavi (2013) proposes an evolutionary method to improve the quality of recommendation for learning materials based on their attributes. Moreover, the works of Bobadilla, Ortega, Hernando, and Alcalá (2011b) and Pascoal, Camilo, da Silva, and Rosa (2014) propose a metric to measure similarity between users based on genetic algorithms and the paper (Al-Shamri & Bharadwaj, 2008) proposes a fuzzy-genetic approach to create a hybrid user model.

To the best of our knowledge, just our preliminary work of da Silva, Camilo, Pascoal, and Rosa (2014) uses EC or other search algorithm to combine results from recommendations systems approaches. This paper introduced the idea of use search algorithm to combine results of recommendation techniques in order to automate the choice of techniques. The experiments were very small just to present the idea, calibrate some parameters, and get more insights. Only one scenario that compare GA and base CF techniques was evaluated. The results shown a potential of GA and directed this new and extended work.

The most related and closest works of this paper are about hybrid approach, especially the mixed and weighted approach (Burke, 2007). The papers based on mixed approach try to present options from different technique in a side-by-side way. For example, ProfBuilder system in Ahmad Wasfi (1999) and PTV system in Smyth and Cotter (2000). Comparing with our proposal, this approach minimizes the cost of design decision too, because designers can use at the same time many techniques to make recommendations, and not requires of reruns during the database changes like our method requires. However, it increases the decision time for the users because they have many recommended items to analyze. The works based on weighted approach calculate the score of each item by a numerical combination of base techniques’ scores for this item. Comparing with our proposal, this approach achieves an effective recommendations based on techniques combination too, and not requires a high computational cost like our evolutionary method. However, it keeps a high complexity decision for system designer, who should discover the good weight for each technique previously. For example, the paper (Mobasher, Jin, & Zhou, 2004) used empirical means to find the best parameters for the system. Moreover, a few more works are presented in recent surveys (Adomavicius & Tuzhilin, 2005; Burke, 2002; 2007; Lu et al., 2015) about mixed and weighted hybrid approaches. But none of them neither uses search algorithms to optimize the combinations of results nor focus on reduce the efforts of the system designers.

3. Basic concepts

This section presents some background concepts used in this work as *recommender systems* (Section 3.1), *collaborative filtering* (Section 3.1.1), *root mean squared error* (Section 3.1.2) and *genetic algorithm* (Section 3.2).

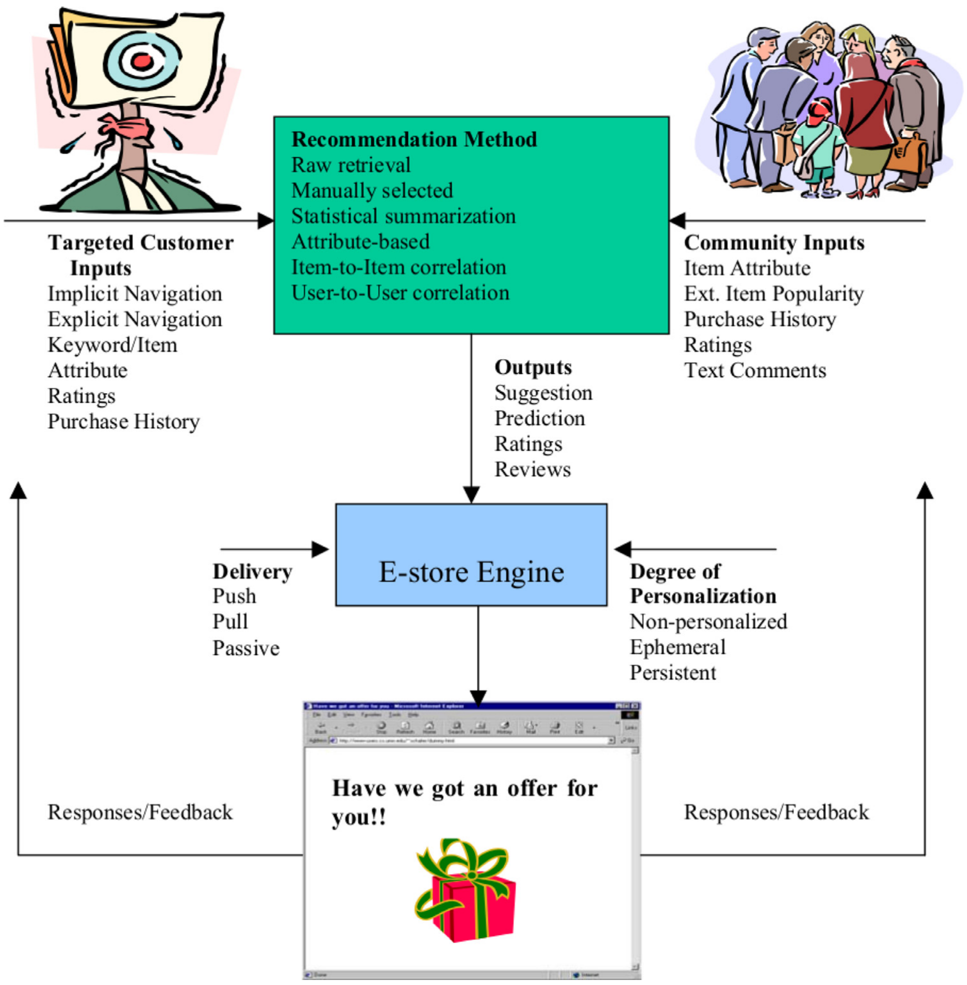

Fig. 1. Taxonomy of recommender systems proposed by Schafer et al. (2001) for e-commerce.

3.1. Recommender systems

Recommender systems (RS) have become an important area of research since the emergence of the first articles on collaborative filtering in the mid-1990s. There has been much work performed in industry and academia with regard to development of new approaches to RS over the last decade.

Over the years RS were classified in different ways by several authors; however, there is not a classification completely accepted by the user community, professionals and researchers. Schafer, Konstan, and Riedl (2001) modeled the architecture of a recommender system and produced a fairly comprehensive taxonomy that considers various characteristics of an RS.

The taxonomy architecture (see Fig. 1) proposed by Schafer et al. (2001) for e-commerce involves three distinct modules, each of which can be modeled and implemented in different ways. This division into blocks facilitates the understanding of these systems. The blocks are separated as follows: *Target-User* is the module responsible for collecting information about the target-user; *Community* is information about interactions of the target-user and other users with the system. These interactions occur at the time when the user evaluates a product, for example; And *Output* represents the system response as a suggestion of product or service. The flow of interaction between these three modules may be observed in Fig. 1.

Also, according to Schafer et al. (2001), applications aimed at generating "recommendations" to users in e-commerce systems combine information about the target-user with the communities where products and the user are located. Thus, the websites use decisions on the level of customization and delivery method to transform them into specific recommendation packages, ensuring personalized recommendations. Comments and ratings of a user about the recommendation received or even about a specific product can generate additional inputs for future recommendations.

In accordance with Bobadilla, Ortega, Hernando, and Gutierrez (2013), the taxonomy divides the methods of recommendation into two categories: *model-based* and *memory-based. Memory-based* methods (Adomavicius & Tuzhilin, 2005; Candillier, Meyer, & Boullé, 2007; Symeonidis, Nanopoulos, & Manolopoulos, 2009) can be defined as methods that (a) act only on the ratings matrix of users and (b) does the use of any generated before the evaluation process reference (that is, the results are always updated). *Memory-based* methods usually use the similarity metrics described in Table 2 for the distance between two users or two items, based on their matrix ratings. *Model-based* methods (Adomavicius & Tuzhilin, 2005; Su & Khoshgoftaar, 2009) use the ratings matrix to learn a model which is then used to make predictions of ratings. Among the most widely used models have *Bayesian classifier* (Park, Hong, & Cho, 2007), *neural networks* (Roh, Oh, & Han, 2003), *fuzzy systems* (Yager, 2003), *genetic algorithms* (Ho, Fong, & Yan, 2007), *latent features* (Hofmann, 2004; Zhong & Li, 2010) and the *matrix factorization* (Luo, Xia, & Zhu, 2012), among others.

The graph in Fig. 2 (Bobadilla et al.) shows the traditional methods of recommendation techniques and algorithms for recommendation process, as well as their relationships and groupings.

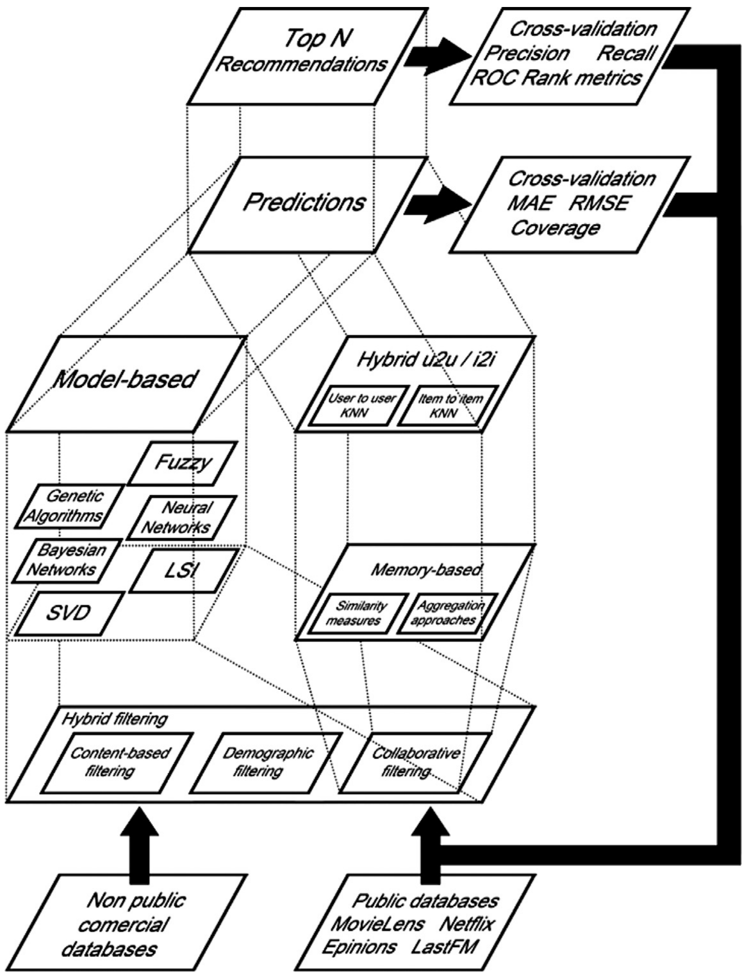

Fig. 2. Traditional recommendations models and their relationships (Bobadilla et al., 2013).

