The Generalist Recommender System GRSK and Its Extension to Groups

Conference Paper *in* Lecture Notes in Business Information Processing · April 2010

DOI: 10.1007/978-3-642-22810-0_16 · Source: DBLP

CITATIONS
3

READS
106

4 authors, including:

Inmaculada García
Universitat Politècnica de València
33 PUBLICATIONS **497** CITATIONS
SEE PROFILE

Laura Sebastia
Universitat Politècnica de València
62 PUBLICATIONS **875** CITATIONS
SEE PROFILE

Eva Onaindia
Universitat Politècnica de València
165 PUBLICATIONS **1,330** CITATIONS
SEE PROFILE

The Generalist Recommender System GRSK and Its Extension to Groups

Inma Garcia, Laura Sebastia, Sergio Pajares, and Eva Onaindia

Dpt. Computer Science, Technical Univ. of Valencia
Camino de Vera s/n, 46022, Valencia, Spain
{ingarcia,lstarin,spajares,onaindia}@dsic.upv.es

Abstract. This paper presents a Generalist Recommender System Kernel (GRSK) and describes the differences of the recommendation process when it is applied to groups. The GRSK is able to work with any domain as long as the domain description is represented within an ontology. Several basic techniques like demographics, content-based or collaborative are used to elicit the recommendations, as well as other hybrid techniques. The GRSK provides a configuration process through which to select the techniques and parameters that best suit the particular application domain. The experiments will show the success of the GRSK in different domains. We also outline the changes and new techniques required by the GRSK when it is used in a group recommendation.

Keywords: Recommender systems, Group recommenders, Tourism, Movies.

1 Introduction

Every day, new data appears on the Web. Everyone browsing on the Internet can have the perception of the huge amount of information available, which can lead to a situation of information overload, that is the situation where there is far too much information at people disposal so that useful information could be hidden by other data. In this case, techniques to retrieve *useful* information become more and more important. The usefulness of information depends on the users and their objectives, so retrieval systems have to try to understand the purpose of a user search in order to propose information he could be interested in. A special kind of information retrieval techniques that focuses on this issue is named *information filtering*. As the name suggests, starting from a big set of information, this technique identifies a small subset which should include the useful/interesting information.

Recommendation systems are a specific type of information filtering technique that attempts to present information items (e.g. movies, songs, activities, etc.) that are likely of interest to the user. A **recommender system** (*RS*) [13] is personalization tool that attempts to provide people with lists of information items that best fit their individual tastes. A RS infers the user's preferences by analyzing the available user data, information about other users and information about the environment.

RS are used to either predict whether a given user will like a particular item or identify the top N items that will be of interest to the user. In RS, how much a particular user likes an item is represented by a *rating*. Basically, a RS estimates ratings for the

J. Filipe and J. Cordeiro (Eds.): WEBIST 2010, LNBIP 75, pp. 215–229, 2011.

items that have not been seen by a user and recommends to the user the items with the highest estimated ratings.

Being an instance of information filtering, recommendation systems can be based on the demographic filtering algorithm, the content-based filtering algorithm or the collaborative filtering algorithm[1]. All these approaches have advantages and disadvantages [1]; a common solution adopted by many RS is to combine these techniques into an hybrid RS [11,4] thus improving recommendations by alleviating the limitations of one technique with the advantages of others.

Recently, some researchers have been focusing on enhancing recommendations by exploiting a semantic description of the domain in which recommendations are provided [16],[15]. In general, items handled by the system are semantically described through an ontology. Then, the recommendations are based on the semantic matching between the user profiles and the item descriptions. The main disadvantage of these approaches is that a semantic representation of the domain has to be available and, up to now, user profiles and items are described manually.

This paper summarizes the main characteristics of the *Generalist Recommender System Kernel (GRSK)*. It is a RS based in a semantic description of the domain that uses a hybrid recommendation technique, fed by the recommendations obtained from different algorithms. The task of GRSK is to generate the list of the top N items that will be of interest to the user. GRSK can be parameterized to adjust the system working model, i.e. to use the desired recommendation techniques. Besides, it is prepared to include as many techniques as desired by simply developing new modules. On the other hand, it is a domain-independent engine, able to work with different catalogs of items to recommend.