Table 1
Example of a rating matrix.

	Item(i_1)	Item(i_2)	Item(i_3)
User(u_1)	5.0	3.0	2.5
User(u_2)	2.0	2.5	5.0
User(u_3)	2.5	–	–
User(u_4)	5.0	–	3.0
User(u_5)	4.0	3.0	2.0

3.1.1. *Recommender systems based on collaborative filtering*

A system based on CF assumes that if two users have similar interests, then users will demonstrate interest for the same products. In general, consider a list of users $U = \{u_1, u_2, \ldots, u_{\|U\|}\}$ and a list of items $I = \{i_1, i_2, \ldots, i_{\|I\|}\}$.

Each user u_i has a list of items m for which he has expressed interest. Thus, if $m \subset I$ (it is possible that m is a null set), there is a distinguishable user $U_a \in U$, called target-user a, for which it is a task of collaborative filter to find an item of interest, in particular seeking recommendations. Thus, there will be a list of n items, $n \subset I$, in which the target-user will be more interested. The recommended list should be of items not evaluated by the target-user, sorted in decreasing order of values of predicted scores by the collaborative filter. This interface of collaborative filtering algorithms is also known as "Top-N" Recommendation (Karypis, 2001).

Taking Table 1 as an example, we can show how to apply in practice the collaborative filtering. The first step of the CF system is based on searching for users with similar habits of consumption, i.e., calculating the similarity among users. When analyzing users 1 and 5, for the item i_1, the difference between their ratings is 1.0; in i_2 there is no difference and for (i_3) the difference is 0.5. Thus, we could say that users 1 and 5 are similar. By the same reasoning, users 1 and 2 would not be so similar. The calculation of similarity can occur only on items that both users have expressed preference. Table 1 is usually referred to as a *rating matrix*.

To calculate the similarity between users there are several techniques (Cacheda et al., 2011a; Owen, Anil, Dunning, & Friedman, 2011). The most common are shown in Table 2, where:

- $w_{a,\,u}$ is the correlation of the target-user a with a given user u.
- $r_{a,\,i}$ is the rating that the target-user gave for the item i.
- $\overline{r}_a$ is the average of all ratings of the target-user (a).
- w_a is the expected utility of the item i for user a.
- d is the default rating (generally a non-committal rating, or slightly negative).
- α is the half-life. The half-life is the rank of the item on the list such that there is a 50% chance that the user will view that item.

The second step of the *CF-based* system is to select a subset of users with higher similarity. After that, the next step is to calculate the predictions. The prediction is the act of inferring what appraised value would give the user a product that he has not assessed yet. An example of this calculation, noting the Table 1, would be filling in the gaps left by the user u_3 on items i_2 and i_3 and user u_4 on the item i_2. Therefore, the prediction is made independent of the technique used, because it is generated by a

Table 2
Techniques for calculating similarity.

Technique	Equation	Reference
Pearson correlation	1	Adomavicius and Tuzhilin (2005)
Euclidean	2	Manning, Raghavan, and Schütze (2008)
Cosine	3	Adomavicius and Tuzhilin (2005)
Spearman rank[a]	1	Herlocker (2000)
Tanimoto	4	Marmanis and Babenko (2009)
Loglikelihood test	5	Breese, Heckerman, and Kadie (1998); Herlocker (2000)

Equation

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i}-\bar{r}_a)(r_{u,i}-\bar{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{a,i}-\bar{r}_a)^2\sum_{i=1}^{m}(r_{u,i}-\bar{r}_u)^2}} \quad (1)$$

$$w_{a,u} = \sqrt{\sum_{i=1}^{m}(r_{a,i}-r_{u,i})^2} \quad (2)$$

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i}*r_{u,i})}{\sqrt{\sum_{i=1}^{m}(r_{a,i})^2}\sqrt{\sum_{i=1}^{m}(r_{u,i})^2}} \quad (3)$$

$$w_{a,u} = \frac{|m_a \cap m_u|}{(|m_a|+|m_u|)-(|m_a \cap m_u|)} \quad (4)$$

$$w_a = \sum_i \frac{\max(r_{a,i}-d,0)}{2^{(i-1)/(\alpha-1)}} \quad (5)$$

[a] The Spearman rank correlation coefficient is similar to Pearson, but rather than compute a correlation based on the original preference values (Eq. (1)), it computes a correlation based on the relative rank of preference values (Herlocker, 2000).

weighted average of ratings of neighbors that have an acceptable coefficient of similarity. According to the authors (Adomavicius & Tuzhilin, 2005; Bobadilla, Hernando, Ortega, & Bernal, 2011a; Cacheda et al., 2011a), the prediction can be calculated by Eq. (6).

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{h}(r_{u,i}-\bar{r}_u)*w_{a,u}}{\sum_{u=1}^{h}|w_{a,u}|} \quad (6)$$

Where h is the amount of best neighbors and it is at the discretion of each system that uses collaborative filtering.

Finally, the sorting is performed in decreasing order of values of the predictions and returns the best n items as recommendations.

3.1.2. Evaluation of recommending systems

The input to a recommender system is a rating matrix M, similar to that presented in Table 1. To evaluate a recommender algorithm A, another matrix M_t is obtained from M by removing k ratings. Matrix M_t is used as input to the recommending algorithm to be evaluated. The objective of algorithm A is to predict correctly the values of the ratings absent from the matrix. Let $R = \{r_1, r_2, \ldots, r_x\}$ be the set of ratings absent from M_t. Algorithm A produces a set of predictions $P = \{p_1, p_2, \ldots, p_x\}$ when trying to guess the corresponding values in R. The evaluation of A is done by computing the accumulated error produced by A in its predictions.

There are different metrics used to compute the error of a recommender algorithm. In this work we use the *root mean squared error* (RMSE), which became extremely popular in recent years, after being used in the Netflix Prize competition (Cacheda et al., 2011a). For a given algorithm A the RMSE is computed as described in Eq. (7).

$$\text{RMSE}_A = \sqrt{\frac{\sum_{i=1}^{x}(p_i-r_i)^2}{x}} \quad (7)$$

Where x is the amount of items that were recommended, p_i the prediction of the algorithm and r_i the corresponding true rating. The most accurate algorithm is the one with the slowest RMSE value for a given matrix M_t.

3.2. Genetic algorithm—GA

The first step when using GA is to represent each possible solution χ in the solution-space as a sequence of symbols chosen from a finite alphabet $\mathcal{A}$, which varies according to the target problem. Each sequence s represents an individual (solution) and can be seen (metaphorically) as a chromosome. Each symbol in s is considered a gene. Most GA solutions use a constant-size population of chromosomes and each chromosome has also a fixed size (Engelbrecht, 2007; Lones, 2011). After definition of each chromosome for the specific problem to be solved by an GA, an initial population POP_0 of candidate chromosomes (solutions for the problem) is created.

GAs are iterative algorithms and in each iteration a new population is derived from the population existing at the begin of the iteration. The control flow of a GA corresponds to the following steps (Mitchell, 1998):

1. Create the initial population POP_0 of chromosomes.
2. Evaluate each chromosome in the current population.
3. Chose parent chromosomes to generate new chromosomes.
4. Apply genetic operators to the chosen parents in order to generate new chromosomes which will compose the next population (the next generation).
5. Kill the old population.
6. Evaluate each new chromosome. whether time is over or the best chromosome has been found, stop; otherwise, go to step 3.

In this work we use GA to automatically combine *memory-based* collaborative filtering techniques derived from distinct similarity measures, with the objective to demonstrate our proposed hypothesis. The next section discusses how we use an GA to perform this combination.

4. Proposal

In this section we describe a proposal, called *Invenire*, that produces a list (L) of items to be recommended. L is composed of the best items from the rankings produced by each individual

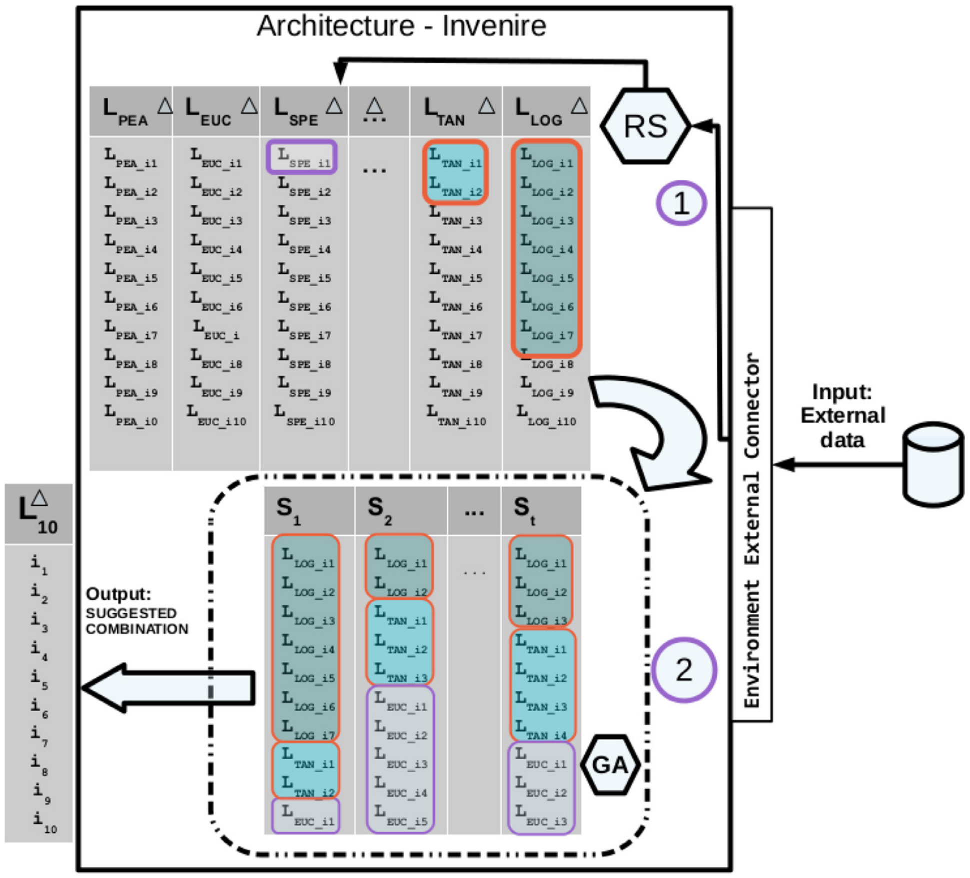
Architecture - Invenire
RS
1
2
GA
Input: External data
Environment External Connector
Output: SUGGESTED COMBINATION
L_PEA, L_EUC, L_SPE, L_TAN, L_LOG
S_1, S_2, ..., S_t
L_10

Fig. 3. General flow of Invenire.

collaborative filtering technique used (Pearson, Euclidean distance, Spearman, Tanimoto an Loglikelihood). The GA is used as a search algorithm to optimize the results combination of CF-techniques.

Thus, the aim of the GA is to obtain a good solution (chromosome) to the problem of choosing the appropriate number of elements on the top of each individual ranking produced by the basis techniques. The chosen elements are used to compose the final ranking. Consequently, each position (gene) in a candidate solution represents an amount of a base ranking (from basis techniques) to be used in the final ranking *L*, see Section 4.4.1. The choice of GA will be justified by the analysis in Section 4.1.

4.1. Mathematical modeling of the problem

This problem can be defined as a combination of the results of many recommendation techniques with the goal of producing a recommended item list with lowest prediction error. Thus, we define $\mathcal{T} = \{t \mid t \text{ is a technique of recommendation}\}$ as a set of basis techniques for recommendation. This set has a cardinality *c* belonging to the set of natural numbers $\mathcal{N}^*$.

Each technique produces a list of recommendations $\mathcal{L}_t = \{i \in \mathcal{I} \mid i \text{ is a recommendation of technical } t\}$. This list has a cardinality *l* belonging to the set of natural numbers $\mathcal{N}^*$.

Thus we have *l* lists of recommendations, each of size *s*, where *s* is the number of items recommended by each technique. Thus, $s \times l$ recommended items may occur. However the system should ultimately recommend only *s* items to the target user. At this time the Invenire acts to combine the results from each list to generate a list $\mathcal{L}_{\text{final}}$ of cardinality *s*. This list $\mathcal{L}_{\text{final}}$ should be a list that achieves lower error rate prediction for all users.

This problem becomes complex because of the amount of existing techniques and their possible combinations. In this work, each similarity measure, normally applied to RS-techniques, is considered a basis technique, such as the Pearson correlation, Euclidean distance, the measure of Cosine, among others. However, without loss of generality, a technique can be regarded as a complete recommendation method, for example, a *model-based* approach.

4.2. Architecture of Invenire

The objective of software architecture is to define the components of a system, their external properties and their relationship with humans, other systems and sensors (Lopes, Silveira, Silveira, Stepaat, & Kung, 2012).

Fig. 3 shows the architecture used in Invenire where:

- *RS* is a recommender system.
- L_{PEA}, L_{EUC}, L_{SPE}, L_{TAN} and L_{LOG} are lists of recommended items from each collaborative filtering techniques configured in Invenire.
- *GA* is the genetic algorithm.
- S_1, S_2 and S_t are candidate solutions, each of which represents a chromosome (individual) in the GA's population.
- L_{10} is a list of ten items to be recommended.

As shown in Fig. 3, the architecture is divided into two modules:

1. Recommender system: Aims to generate lists of the items to be recommended to the target user. Each base technique produces one list.
2. Genetic algorithm: Aims to choose the best combination of items produced by the previous module.

Fig. 3 also shows an element termed "Environment External Connector". This connector is responsible for converting the data

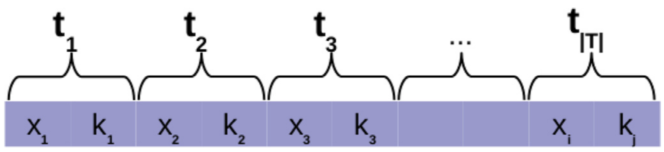

Fig. 4. The structure of a chromosome used by the proposed GA.

from the external environment into useful information for Invenire. Thus, the Invenire can abstract representations of the types of information from different data sources.

4.3. Computational costs

To be able to define the final list $\mathcal{L}_{\text{final}}$, the lists of all the techniques in evaluation is necessary. Making an asymptotic analysis, assume that the recommendation generated by each technique is $O(u \times n)$ (Cacheda, Carneiro, Fernandez, & Formoso, 2011b), where u is the number of users and n is the number of items not rated in a database.

For simplicity, we assume that the *user* and textititems not rated have the same cardinality. Thus, we have that each technique is quadratic ($O(n^2)$). The Invenire has to calculate the lists of t techniques, so the complexity in the module 1 is given by:

$$O(t \times n^2) \tag{8}$$

If the search technique used is exhaustive in the module 2 the complexity is $O(2^{s \times t})$, given the search space defined in Eq. (9).

$$\frac{(s \times t)!}{s! \times ((s \times t) - s)!} \tag{9}$$

Thus, the problem to be solved has complexity $O(2^{s \times t})$. We can see that even with a few techniques, the computational cost is very high. Therefore, we opted for the use of a genetic algorithm (metaheuristic) in the module 2.

4.4. The proposed GA

The initial population POP_0 used by the GA is created randomly, with x_i receiving any random values since the sum of them satisfies the constraint in Eq. (10).

$$\sum_{t=1}^{\|T\|} x_t = \|L\| \tag{10}$$

where $\|L\|$ is the size of the final ranking produced by GA, which is a value known in advance.

4.4.1. Chromosome representation

Consider the set of techniques $T = \{t_1, t_2, t_3, \ldots, t_{\|T\|}\}$. A chromosome (individual) is composed of $\|T\|$ genes, where each gene corresponds to a pair as shown in Fig. 4.

The x_i represents the quantity of items obtained from the list (ranking) produced by technique $t_i \in T$ that will be used to compose the final ranking. The second element of the pair (k_i) corresponds to the number of ratings that are removed from the rating matrix M to generate the reduced matrix M_t.

Table 3 shows a sample of chromosomes that satisfy the restriction in Eq. (10) for $\|L\| = 10$.

4.4.2. Workflow of the AG

For each chromosome in a given population POP (initially $POP = POP_0$) the GA computes the accumulated RMSE (RMSE_t) for each technique t. The accumulated values of RMSE for techniques are used to compose the fitness function that is used to evaluate each chromosome as will be explained later in this section.

Table 3
Examples of chromosomes (each line one chromosome).

PEA		EUC		SPE		TAN		LOG	
X	k	X	**k**	X	k	X	k	X	k
3	0	2	0	2	0	1	0	2	0
1	0	6	0	1	0	1	0	1	0
0	0	0	0	0	0	0	0	10	0

Fig. 5 shows how the GA computes the RMSE for each technique and how these partial RMSE computations are totalized to generate the RMSE for a chromosome. Where:

- t is one of the technique in T.
- u is one user in M.
- M_{full} is the complete matrix containing all ratings of users to items as exemplified in Table 1. In addition to giving the result in matrix M, the matrix M_{full} aims to provide the value of the actual rating (r_i) that the user entered for the item and this value is then used for the calculation of RMSE_u.
- M is the matrix composed by the ratings of 5% of the users contained in M_{full} −283 the first users found in M_{full}.
- k is the number of ratings made by the target-user u, removed from M to obtain the matrix M_t.
- M_t is the resulting reduced matrix obtained from the rating matrix M by eliminating k of the ratings of user u.
- x is the number of recommendations to be generated for the computation of RMSE user u in the technique t.
- RMSE_u is the value of RMSE for the predictions of a technique t to a given user u.
- RMSE_t is the accumulated values of RMSE_u due to the predictions made by technique t for each user u.

To compute the accumulated RMSE_t for each technique t the GA produces a matrix M_t for each user u. Matrix M_t is obtained by copying the ratings from M and next removing k_t ratings of user u, where k_t corresponds to the second component of the tth gene in the chromosome. Then, predictions (p_i) are generated for user u using technique t applied over the matrix M_t. The real values (r_i) of each item are obtained by performing queries on matrix M_{full}. Next, the value of RMSE_u is computed for user u using Eq. (7), substituting x in the equation for the value of x_t in the chromosome. The value of RMSE_u is then accumulated in RMSE_t. The above processing is repeated for each user.

The accumulated values RMSE_t are computed for each technique t as just explained and these values are also summed up to obtain the grand total $\text{RMSE}_{\text{total}}$.