This paper is organized as follows. Section 2 gives an overview on the GRSK architecture, the information GRSK needs (ontology and user information) and, finally, the GRSK recommendation process. Section 3 explains the process of GRSK configuration to be integrated into a system. Section 4 presents the results we have obtained when working with a tourism domain and with a movies domain. We finish with some conclusions and future work.

2 GRSK: Generalist Recommender System Kernel

2.1 GRSK Ontology

The GRSK behaviour relies on the use of a **ontology** to describe the user's preferences and the items to recommend. It has been designed to be *generalist*, so GRSK is able to work with any application domain as long as the data of the new domain can be defined through an ontology representation.

An ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. The GRSK ontology contains the **features** that describe the items in the domain. For example, in the tourism domain, the ontology is composed of terms describing architectonic styles or types of buildings. Figure 1 shows an example of this ontology. In the movies domain (figure 2), the feature denote the

[1] These algorithms will be detailed later on.

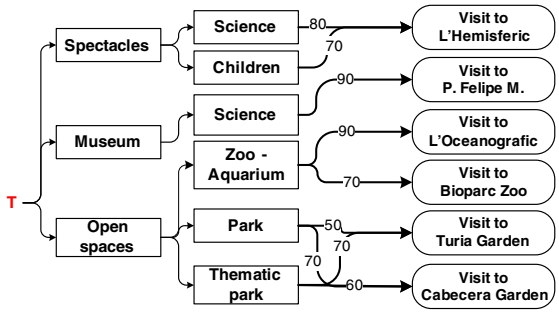

Fig. 1. Part of the *e-Tourism* ontology

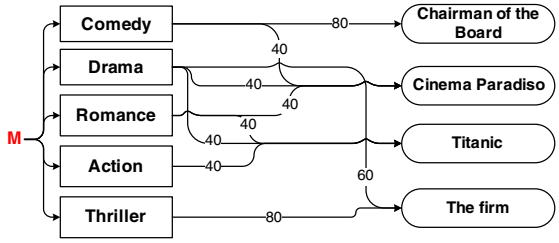

Fig. 2. Part of the *e-Movies* ontology

film genres. It is important to remark that GRSK is able to work from simple ontologies (such as the movies ontology, which is basically a list of genres) to more complex ontologies (with several levels of refinements, for example).

The **items** in the domain are described by the features of the ontology. Moreover, each pair item-feature is associated a value to indicate the **degree of interest** of the item under the feature, i.e. as a member of the category denoted by the feature. An item can also be categorized by more than one feature in the ontology. Formally, an item i is described by means of a list of tuples of the form (i, f, d^{if}), where f is a feature defined in the ontology and $d^{if} \in [0, 100]$ is the degree of interest of the item i under the feature f. Additionally, items are associated a numeric value AC^i (**acceptance counter**) to represent how popular the item i is among users; this value indicates how many times this item has been accepted when recommended.

2.2 User Information

In order to compute a recommendation, GRSK records a profile of each user, which models the user tastes and preferences as well as his historical interaction with the system. The **profile** of a given user u records, in first place, personal and demographic details about the user like the age, the gender, the family or the country. Second, the user profile also contains the user general-likes model, denoted by GL^u, which is a list of the features f in the ontology the user is interested in along with the user ratings r^{uf} for those features: $GL^u = \{(u, f, r^{uf})\}$, where $r^{uf} \in [0, 100]$. A user profile in GRSK also contains information about the historical interaction of the user with the RS,

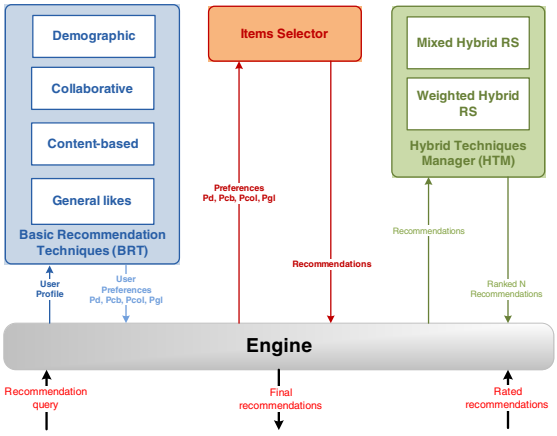

Fig. 3. GRSK Architecture

namely the set of items i the user has been recommended and his degree of satisfaction r^{ui} with the recommended items: $RT^u = \{(u, i, r^{ui}\}$, where $r^{ui} \in [0, 100]$.