After the accumulated values RMSE_t have been computed the chromosome is evaluated using the fitness function expressed in Eq. (11).

$$\text{MIN}(f(x)) = \frac{w_1 \sum_{t=1}^{\|T\|} \text{RMSE}_t + w_2 \sum_{t=1}^{\|T\|} (1 - q)}{\|T\|} \tag{11}$$

Where:

- RMSE_t is the accumulated value of RMSE for technique t.
- w_1 and w_2 are input parameters for technique t for the GA, with values in the range [0, 1]. The $w1$ is used as a weight of the importance of the technique in the composition of the final ranking produced by GA. The w_2 corresponds to the weight given to the quantity of ratings that are removed for the target-user when generating the matrix (M_t) as shown in Fig. 5.
- q is the value computed by Eq. (12), where R is the quantity of ratings of each user in the rating Matrix M.

$$q = \frac{k}{R - 1} \tag{12}$$

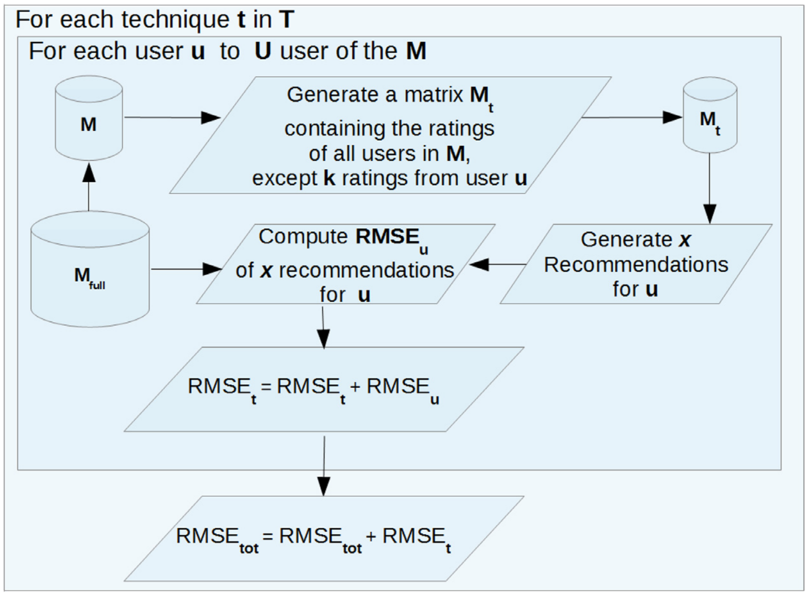
For each technique **t** in **T**
For each user **u** to **U** user of the **M**
M
Generate a matrix M_t containing the ratings of all users in **M**, except **k** ratings from user **u**
M_t
M_{full}
Compute $RMSE_u$ of *x* recommendations for **u**
Generate *x* Recommendations for **u**
$RMSE_t = RMSE_t + RMSE_u$
$RMSE_{tot} = RMSE_{tot} + RMSE_t$

Fig. 5. Computation of the RMSE for each technique and for the chromosome (candidate solution).

The fitness function (Eq. (11)) is composed of two components. The first component ($\frac{w_1 \sum_{t=1}^{\|T\|} \text{RMSE}_t}{\|T\|}$) is responsible for the summarization of the error in the prediction process. To do this, the RMSE was chosen as a metric, because it has been used in many studies that measures the accuracy of predictions (Cacheda et al., 2011a). The second component ($\frac{w_2 \sum_{t=1}^{\|T\|} (1-q)}{\|T\|}$) represents the total performance of each technique taking into account the complexity of the scenario in which it was inserted. The complexity to generate predictions is related to the amount of the target-user ratings. Thus, the sum of these components guide the GA through search space.

The whole process described above is repeated for each chromosome of population *POP*. Then the chromosomes are operated by genetic operators (crossover and mutations) to produce a new population *POP*′. Finally, the GA repeats the whole process with the new population POP′. The generation of new populations is repeated until the stop condition is achieved.

5. Experiments and results

We want to evaluate our proposal comparing it with traditional methods of CF, using Pearson's correlation, Euclidean distance, Spearman Correlation, Tanimoto coefficient and Loglikelihood. In this work we used the approaches of collaborative filtering because they are widely adopted in literature.

We used also the movieLens dataset (*MovieLens 1M Data Set (.zip)* [2]) to perform the experiments. These files contain 1, 000, 209 anonymous ratings of approximately 3900 movies made by 6040 MovieLens users who joined MovieLens in 2000. We called this database as *1M Data Set*. The 5% of *1M Data Set* as *M* was used in the experiments, which corresponds to 283 users with 10 ratings each. To implement the proposed model we used two frameworks enshrined in their respective fields, Apache Mahout[3](Owen et al., 2011) as recommender and Jenes 2.0[4] as the GA engine. Finally, the experiments described in the following sub-sections are

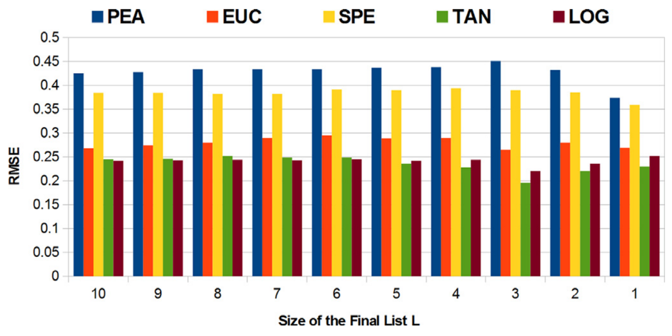

Fig. 6. RMSE of CF-techniques for each size of final list *L*.

an extension of the work described in da˜Silva et al. (2014), thus the configuration of the GA parameters are the same.

5.1. Experiment one—GA vs CF in isolation techniques

In this experiment we fixed the values of every k_t in the chromosomes of the populations to zero. This was done because the intent of this experiment was to compare the accuracy of the combination of results produced by GA with the accuracy of each technique alone.

Table 4 shows an example of the target users, 22, 23 and 26, where *R* is the actual rating given by the user to the item and *P* is the forecast provided by the recommender. Taking as an example the item 111 for the user 22, it is observed that it was recommended by all CF-techniques. In Pearson's list it appears in the first position, in the Spearman's list it is the second and for Euclidean, Tanimoto and Loglikelihood it is the sixth item in the lists. Thus we can see that the items fluctuate the position in recommendation lists and that the items recommended by a technical t_1 will not necessarily be generated by the technique t_2, for example.

Fig. 6 shows the results of RMSE for each of the five selected techniques. Where *PEA* is the result obtained using the Pearson correlation alone, *EUC* for Euclidian, *SPE* for Spearman, *TAN* for Tanimoto and *LOG* for Loglikelihood. The PEA and SPE were the worst and more affected by the size of final list *L*.

Fig. 7 shows the RMSE values obtained in each technique, whether used a single technique to recommend a final list *L* of size

[2] The database may be obtained through the page link: http://www.grouplens.org/node/73
[3] http://mahout.apache.org
[4] http://jenes.intelligentia.it/

Table 4
Forecast by CF-techniques for some users.

	Pearson			Euclidiana			Spearman			Tanimoto			Loglikelihood		
Usuário	Item	*R*	*P*	Item	*R*	*P*	Item	*R*	*P*	Item	*R*	*P*	Item	*R*	*P*
22	111	4	5	296	3	4.73	318	5	5	104	3	4.48	318	5	4.49
	163	3	5	858	4	4.6	111	4	4.37	318	5	4.4	104	3	4.34
	318	5	4.94	457	3	4.56	296	3	4.36	296	3	4.35	296	3	4.25
	112	2	4.31	288	2	4.39	112	2	4.09	457	3	4	457	3	4
	104	3	4.11	318	5	4.38	288	2	4	288	2	4	288	2	4
	344	2	4	111	4	4.3	457	3	4	111	4	3.88	111	4	3.99
	288	2	4	608	5	4.21	223	4	4	165	3	3.76	16	4	3.71
	457	3	4	234	3	4.17	344	2	4	70	4	3.7	165	3	3.67
	95	4	3.98	589	4	4	16	4	3.77	16	4	3.6	70	4	3.62
	292	3	3.84	480	2	4	231	3	3.71	10	3	3.55	145	2	3.59
23	29	3	4.71	29	3	4.81	296	5	4.53	29	3	4.44	29	3	4.37
	260	5	4.66	296	5	4.72	111	3	4.52	6	3	4.2	6	3	4.11
	296	5	4.62	260	5	4.63	29	3	4.52	260	5	4.17	260	5	4.01
	6	3	4.15	608	4	4.61	260	5	4.5	70	4	3.87	296	5	3.98
	111	3	4.1	162	2	4.48	6	3	4	296	5	3.82	70	4	3.75
	52	4	4	595	3	4.36	593	4	4	111	3	3.78	11	3	3.75
	161	2	3.78	111	3	4.35	457	3	4	161	2	3.7	161	2	3.74
	34	3	3.74	589	4	4.33	52	4	4	52	4	3.67	11	2	3.59
	70	4	3.64	235	3	4.28	70	4	3.82	11	2	3.66	34	3	3.58
	318	3	5	6	3	4.27	161	2	3.79	34	3	3.6	52	4	3.57
26	3	2	5	593	3	5	318	4	5	260	3	4.89	260	3	4.87
	168	4	4.79	1196	2	5	160	3	5	318	4	4.45	318	4	4.49
	318	4	4.66	480	4	4.83	48	3	5	356	5	4.45	356	5	4.47
	260	3	4.65	356	5	4.73	62	4	4.78	104	2	4.4	104	2	4.34
	207	5	4.43	260	3	4.68	260	3	4.41	16	4	3.83	1	3	3.98
	1	3	4.16	377	4	4.5	168	4	4.33	1	3	3.83	207	5	3.98
	104	2	4	318	4	4.4	1	3	4.14	539	2	3.67	364	3	3.84
	45	5	3.4	780	3	4.33	207	5	4.12	364	3	3.64	122	2	3.81
	16	4	3.37	207	5	4.11	356	5	4	207	5	6.56	16	4	3.75
	261	2	3.33	364	3	4	11	3	3.43	339	5	3.5	168	4	3.62

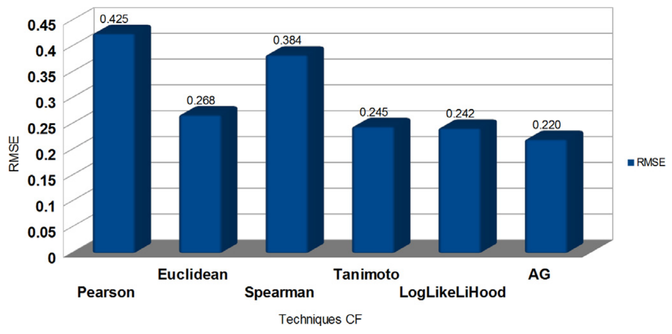

Fig. 7. RMSE of basic techniques and GA.

10. This graphic also shows the GA's result. The GA made the combination of techniques and got a great solution who had 3 items of Tanimoto and 7 items of Loglikelihood, totaling 10 items in the final list.

The GA was the best among the methods with only 0.22022 error, which represents a decrease of 9.028% in the RMSE compared to the best result obtained using a single CF technique to generate recommendations (Loglikelihood). If we compare with Pearson the difference increases to 48.21%. Thus, it is evident that the combination of the techniques outputs can reach a smaller error than the use of only one in the recommendation process based on CF.

5.2. Experiment two—The learned model in a matrix M applied to other matrices

The previous experiment demonstrated that the combination of the CF techniques is better than the choice of one isolated. During the execution of the GA using the matrix M as input, we verified that the best combination (best individual) is the one making use of 3 items from Tanimoto technique and 7 items from Loglikelihood to compose a final list (L_n) of 10 items; we called it as GA_M (specialist on M).

According to Table 5, the GA takes about 13 hr and 13 min to find the best combination. This time is very high when compared to how much time it takes to generate the recommendation using an isolated CF technique (about 1 s).

Therefore, the objective of this experiment is to verify whether the best individual (GA_M) is generalizable to other M (input matrix), which would considerably reduce the computational cost.

We executed the GA_M using $M1$ as an input matrix and after that we did the same procedure with the chosen CF techniques. The same procedure was also performed using the matrix $M2$. In this experiment we set $k_t = 0$.

The creation of $M1$ and $M2$ follows the same criteria that was used for matrix M. $M1$ and $M2$ are matrices formed by the extraction of 5% of the users contained in M_{full} (283 users from M_{full} different from M). It is worth noting that $M1 \cap M2 = \emptyset$.

Thus, to compute the RMSE of each technique and GA_M, we replaced the matrix M from Fig. 5 with the matrix $M1$ and the result can be observed in Table 5. We also replace the matrix M of Fig. 5 with the matrix $M2$ and the result can be seen in Table 5 too.

Table 5 shows that the learned individual (GA_M) in a matrix (M) can be utilized in another matrix ($M1$ e $M2$) without a large increase of error (RMSE) and execution time, in comparison to the adopted CF-techniques.

In $M1$, the GA_M underperformed the *LOG* (reduction in RMSE of 1.05%) and *EUC* (reduction in RMSE of 11.98%); However, if we compare with the techniques of *TAN* (increase in RMSE of 1.36%), *SPE* (increase in RMSE of 47.02%) and *PEA* (increase in RMSE of 75.96%), the GA_M outperformed them.

Comparing the results achieved by GA_M and other techniques in $M2$, it is observed that the *LOG* (reduction in RMSE of 7.83%) and *TAN* (reduction in RMSE of 15.33%) achieve better than GA_M.

Table 5
Results of the generalist GA (GA_M) learned in the matrix M and applied in the matrices $M1$ and $M2$

	M		***M1***		***M2***	
	RMSE	Time	RMSE	Time	RMSE	Time
PEA	0.4253	00:00:01,10	0.3866	00:00:01,04	0.3076	00:00:01,02
EUC	0.2677	00:00:01,06	0.1951	00:00:01,07	0.2065	00:00:01,06
SPE	0.3837	00:00:01,07	0.3230	00:00:01,07	0.2619	00:00:01,05
TAN	0.2450	00:00:01,04	0.2227	00:00:01,09	0.1557	00:00:01,10
LOG	0.2421	00:00:01,06	0.2174	00:00:01,15	0.1695	00:00:01,13
GA_M	0.2202	12:19:55	0.2197	00:00:01,98	0.1839	00:00:01,97
GA	0.2202	12:19:55	0.1610	11:29:11	0.1557	13:23:55

Table 6
Results from GA_M and the CF-techniques in the $M1$ and $M2$ matrices varying k_t.

	PEA		EUC		SPE		TAN		LOG		GA_M	
K	*M*1	*M*2	*M*1	*M*2	*M*1	*M*2	*M*1	*M*2	*M*1	*M*2	*M*1	*M*2
0	0.3866	0.3076	**0.1951**	0.2065	0.3230	0.2619	0.2227	**0.1557**	0.2174	0.1695	0.2197	0.1839
1	0.4326	0.3659	**0.2076**	0.2110	0.2840	0.2570	0.2175	**0.1726**	0.2398	0.1769	0.2474	0.1773
2	0.4079	0.3740	**0.2003**	**0.1781**	0.3321	0.2477	0.2202	0.1837	0.2278	0.1935	0.2294	0.1896
3	0.3896	0.4057	0.2021	0.1864	0.3639	0.3150	0.2061	0.2010	0.2117	0.1897	**0.1980**	**0.1706**
4	0.4232	0.5113	0.2082	**0.1774**	0.3841	0.3618	0.2244	0.1871	0.2139	0.1763	**0.2050**	0.1782
5	0.5965	0.6212	0.2062	0.1666	0.4544	0.4844	0.2071	**0.1567**	0.2099	0.1708	**0.2033**	0.1657
6	0.7200	0.6648	0.2237	0.2203	0.5328	0.4610	0.2173	0.2239	0.2162	**0.2180**	**0.2145**	0.2304
7	0.8533	0.8545	0.2188	0.2248	0.6452	0.5731	0.2206	**0.2440**	0.2203	0.2562	**0.2047**	0.2544
8	1,0000	0.9025	0.2516	0.2860	0.8849	0.6769	0.2598	**0.2843**	0.2596	0.2969	**0.2440**	0.2859
9	1,0000	1,0000	0.3733	0.4757	1,0000	1,0000	**0.3606**	**0.4734**	**0.3606**	**0.4734**	0.3668	0.4847
MD	0.6209	0.6007	0.2286	0.2333	0.5204	0.4639	0.2356	0.2282	0.2377	0.2321	0.2332	0.2321

However, *EUC* (increase in RMSE of 12.28%), *SPE* (increase in RMSE of 42.41%) and *PEA* (increase in RMSE of 67.26%) achieve worse results than the GA_M.

Thus, the generalization brings an increase in the error, but the execution time of GA_M in $M1$ and in $M2$ decreased significantly compared to the execution in M. Thus, in a real scenario that prioritizes the computational cost, the generalist model (GA_M on $M1$ and GA_M in $M2$) can be adopted without encountering large increase of RMSE.

Given the results of the experiments carried out here, we come across a new hypothesis. The hypothesis that the individual obtained by running the GA, both in the matrices $M1$ and $M2$, is better than the individual obtained from the generalization model (GA_M). To confirm this hypothesis, further experiments were performed and the results can be seen in Table 5, where we established that:

- To $M1$:
  - GA_M is the generalized model (learned) in M and RMSE of 0.2197.
  - GA_{M1} is the specialist model, where the best individual (9 items of *Euclidean* technique and 1 item of *Tanimoto* technique) obtained RMSE of 0.1610.
- To $M2$:
  - GA_M is the generalized model (learned) in M and RMSE of 0.1839.
  - GA_{M2} is the specialist model, where the best individual (10 items of the *Tanimoto* technique) obtained RMSE of 0.1557.

Thus, the GA_{M1} was 26.74% better than GA_M in $M1$. The GA_{M2} was 15.34% better than GA_M in $M2$. Therefore, a real scenario that prioritizes the computational cost, the designer can choose the generalist model (GA_M), and this does not involve large increase in the RMSE. However, if the designer's priority is the quality (lower error RMSE) of the recommendation without care about the computational cost, the specialist model (GA_{M1} and GA_{M2}) are better.

5.3. Experiment three—The effect of variation of k_t *on the CF techniques and at* GA_M

In da˜Silva et al. (2014) it is demonstrated that the Spearman and Pearson techniques are sensitive to removal of reviews realized by the target-user in the matrix M. Thus, the purpose of this experiment is to verify whether the best individual (GA_M) is generalizable to the input matrices ($M1$ and $M2$) in scenarios where we vary the amount of target-user reviews (k_t). The creation of $M1$ and $M2$ follow the same criteria of the experiment performed in Section 5.2.

For all the selected techniques and to GA_M, we varied the value of k_t (second gene of the individual chromosome) that corresponds to the number of reviews that will be removed from the target-user.

We conduct experiments varying k_t from 0 to 9. In the first scenario we obtain chromosomes where the value of each k_t is defined as 0. In the second scenario, the values of each k_t is defined as 1, meaning that all the target-users in $M1$ and $M2$ will have 1 less review. We repeat this until all the k_t of each chromosome are defined as 9.

Thus, to compute the RMSE of each technique and GA_M, we replaced the M matrix of Fig. 5 by the $M1$ matrix, and the result can be observed in Table 6. We rerun the experiment replacing the M matrix of Fig. 5 by the $M2$ matrix and the result also can be observed in Table 6.

From Table 6 and comparing the scenario of $k = 0$ versus $k = 9$, all technical and GA_M presented an increase of RMSE in both matrices $M1$ and $M2$. In $M1$ Spearman had an increase of 67.70% of RMSE, Pearson was increased by 61.33%, the Euclidean distance by 47.74%, Tanimoto by 38.22%, GA_M by 40.10%, and finally Loglikelihood by 39.69%. This demonstrates that the Spearman and Pearson techniques were sensitive to the withdrawals reviews. This also demonstrates that Tanimoto, GA_M and Loglikelihood can keep RMSE low when the target-user has few reviews (in $M1$). In $M2$, Spearman had an increase of 73.81% for RMSE, Pearson was increased by 69.24%, Tanimoto in 67.11%, Loglikelihood by 64.19%, GA_M by 62.07% and the Euclidean distance by 56.59%. This shows that all the techniques and the GA_M worsened compared to $M1$. Only the Euclidean distance technique remained stable, while the other techniques and GA_M demonstrated sensitivity to ratings withdrawal of the target-user.