2.3 The GRSK Architecture

Figure 3 shows an sketch of the GRSK architecture. The **Engine** module is the core of GRSK. The first task of the Engine is to capture and store the user profile when the user logs in the system for the first time. Then, the information obtained during the interaction of the user with the system *after* the recommendation (*rated recommendations*) will be used to update his profile to better capture his preferences.

The Engine is also in charge of controlling the recommendation process, which consists of two steps: first, each basic recommendation technique calculates a set of preferences for the user profile; and then, the items selector obtains the items that match the user preferences which are combined by the hybrid technique to obtain the final list of recommended items. The modules used by the engine to obtain the recommendation are:

- **Basic Recommendation Techniques** (BRT) [4] (demographic RS, content-based RS, collaborative RS and general likes-based filtering) are used to obtain the *user preferences* by analyzing his own profile, the profiles of other users and the items selected by the users that have utilized the system before. For a given user, each BRT creates a different list of preferences according to the parameters and data handled by the technique. The system configuration allows to select the set of BRT to use in the recommendation process (see section 3).
- **Items Selector:** receives the lists of *user preferences* and, for each list, it returns the set of items that better match the elements in the list.
- The **Hybrid Techniques Manager** (HTM) combines the lists of items in a single list, that conform the final user recommendation list. The hybrid techniques are applied on items, not on preferences. At this moment, GRSK includes two hybrid recommendation methods: the mixed hybrid technique and the weighted hybrid

technique. The system configuration allows to select *only one* hybrid technique to use in the recommendation process.

At this moment, GRSK includes several BRT and two hybrid techniques, but it is prepared to work with as many techniques as desired by simply developing new modules. We opted for these techniques because we considered them more suitable for the most common domains.

2.4 GRSK Recommendation Process

The recommendation process in GRSK is divided in two steps. The first one is to obtain the preferences that define the items that will be of interest to the user (section 2.4). The user introduces his query, which is sent joint with his profile to the BRT to produce a list of individual preferences for each technique. The second step is to obtain the list of items to recommend (section 2.4). This second step includes to obtain the list of items that match the preferences and to apply an hybrid recommendation technique to obtain the final ranked list of recommended items.

Modeling of User Preferences. This step consists of analyzing the user profile and eliciting the corresponding list of preferences. It is important to note that, unlike most RS, GRSK is a semantic RS that does not initially work with the items that will be later recommended to the user. In contrast, GRSK makes use of the concept of feature to elicit the user preference model, which is a more general and flexible entity. This makes GRSK able to work with any application domain as long as the data can be represented through an ontology.

A *preference* (which is a tuple of the form (u, f, d^{uf})) is a feature f in the ontology with a interest-degree of d^{uf} for a user u, selected by one of the four basic recommendation techniques: demographic recommendation, content-based recommendation, collaborative recommendation and general likes-based filtering. Each BRT generates a different set of preferences, an independent list of preferences and hence the lists may contain different features or the same feature with different degrees of interest. We will call these lists P_d^u for the demographic preference list, P_{cb}^u for the content-based preference list, P_{col}^u for the collaborative preference list, and P_{gl}^u for the general-likes-based preference list.

The **demographic BRT** classifies the user into a demographic category according to his profile details. Each demographic category is associated a list of preferences (P_d^u) during the system configuration because they depend on the application domain. The success of the demographic recommendation is strongly dependant of this user classification. We opted for a demographic BRT because it is a good alternative to solve the problem of the *new user* since it is able to always give a recommendation.