Table 6 shows that a learned individual (GA_M) from matrix (M) can be used in another matrix ($M1$ or $M2$) without a large increase in RMSE compared to CF-techniques adopted, even when there are few evaluations of the target user. The results demonstrate that the (GA_M) is better in 6 ($k = 3, 4, 5, 6, 7$ and 8) of the 10 created scenarios in $M1$. Looking at $M1$, the GA_M (0.2333) compared to the Euclidean distance technique (0.2287) had a higher average RMSE, but this value was not significant ($p = 0.84$). Still in $M1$, if we compare the average RMSE of GA_M with remaining techniques we observe that the GA_M get the lowest average RMSE. In $M2$, the GA_M outperformed the techniques only in the scenario where $k = 3$, but the result of GA_M on the other scenarios are satisfactory because it approached of the best technique for each scenario. Comparing the average of GA_M (0.2320) with all the other techniques, we see that the GA_M has the largest RMSE only in comparison to the Tanimoto technique (0.2282), but this value was not significant ($p = 0.93$). When we compare the GA_M with the other techniques, the GA_Mhas the lowest average RMSE. It is also possible to notice that the GA_M is better than the Pearson and Spearman techniques in any scenario, both for $M1$ and $M2$.

Thus, it is evident that generalization does not always reach the lowest RMSE. Therefore, it is recommended to rerun the AG and get an individual best adapted to each scenario from M.

5.4. Experiment four—The specialist and generalist models in the scenario of changing the amount of reviews

In previous experiments, the removal of k_t reviews are made only on the target user, simulating a user with less reviews compared to the others. But in this experiment, we aim to remove k_t reviews of all the users at the same time, simulating a gradual increase of the database through the time.

For conducting the experiments, the matrix used as input is still $M - M$ contains 283 users and each one of these users have 10 reviews, totaling 2830 evaluations. The matrices M_k are generated from M having as parameter the value of k_t (second gene of the individual chromosome), where k represents the amount of withdrawal of ratings from all the users of the database. For example, supposing that $k = 8$, this means that the new matrix M_{k8} will have the first two reviews of each user present in M, thus containing a total of 566 reviews. If $k = 7$, then the new matrix M_{k7} will have 849 reviews, and so on.

The flow for calculating the RMSE of each strategy, for a list (L_n) of recommendation with size of 10 is shown in Fig. 8.

Table 7 shows the result of the RMSE for each of five tested techniques along with the GAs behavior, where:

- *matriz* (M_{k8} until M_{k0}) are matrices created by varying $k_t = 8$ until $k_t = 0$, being used as input for the GA.
- GA_{Mk8} is the specialist in M_{k8} and generalist when applied in other scenarios. The specialist GA_{Mk8} consists of the combination of one item from the Euclidean technique, one item from Tanimoto and eight items from Loglikelihood.
- GA_{Mk5} is the specialist in M_{k5} and generalist when applied in other scenarios. The specialist GA_{Mk5} consists of the combination of eight items from the Euclidean technique, one item from Tanimoto and one item from Loglikelihood.
- GA_{Mk0} is the specialist in M_{k0} and generalist when applied in other scenarios. The specialist GA_{Mk0} consists of the combination of three items from the Tanimoto technique and seven items from Loglikelihood.
- GA_{MkE} is the specialist in each M_{k_t}.
  - M_{k8} and M_{k7} are composed by the combination of one item from the Euclidean technique, one item from Tanimoto and eight from Loglikelihood.
  - M_{k6} is composed by the combination of six items from the Euclidean technique, three items from Tanimoto and one from Loglikelihood.
  - M_{k5} is composed by the combination of eight items from the Euclidean technique, one item from Tanimoto and one from Loglikelihood.
  - M_{k4} is composed by 10 items from the Euclidean technique.
  - M_{k3} is composed by the combination of three items from the Tanimoto technique and seven from Loglikelihood.
  - M_{k2} is composed by 10 items from the Tanimoto technique.
  - M_{k1} is composed by the combination of one item from the Euclidean technique and nine from Loglikelihood.
  - M_{k0} is composed by the combination of three items from the Tanimoto technique and seven from Loglikelihood.

From Table 7 and comparing the scenario $Mk8$ *versus* $Mk0$, all the techniques and all the GAs present a reduction in RMSE with an increase of reviews in the matrix. The Pearson technique reduced its RMSE by 57.47%, Euclidean reduced by 60.71%, Spearman in 61.63%, Tanimoto by 63.99% and Loglikelihood by 64.41%. In this same scenario, the GA_{Mk0} showed a reduction in RMSE of 66.91%, the GA_{MkE} reduced in 65.71%, the GA_{Mk8} in 61.33% and the GA_{Mk5} in 61.02%. This shows that when increasing the number of ratings in the matrix both GA and techniques CF improve its performance, thereby reducing the prediction error in calculating the rating.

Table 7 demonstrates that the individual GA_{Mk8}, resulting from the execution of the GA in M_{k8}, continues with lower RMSE than the CF techniques when applied into a matrix M_{k7}. Note that the matrix M_{k7} has more information (about all users) than the matrix M_{k8}. The GA_{Mk8} applied to M_{k6} and to M_{k5} also obtained lower RMSE than the CF techniques. However the generalization of GA_{Mk8} is no longer advantageous in M_{k4}.

Analyzing the performance of GA_{Mk5} we perceive that it is useful in scenarios of M_{k5} (specialist), in M_{k6} (generalist), and in M_{k7} (generalist). Note that in scenarios where GA_{Mk5} was better, there was always a reduction in the amount of user ratings in the matrix. Thus, the GA_{Mk5} has a lower generalization than the GA_{Mk8} whether we compare scenarios of growth in the amount of user ratings in the matrix. In the case of GA_{Mk0}, its generalization always results in a reduction of the amount of user ratings in the matrix. When compared with CF techniques GA_{Mk0} is less sensitive of reduction on the number of reviews in M_k. The generalization of GA_{Mk0} outperformed them in M_{k0}, M_{k3}, M_{k5}, M_{k6}, M_{k7} and in M_{k8}.

Still in accordance with Table 7, it is noticeable that the mean RMSE of GA_{Mk8} is lower than the mean of all the CF-techniques, even the GA_{Mk8} having learned from less information about the target-user than the CF techniques. Comparing the mean of GA_{Mk8} with the lowest mean among the CF techniques (Loglikelihood), the reduction of RMSE corresponds to 4, 3%. If compared with the highest mean among the CF techniques (Pearson) the reduction is of 105.32%. The mean of GA_{Mk5} is also lower than the mean of all the CF techniques. Comparing the mean of GA_{Mk5} with the lowest mean among the CF techniques (Loglikelihood), the reduction of RMSE corresponds to 3.6% and if compared with the highest mean among the CF techniques (Pearson), the reduction is of 103.93%. The mean of GA_{Mk0} is also lower than the mean of all the CF techniques. Comparing the mean of GA_{Mk0} with the lowest mean among the CF techniques (Loglikelihood), the reduction of RMSE corresponds to 4.3% and if compared with the highest mean among the CF techniques (Pearson), the reduction is of 106.25%. So, still

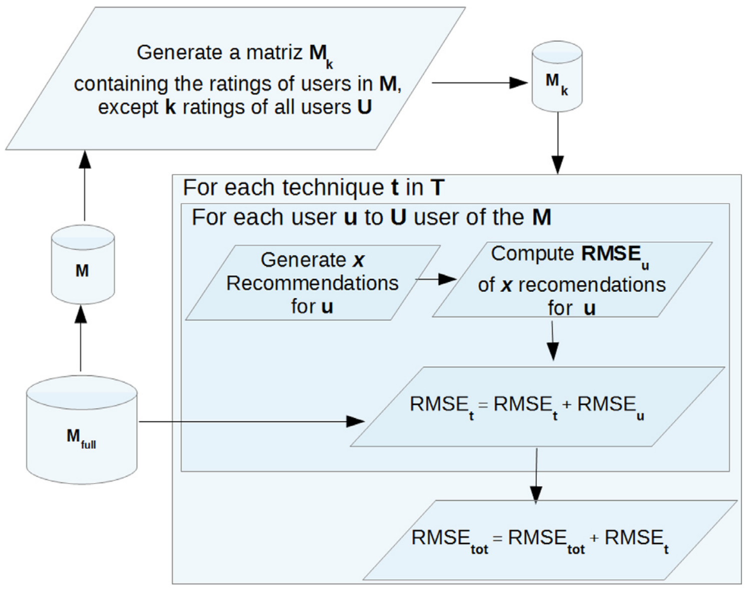
Generate a matriz M_k containing the ratings of users in M, except k ratings of all users U
M_k
M
M_full
For each technique t in T
For each user u to U user of the M
Generate x Recommendations for u
Compute RMSE_u of x recomendations for u
RMSE_t = RMSE_t + RMSE_u
RMSE_tot = RMSE_tot + RMSE_t

Fig. 8. Calculating the RMSE for each technique and the chromosome (candidate solution).