The **content-based RS technique** computes a set of preferences by taking into account the items previously rated by the user (historical interaction). This technique will allow us to increase the user satisfaction by recommending items similar to those already accepted by the user. Let f be a feature and I a list of items described under the feature f in the ontology; I will be a list of tuples of the form (i, f, d^{if}) for a particular feature f. Let $RT^u = \{(u, i, r^{ui})\}$ be the set of items valued by user u with respective ratings of r^{ui}; a preference (u, f, d^{uf}) is added to the list P_{cb}^u where:

$$d^{uf} = \frac{\sum_{\forall i \in I \cap RT^u} d^{if} * r^{ui}}{|RT^u|}$$

The value d^{uf} denotes the interest-degree of a user for the items described under the feature f amongst the whole set of items rated by the user.

The **collaborative RS technique** suggests those items preferred by people with a profile most similar to the given user profile (i.e. the user will be recommended items that people with similar tastes and preferences liked in the past). This technique is only useful when there is a great amount of data concerning items rated by other users. In order to obtain the corresponding list of preferences P^u_{col}, this technique decides whether a user v is similar to the given user u ($s_{u,v}$) by applying the Pearson Correlation with respect to the items that have been rated by both users. Then, by taking into account all the users v similar to u, a preference (u, f, d^{uf}) is added to P^u_{col} for each f that describes an item i rated by v, where:

$$d^{uf} = avg(d^{if} * r^{vi}), \forall v : s_{u,v}$$

The **general-likes-based filtering** is an information filtering technique that obtains the preferences that match with the main user interests specified by the user in his profile (GL^u). The accuracy of this technique depends on the information provided by the user. However, GRSK is able to work with few information. In this case, the set of preferences P^u_{gl} is simply built as $P^u_{gl} = GL^u$; that is, the interest-degree of the preferences in P^u_{gl} will be the ratings given by the user to that particular feature in his profile ($d^f = r^f$).

Obtention of the List of Recommended Items. In the second step of the recommendation process, the **Items Selector** selects, among all of the items in the domain, those that best match the preferences in the lists P^u_d, P^u_{cb}, P^u_{col} and P^u_{gl}. Afterwards, the selected **Hybrid Technique** obtains a single list of ranked recommendations that we will denote as $RI^u = \{(u, i, d^{ui})\}$, where i is the item, and d^{ui} is the estimated interest-degree of the item i for the user u.

The method for selecting an item is quite simple: an item i represented by the tuple (i, f, d^{if}) matches a preference in P^u_{brt} if there is a tuple (u, f, d^{uf}) in P^u_{brt} such that the item has not previously rated by the user. The outcome of the Items Selector is a set of lists of ranked items, one list per BRT. The lists of recommended items computed by the Items Selector are then processed by the selected Hybrid Technique and returns a single list of ranked items (RI^u). The value d^{ui} of a tuple in RI^u depends on the selected Hybrid Technique. At this moment, GRSK includes two hybrid techniques: mixed and weighted techniques.

The **Mixed Hybrid Technique** mixes the items in the lists of all the BRT. All items are handled in the same way with independency the BRT they belong to. In this case, the value d^{ui} of a tuple in RI^u is calculated as follows:

$$d^{ui} = percentile(AC^i) + avg_{\forall f}(d^{if} + d^{uf})$$

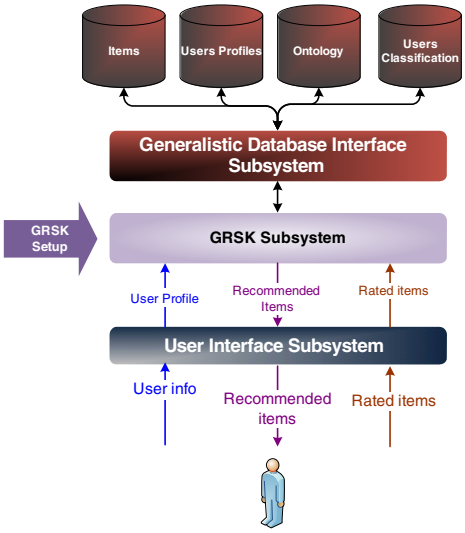

Fig. 4. GRSK Integration into a System

where $percentile(AC^i)$ refers to the percentile of the acceptance counter of i (AC^i) with respect to the whole set of items rated by the users. The second part of the formula considers the average interest-degree of all the features that describe the item i in both the ontology (d^{if}) and in the user preferences (d^{uf}).

The **Weighted Hybrid Technique** mixes the items in the lists, but the value d^{ui} is computed according to the weight of the BRT that selected the preference for which the item has been recommended. The weight of each BRT, defined in the configuration process, is denoted by ω_d for the demographic RS, ω_{cb} for the content-based RS, ω_{col} for the collaborative RS and ω_{gl} for the general-likes filtering. In this case the value d^{ui} of a tuple in RI^u is calculated as:

$$d^{ui} = percentile(AC^i) + avg_{\forall f}((d^{if} + d^{uf}) \times \omega_{brt})$$

The Hybrid Technique obtains a list of ranked items and retrieves the best N ranked elements.

3 GRSK Integration Process

This section describes the GRSK requirements to be integrated into any system (figure 4).

3.1 Database and External Subsystems

In first place, GRSK needs a **database** of the particular application domain containing: (1) *the domain ontology*; (2) *the set of items that can be recommended:* these items must be classified according to the ontology and the quality of the recommendation depends,

in part, on the accuracy of the classification of items; (3) *the user profiles with the demographic user information* (if the demographic RS is used in GRSK) *and the user general likes* GL^u (if the general-likes filtering is used): the quality of the recommendation also depends on the information provided by the user - the more information, the more accurate the recommendation -, but it is possible to obtain a recommendation with a minimum amount of data; (4) *demographic classification of users* according to the ontology.

In order to obtain a complete recommender system, two external modules must be plugged to GRSK: the Database Interface and the User Interface. The **Database Interface subsystem**, which is the interface between GRSK and the database, processes the queries coming from GRSK, such as obtaining the user profile of the current user or the list of items that match a given preference. On the other hand, the **User Interface Subsystem** initiates the execution of GRSK and centralizes the exchange of information between the user and GRSK. This includes converting the user data into a user profile, showing the list of recommended items and recording the ones that are selected and discarded by the user joint with the rating of the user satisfaction with a given recommended item. The User Interface Subsystem is also in charge of deciding which information must be initially introduced by the user (which depends on the particular application domain).

3.2 GRSK Setup

GRSK requires an initial configuration to adjust the GRSK behaviour to the current application. First, it is possible to select *which BRT* among all the available BRT (demographic RS, content-based RS, collaborative RS and general likes-based filtering), will be used in GRSK to give a recommendation. Second, it is necessary to select only *one hybrid recommendation technique*. Moreover, for all hybrid techniques, it is possible to select the way to compute the interest-degree of items in case an item is selected by more than one preference. The techniques are: maximum ratio, median ratio and several techniques to compute the average.

On the other hand, some other computations can be parameterized. For example, a threshold of the interest-degree can be defined to consider or not a given preference. Or the acceptance counter can be computed in several ways.

4 Case Studies

This subsection discusses the experiments conducted to evaluate the behavior of GRSK.

Two domains have been used to evaluate GRSK: a tourism domain and a movies domain. Through these case studies, we will show that GRSK has been successfully used in both cases.

In order to test GRSK, we selected two classical Information Retrieval metrics: *precision* and *recall*. In an Information Retrieval scenario, precision is defined as the number of retrieved relevant items divided by the total number of items retrieved by that search; and recall is defined as the number of retrieved relevant items divided by the total number of existing relevant items. That is, precision represents the probability that

a retrieved item is relevant to the user and recall is the probability that a relevant item is retrieved by the search.

Specifically, we call Ns the number of retrieved items by GRSK, that is, the number of recommendations solicited by the user. The number of relevant items is denoted by Nr and Nrs is the number of relevant items retrieved in the recommendation, that is, $Nrs = Nr \cap Ns$. Then, precision and recall are calculated as follows:

$$P = \frac{Nrs}{Ns} \qquad R = \frac{Nrs}{Nr}$$

Often, there is an inverse relationship between P and R, where it is possible to increase one at the cost of reducing the other. For example, R can be increased by increasing Ns, at the cost of increasing the number of irrelevant items retrieved (decreasing P). For this reason, P and R ratios are not discussed in isolation.