Table 7
Result of the learned GA from a matrix and applied in other matrices versus CF-techniques.

matriz	*PEA*	*EUC*	*SPE*	*TAN*	*LOG*	GA_{Mk8}	GA_{Mk5}	GA_{Mk0}	GA_{MkE}
M_{k8}	1.0000	0.6815	1.0000	0.6803	0.6803	**0.6423**	0.6528	0.6656	0.6423
M_{k7}	1.0000	0.3855	1.0000	0.3780	0.3780	**0.2856**	0.2878	0.3373	0.2856
M_{k6}	0.9438	0.3141	0.6909	0.3155	0.3135	0.2738	**0.2670**	0.2805	0.2652
M_{k5}	0.8618	0.3036	0.5608	0.2980	0.3019	0.2647	**0.2498**	0.2800	0.2498
M_{k4}	0.5854	**0.3038**	0.4756	0.3303	0.3279	0.3295	0.3211	0.3129	0.3038
M_{k3}	0.4777	0.3276	0.5103	0.3213	0.3074	0.3240	0.3199	**0.2895**	0.2895
M_{k2}	0.4722	0.3443	0.4099	**0.3351**	0.3358	0.3864	0.3843	0.3562	0.3351
M_{k1}	0.4481	0.2798	0.3943	**0.2663**	0.2691	0.2722	0.3105	0.2708	0.2464
M_{k0}	0.4253	0.2677	0.3837	0.2450	0.2421	0.2483	0.2544	**0.2202**	0.2202
MD	0.6905	0.3564	0.6028	0.3522	0.3507	0.3363	0.3386	0.3348	0.3153

observing the mean of RMSEs of the CF techniques and GAs, it is observed that the GA_{MkE} always reached the minimal RMSE value. The reduction of RMSE of GA_{MkE} compared to the lower mean RMSEs among the CF techniques (Loglikelihood) was of 11.23% and for the highest mean of RMSEs among the CF techniques (Pearson) was of 118.99%. The reduction of RMSE achieved by GA_{MkE} can be explained by the fact that it is always executed to each matrix M_k; instead it increases the computational cost of this procedure. Finally, comparing the mean RMSE between GA_{Mk0} (0.3348), GA_{Mk5} (0.3386) and GA_{Mk8} (0.3363), it is observed that the difference between them is small, which validates the hypothesis that the combination method is robust even applied in scenarios where there is some degree of generalization, for example, in scenarios of GA_{Mk8}. However, there is always the possibility of improving the RMSE whether the designer opts for a specialist strategy (GA_{MkE}).

Observing the column GA_{MkE} of Table 7, it is clear that the Specialist GA (execute every scenario), taking as input the current matrix (M_{k_t}), has the lowest RMSE among all technical and other GAs. It is validated the hypothesis that combining is better than choosing a technique isolated, even in scenarios where there is an increase in the amount of reviews about all users in the matrix.

Still according to the column GA_{MkE} of Table 7 it is possible to verify that by running the GA in M_{k0} the RMSE was 0.2202, making the use of 5% of M_{full}. And when running the GA in M_{k5} the RMSE was 0.2498, making the use of 2.5% of M_{full}. Thus, the difference of the RMSE between GA_{Mk0} and GA_{Mk5} increased 0.029, which we do not consider significant. But between GA_{Mk0} and GA_{Mk8} the incremen on RMSE (0.422) gives us an indication that there is a threshold of tolerance to use the generalization. It is worth noting that when there is a great reduction in the percentage of the matrix used for training, this difference tends to increase a lot.

Fig. 9 shows the performance of the GAs and the CF techniques. Note, in the chart *A*, that the GA_{Mk8} (specialist in M_{k8}) starts with a performance similar to GA_{MkE}, and as the database grows, it worsens in relation to the specialist. Thus, we can conclude that the generalization of GA_{Mk8} is no longer advantageous in *Mk*4. In the chart *B*, the GA_{Mk5} starts with a performance close to GA_{MkE} (in M_{k8}, M_{k7}, M_{k6}, M_{k5}) and after that its performance starts to deteriorate in relation to the specialist. In the chart *C*, the GA_{Mk0}, for even this being trained with plenty of reviews in the matrix, its RMSE is bad in M_{k8} and improves when approaching M_{k0}. Also it is noted from the chart *D*, that:

- The GA_{Mk8} decreases the RMSE as there are insertions of new reviews in the matrix.
- The GA_{Mk5} increases the RMSE in both cases, that is, increasing or decreasing the number of evaluations on the matrix.
- The GA_{Mk0} increases the RMSE as there is removal of reviews in the matrix.

Therefore, we conclude that the results achieved in other experiments keeps valid even when the removal of reviews is applied to all users of the matrix. In other words, we can conclude that the GA learned in a matrix M_{k8}, e.g. can be applied in future matrices (M_{k7}) but when the volume of data increases significantly it is prudent to rerun the GA and get an individual better adapted because the GA specialist always reaches a lower RMSE.

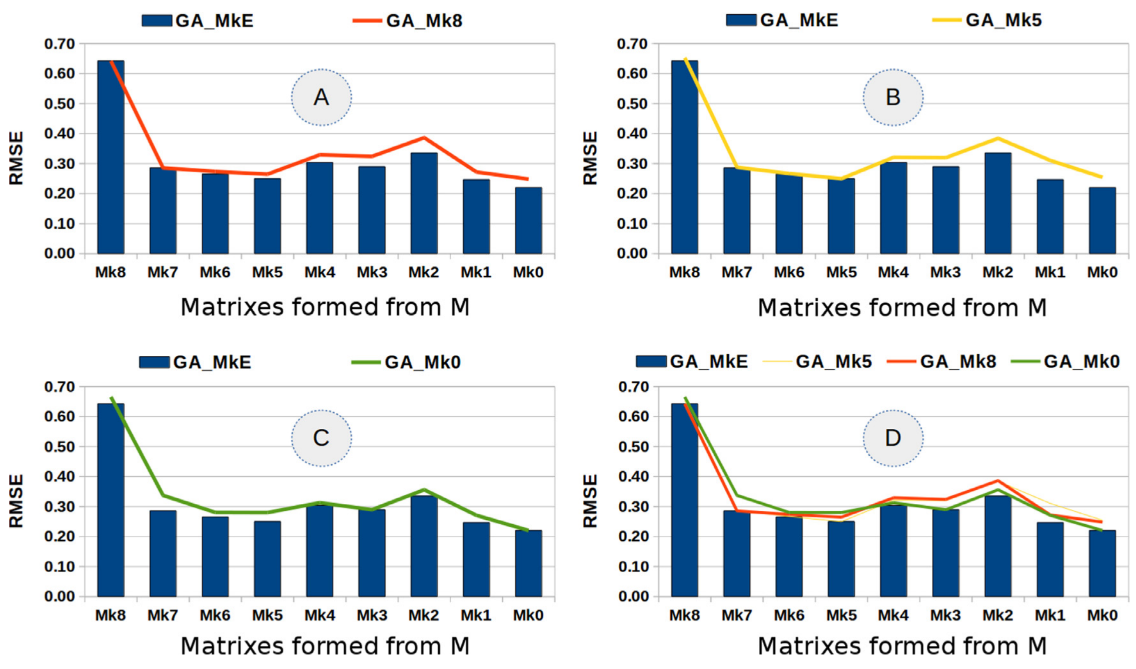

Fig. 9. RMSE performance of the GAs and CF techniques.

6. Conclusion and future work

This work had as a main objective the implementation of an evolutionary approach to combine results of RS-techniques. The genetic algorithm was chosen as a search algorithm and *CF-based* techniques were choose as examples of RS-techniques. Our insight was that an automated combination of techniques by a search algorithm could reach a great recommendation, and avoids the manual selection of the best technique for an application by designers.

Many experiments were run to evaluate the performance of recommendations for different scenarios, e.g. different states of database and different specialization levels of the model. The results showed that the combination of different techniques reduces the error (RMSE) compared to the basis approach alone. The specialist model got the best results in all experiments but had the highest computational costs. The generalist models got great results and low computational costs. The Loglikelihood and Tanimoto were the best between *CF-based* techniques in average.

Based on results, if one application do not change the database state frequently and thus do not require frequent updates in the model, e.g. every hour, it can use a specialist model, which brings great benefits because reaches very low values of RMSE (see Section 5.2). But this model obtained the high computational cost, thus an alternative of specialist model is the generalist one which brings great benefits because reduces considerably the computational cost and reaches solutions with satisfactory values of RMSE (see Section 5.2). The experiment 5.3 shows that the specialist model for a given matrix can be used successfully in another matrix (as generalist model), even when there are few ratings from the target-users. Finally, Section 5.4 showed that even when just a few number of rating are available for all users, the generalist model reached a satisfactory solution and specialist model is still the most effective.

Despite of the great performance of proposal, some limitation were identified. The method is totally dependent of the basis recommendation methods initially selected to combine. If none of them has good performance, the combination of them cannot reach good results either. Another limitation is the need of tuning the GA's parameter. For some applications the tuned GA could reach better results than a canonical one. Lastly, the proposal could not be applied in a very small database or initial-stage applications because is necessary a representative dataset, with items already ranked by the application's users, to evaluate the RMSE of each basis RS-technique.

The RS should improve the applicability in real-world problems (Lu et al., 2015). This work highlighted two ways to address this challenge, first by improving the performance of the techniques, and second by automation of the RS's design process. Thus, this work contributes with RS field with a proposal that uses search algorithm as a new automated way to optimize the combination of results from different techniques, which presented a great effectiveness. This initiative could inspire others hybrid approaches to be more automated and effective. Also, this method could be extended for many subfields of Expert and Intelligent Systems, since the proposal is a technique-independent approach and the results combination is a good way to obtain the strength of basis techniques without the complexity of match them as a new method.

Thus as future work, we suggest the use of different approaches to RS, such as content-based algorithms and hybrid approaches. Besides the recommendation techniques, the literature has several strategies from other areas of computer science that are used to improve the accuracy of the recommendations, e.g. web semantic and information retrieval, as described in Section 2. Thus, this proposal can also combine the results obtained from these strategies and RS-techniques to demonstrate the benefits of Invenire with diverse approaches and reduce the limitation of RS-techniques. Finally, we suggest to use parallel programming techniques in order to reduce the computational cost of GA or other evolutionary algorithms (Camilo-Junior & Yamanaka, 2007, 2011) in this method.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17*(6), 734–749. doi:10.1109/TKDE.2005.99.

Ahmad Wasfi, A. M. (1999). Collecting user access patterns for building user profiles and collaborative filtering. In *Proceedings of the 4th international conference on intelligent user interfaces (IUI'99)* (pp. 57–64). New York, NY, USA: ACM. doi:10.1145/291080.291091.

Al-Shamri, M. Y. H., & Bharadwaj, K. K. (2008). Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. *Expert Systems with Applications, 35*(3), 1386–1399.

Bobadilla, J., Hernando, A., Ortega, F., & Bernal, J. (2011a). A framework for collaborative filtering recommender systems. *Expert Systems with Applications, 38*(12), 14609–14623. doi:10.1016/j.eswa.2011.05.021.

Bobadilla, J., Ortega, F., Hernando, A., & Alcalá, J. (2011b). Improving collaborative filtering recommender system results and performance using genetic algorithms. *Knowledge-Based Systems, 24*(8), 1310–1316. http://dx.doi.org/10.1016/j.knosys.2011.06.005.

Bobadilla, J., Ortega, F., Hernando, A., & Gutierrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109–132. doi:10.1016/j.knosys.2013.03.012.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence (UAI'98)* (pp. 43–52). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL: http://dl.acm.org/citation.cfm?id=2074094.2074100.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331–370. doi:10.1023/A:1021240730564.

Burke, R. (2007). The adaptive web. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web chapter Hybrid Web Recommender Systems* (pp. 377–408). Berlin, Heidelberg: Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=1768197.1768211.

Cacheda, F., Carneiro, V., Fernandez, D., & Formoso, V. (2011a). Comparison of collaborative filtering algorithms. *ACM Transactions on the Web, 5*(1), 1–33. doi:10.1145/1921591.1921593.

Cacheda, F., Carneiro, V., Fernandez, D., & Formoso, V. (2011b). Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web, 5*(1), 2:1–2:33. doi:10.1145/1921591.1921593.

Camilo-Junior, C. G., & Yamanaka, K. (2007). Assisted recombination: Accelerating genetic improvements of populations in a ga. In *Proceedings of 10th international conference on intelligent systems and control, Cambridge MA, USA* (pp. 371–376).

Camilo-Junior, C. G., & Yamanaka, K. (2011). In vitro fertilization genetic algorithm, evolutionary algorithms. In E. Kita (Ed.), *Evolutionary algorithms* (pp. 57–68). Rijeka, Croatia: InTech. doi:10.5772/16074.

Candillier, L., Meyer, F., & Boullé, M. (2007). Comparing state-of-the-art collaborative filtering systems. In P. Perner (Ed.), *Machine learning and data mining in pattern recognition*. In *Lecture Notes in Computer Science: vol. 4571* (pp. 548–562). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-73499-4_41.

Colombo-Mendoza, L. O., Valencia-García, R., Rodríguez-González, A., Alor-Hernández, G., & Samper-Zapater, J. J. (2015). RecomMetz: A context-aware knowledge-based mobile recommender system for movie showtimes. *Expert Systems with Applications, 42*(3), 1202–1222. URL: http://www.sciencedirect.com/science/article/pii/S0957417414005557.

da Silva, E. Q., Camilo, C. G., Pascoal, L. M. L., & Rosa, T. C. (2014). An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 959–966). IEEEXplore. doi:10.1109/CEC.2014.6900631.

Dellarocas, C. (2000). Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the 2nd ACM conference on electronic commerce* (pp. 150–157). ACM. URL: http://dl.acm.org/citation.cfm?id=352889.

Engelbrecht, A. P. (2007). *Computational intelligence—An introduction* (2nd ed.). Andries P. Engelbrecht.

Fong, S., Ho, Y., & Hang, Y. (2008). Using genetic algorithm for hybrid modes of collaborative filtering in online recommenders. In *Proceedings of the eighth international conference on hybrid intelligent systems* (pp. 174–179). IEEE Computer Society. doi:10.1109/HIS.2008.59.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*, 61–70. http://doi.acm.org/10.1145/138859.138867.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*(2), 95–99.

Herlocker, J. L. (2000). *Understanding and improving automated collaborative filtering systems*. University of Minnesota Ph.D. thesis. AAI9983577

Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 230–237). Berkeley, California, USA: ACM. doi:10.1145/312624.312682.

Ho, Y., Fong, S., & Yan, Z. (2007). A hybrid ga-based collaborative filtering model for online recommenders. . In J. Filipe, D. A. Marca, B. Shishkov, & M. van Sinderen (Eds.), *Ice-b* (pp. 200–203). INSTICC Press. URL: http://dblp.uni-trier.de/db/conf/icete/ice-b2007.html#HoFY07.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems, 22*(1), 89–115. doi:10.1145/963770.963774.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Holland, J.H.

Hu, R., & Pu, P. (2010). Using personality information in collaborative filtering for new users. *Recommender systems and the social web*, 17–24. URL: http://www.dcs.warwick.ac.uk/~ssanand/RSWeb_files/Proceedings_RSWEB-10.pdf.

Hwang, C.-S., Su, Y.-C., & Tseng, K.-C. (2010). Using genetic algorithms for personalized recommendation. In *Proceedings of the second international conference on computational collective intelligence: Technologies and applications (vol. Part II) (ICCCI'10)* (pp. 104–112). Springer-Verlag.

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on information and knowledge management (CIKM'01)* (pp. 247–254). New York, NY, USA: ACM. doi:10.1145/502585.502627.

Lampropoulos, A. S., Sotiropoulos, D. N., & Tsihrintzis, G. A. (2012). Evaluation of a cascade hybrid recommendation as a combination of one-class classification and collaborative filtering. In *IEEE 24th international conference on tools with artificial intelligence, Athens, Greece, November 7–9, 2012* (pp. 674–681).

Liu, L., Koutrika, G., & Wu, S. (2015). LearningAssistant: A novel learning resource recommendation system. In *Proceedings of the IEEE 31st international conference on data engineering (ICDE)* (pp. 1424–1427). IEEEXplore. doi:10.1109/ICDE.2015.7113392.

Lones, M. A. (2011). Sean luke: Essentials of metaheuristics.. *Genetic Programming and Evolvable Machines, 12*(3), 333–334. URL: http://dblp.uni-trier.de/db/journals/gpem/gpem12.html#Lones11.

Lopes, S., Silveira, P., Silveira, G., Stepaat, N., & Kung, F. (2012). *Introdução à Arquitetura e Design de Software*. São Paulo: Elsevier Editora Ltda. http://www.sciencedirect.com/science/book/9788535250299.

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems, 74*, 12–32. http://dx.doi.org/10.1016/j.dss.2015.03.008.

Luo, X., Xia, Y., & Zhu, Q. (2012). Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems, 27*, 271–280. doi:10.1016/j.knosys.2011.09.006.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press.

Marmanis, H., & Babenko, D. (2009). *Algorithms of the intelligent web* (1st ed.). Greenwich, CT, USA: Manning Publications Co..

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, MA, USA: MIT Press.

Mobasher, B., Jin, X., & Zhou, Y. (2004). Semantically enhanced collaborative filtering on the web. In B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, & G. Stumme (Eds.), *Web mining: From web to semantic web*. In *Lecture Notes in Computer Science: vol. 3209* (pp. 57–76). Berlin Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-30123-3_4.

Owen, S., Anil, R., Dunning, T., & Friedman, E. (2011). *Mahout in action*. Greenwich, CT, USA: Manning Publications Co..

Park, M.-H., Hong, J.-H., & Cho, S.-B. (2007). Location-based recommendation system using bayesian user's preference model in mobile devices. In *Proceedings of the 4th international conference on ubiquitous intelligence and computing (UIC'07)* (pp. 1130–1139). Berlin, Heidelberg: Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=2391319.2391438.

Pascoal, L. M. L., Camilo, C. G., da Silva, E. Q., & Rosa, T. C. (2014). A social-evolutionary approach to compose a similarity function used on event recommendation. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1512–1519). IEEEXplore. doi:10.1109/CEC.2014.6900495.

Qingshui, L., & Meiyu, Z. (2010). The study of personalized recommendation technology based content and project collaborative filtering combines. In *Proceedings of the 3rd international conference on advanced computer theory and engineering (ICACTE): vol. 5* (pp. 506–510). IEEEXplore. doi:10.1109/ICACTE.2010.5579480.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on computer supported cooperative work (CSCW'94)* (pp. 175–186). New York, NY, USA: ACM.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM, 40*, 56–58. http://doi.acm.org/10.1145/245108.245121.

Roh, T. H., Oh, K. J., & Han, I. (2003). The collaborative filtering recommendation based on SOM cluster-indexing CBR. *Expert Systems with Applications, 25*(3), 413–423. doi:10.1016/s0957-4174(03)00067-8.

Salehi, M., Pourzaferani, M., & Razavi, S. A. (2013). Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model. *Egyptian Informatics Journal, 14*(1), 67–78. http://dx.doi.org/10.1016/j.eij.2012.12.001.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on electronic commerce (EC'00)* (pp. 158–167). New York, NY, USA: ACM.

Schafer, J., Konstan, J., & Riedl, J. (2001). E-commerce recommendation applications. *Data Mining and Knowledge Discovery, 5*, 115–153. URL: http://link.springer.com/chapter/10.1007/978-1-4615-1627-9_6.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 210–217). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. URL: http://dl.acm.org/citation.cfm?id=223931.

Smyth, B., & Cotter, P. (2000). A personalized tv listings service for the digital tv age. *Knowledge-Based Systems, 13*(2-3), 53–59.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, 2009*, 1–19. doi:10.1155/2009/421425.

Symeonidis, P., Nanopoulos, A., & Manolopoulos, Y. (2009). Moviexplain: A recommender system with explanations. In *Proceedings of the third ACM conference on recommender systems (RecSys'09)* (pp. 317–320). New York, NY, USA: ACM. doi:10.1145/1639714.1639777.

Yager, R. R. (2003). Fuzzy logic methods in recommender systems. *Fuzzy Sets Systems, 136*(2), 133–149. doi:10.1016/S0165-0114(02)00223-3.

Zhong, J., & Li, X. (2010). Unified collaborative filtering model based on combination of latent features.. *Expert Systems with Applications, 37*(8), 5666–5672. URL: http://dblp.uni-trier.de/db/journals/eswa/eswa37.html#ZhongL10.