We run our experiments in terms of two parameters, Ns the number of retrieved items, and the information about past visits in the user profile. As for Ns, we run tests with $Ns = 10$ and $Ns = 25$. In both experiments, we obtained the same list of retrieved items, but in the first case, the system considered the first 10 items and, in the second case, the first 25 items were considered. Regarding the second parameter, we took into account four levels of historical information in the user profile; a new user and user profiles that store 25%, 50% and 75% of (randomly selected) rated items, respectively.

4.1 E-Tourism: A Touristic Recommender System

e-Tourism is a web-based recommender system that computes a user-adapted leisure and tourist plan for a given user. The system does not solve the problem of traveling to an specific place but it works on recommending a list of the activities that a tourist can perform in the city of Valencia (Spain). It also computes a time schedule for the list of recommended activities taking into account the distance between places, the opening hours, etc. - that is, an agenda of activities [14]. It is intended to be a service for foreigners and locals to become deeply familiar with the city and plan leisure activities.

Data Warehouse E-Tourism. As this is a new domain, a survey was filled by 58 people in order to obtain data for testing the system. Personal data like name, age, marital status and tourist profile (cultural, business, family, etc.) were collected. They also identified sites already visited along with a degree of interest (rating) for each site. There are 115 preferences structured in the ontology (see figure 1), 141 sites stored and 58 user profiles. Each user rated (positively or negatively) all sites (RT^u) and an average of 110 preferences (GL^u).

E-Tourism Experimental Results. When performing the experimental results in this domain, we divided the user profiles database into two sets: 48 users were the training users and 10 users were the test users. Then, as all users rated all sites, we considered as relevant items (Nr) those visits that the test users marked as visited with a positive degree of satisfaction in the survey.

Figure 5 shows a comparison between the average of precision (P) and recall (R) for all the different cases of user feedback. When $Ns = 10$, the difference between the precision and the recall is remarkable, and the precision decreases as the recall increases, as

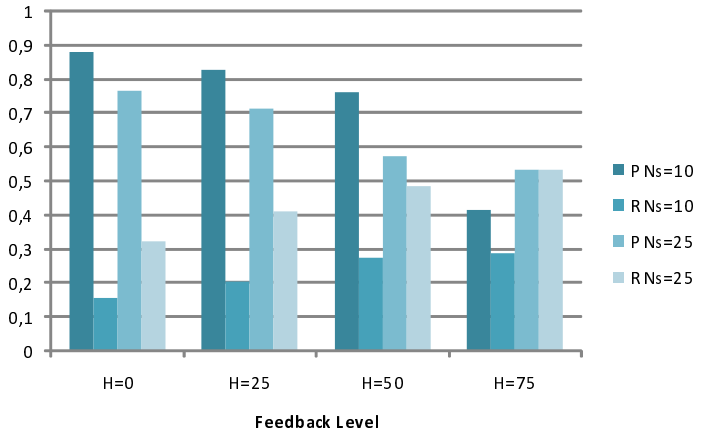

Fig. 5. Comparison of the P and R values obtained when Ns=10 and Ns=25 and for the four degrees of historical information for the tourism domain

expected. However, when $Ns = 25$, this difference is not so noticeable. When $Ns = 10$ and the information provided to the system increases ($H = 25$, $H = 50$), GRSK improves the quality of the recommendations if we consider P and R together. However, in some of the cases in which the user feedback is rather high ($H = 75$), the quality of the recommendation worsens. This is because the database does not contain a large number of items and, therefore, GRSK is not able to recommend places other than those ones already visited by the user. When $Ns = 25$, the general impression is similar. However, in this case, the relation $P - R$ is better because, although the precision is a bit lower, the recall increases in a higher order. Here again, the more feedback, the better the quality of the recommendation, and, unlike the previous case, the worsening in the case of $H = 75$ is not as noticeable.

4.2 E-Movies: A Movies Recommender System

e-Movies is a application-based recommender system that computes user tastes regarding preferences movies for a given user, in order to obtain the best list of movies for the user. It is intended to be a service for any cinephile, working with a multitude of movies.

Data Warehouse E-Movies. In this case, we selected a well-known movies database, MoviLens[2], which has been created by the GroupLens research group at the University of Minnesota. It contains 900 user profiles with their respective histories of interaction with the system and a set of 1682 films. A user has scored between 20 and 700 movies. Each film is described by a title, number of people who were recommended the film and watched it, the year was recorded, etc. All the films have been cataloged through an ontology of twenty preferences (see figure 2). Each user has an average of 15 preferences associated with several ratios (GL^u) and has rated an average of 45 movies

[2] http://www.grouplens.org/

(RT^u). Moreover, each movie has been rated by 57 different users in average and has been described by 2 preferences in average.

E-Movies Experimental Results. When performing the experimental results in this domain, we divided the user profiles database into two sets: 890 users were the training users and 10 users were the test users. We considered as relevant items (Nr) those movies that the test users have marked with a value between 2 and 5.

Figure 6 shows a comparison between the average of precision (P) and recall (R) for all the different cases of user feedback. In all cases, the difference between the precision and the recall is quite remarkable. The reason behind is that the number of relevant items (Nr) is quite high compared to the number of retrieved items as each test user has rated up to 685 movies and has an average of 551 movies. On the other hand, we expected (as in the tourism domain) that both measures (considered together) increased as the user history also increased (except when H=75, as explained above). However, figure 6 shows that when H = 50 the precision decreases slightly. The reason is the following. The precision P is calculated by taking into account a user history. Remember that the possibility that a retrieved relevant item in Nrs was not included in this Nr is not considered in P, therefore it must be satisfied that $Nrs \subseteq Nr$. Thereby the more feedback level, the lower Nr is observed, being Ns constant. This is the reason why P with 25% is a little bit better than P with 50%, because it is easier to find a retrieved relevant item within the 75% user history (25% feedback) than with 50%. If we could ask the user about his satisfaction with respect to a given recommendation, we would have a better picture of the GRSK performance in this domain. This does not happen in the tourism domain because we have a complete feedback for all users. We also have the intuition that a more complex ontology and a more complete description of items (such as in the tourism domain) improves the quality of the recommendations. However, we need to perform further experiments to confirm this intuition.

5 Extension of GRSK to Groups

RS usually give a recommendation for a single user considering his/her interests and tastes. However, many activities such as watching a movie or going to a restaurant involve a group of users, in which case recommendations must take into account the likes and tastes of all members in the group, thus giving rise to a Group Recommender System (GRS). In GRSs, individual profiles are merged so as to elicit a single set of preferences that represents the preferences of the group as a whole, and elicit a set of recommendations as if the group were a single user.

The first task of a GRS is to identify the individual preferences and then find a compromise that is accepted by all the group members. This is the crucial point in GRSs because how individual preferences are managed to come up with the group preferences will determine the success of the recommendation [12,7]. The purpose is that of elicitng a recommendation that equally satisfies, as much as possible, all the users and that no member is particularly satisfied or dissatisfied with the final recommendations.

This section outlines how our system is able to provide a recommendation for a group. When the GRSK receives a recommendation request from a group, it redirects

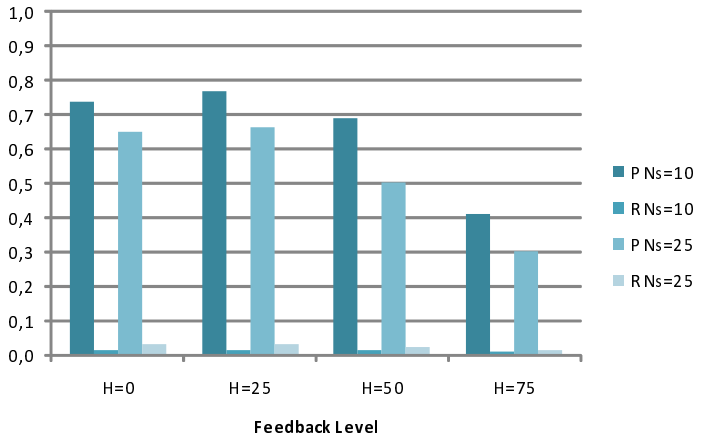

Fig. 6. Comparison of the P and R values obtained when Ns=10 and Ns=25 and for the four degrees of historical information for the movies domain

the request to the *Group Preference Manager* (GPM) which is in charge of controlling the group recommendation process. This process is composed of the following steps:

1. **Elicit the Individual Preferences.** The individual profiles of the users in the group are analyzed, and a list of preferences for each user in the group is elicited by using the Basic Recommendation Techniques (BRT). Each BRT returns a different list of individual preferences. Thus, after this stage, we have, for each user u in the group, four lists of individual preferences ($P^u_{demographic}$, $P^u_{collaborative}$, $P^u_{content-based}$, $P^u_{general-preferences}$) which describe the usual tastes of the user according to the criteria used by the BRT.
2. **Elicit the Group Preferences.** The aim of this process is to elicit a group preference model that reflects the preferences of all group members. Once the individual preferences are modeled (P^u_{brt}), they are merged in order to elicit the group preference model (P^G). This process is performed through the application of methods like *Aggregation*, *Intersection*, *Incremental Intersection* or *Incremental Collaborative Intersection*, which combine the individual preferences and generate the final group preference model P^G, composed of as many lists as applied BRT (one group preference list per BRT). From this point on, the GRSK follows the usual steps to make the final recommendations, as if P^G were the profile of a single user.
3. **Elicit the List of Recommended Items.** The Items Selector selects the items that match the group preference model and elicit the final recommendations. The Hybrid Technique joins the lists of group preferences into a single list, and the best recommendations are finally elicited for the group.

At this moment, the GPM makes use of four disjunctive methods to elicit the group preferences: *Aggregation*, *Intersection*, *Incremental Intersection* and *Incremental Collaborative Intersection*. These methods differ on the way the lists of individual preferences are merged. The work in [6] presents some comparison results on the application of these techniques.

The **Aggregation** mechanism is a common technique that is used in several GRSs. Aggregation gathers the preferences of all members in the group computed by each BRT, and builds a single set of preferences. A preference is included in the group profile if it belongs to the individual profile of at least one of the members in the group.

The **Intersection** finds the preferences that are shared by all the members in the group to build the group preference model. The advantage of this mechanism is that all of the users in the group will be equally satisfied with the resulting group model. However, the risk of using intersection is that we might end up with an empty list of preferences if the group is rather heterogeneous.

The **Incremental Intersection**, based on a voting strategy, incrementally lessens the number of users that should satisfy a particular preference, and hence it is always able to calculate a recommendation for a group. This mechanism reports good results as it brings together the benefits of the Aggregation and the Intersection techniques.

The **Incremental Collaborative Intersection** selects the preferences that better match the group among the preferences included in the Intersection list of the group. This technique does not consider all the preferences shared by all group members but only the ones with a higher ratio. From these preferences, and by using a collaborative RS, the Incremental Collaborative Intersection elicits a new set of preferences for the group. This technique makes recommendations that incrementally satisfy the group satisfaction in contexts like a tourism domain, thus alleviating the limitations of the basic recommendation techniques.

As [6] shows, the selection of the appropriate group modeling techniques depends on issues such as the characteristics of the application domain or the available information about the users.

6 Related Work

Some general-purpose domain independent open source libraries and engines have been developed in order to reuse the effort to design recommender systems. Some of these systems are: *RACOFI* [2], *SUGGEST* [5], *Vogoo* [9], *Taste*[3], *CoFE* [10], *ColFi* [3], *Duine Toolkit* [16] and *Aura* [8].

Most of these engines are Java-based, with the exception of *SUGGEST* (C) and *Vogoo* (PHP). At this moment, we have two versions of the GRSK, one written in Java and the other in C#. GRSK Java version is agent-based like *RACOFI* and *Aura*.

Duine Toolkit, *RACOFI* and *ColFi* are developed with a modular architecture that allows developers to change and add algorithms easily, in the same manner than GRSK. The GRSK configuration process allows to select which techniques to use and to parameterize different aspects of the recommendation process, in order to adjust the GRSK behavior to the particular application domain.

Most of these systems are collaborative recommendation engines (*ColFi*, *Cofi*, *Taste*, *SUGGEST* and *Vogoo*). *RACOFI*, *Aura* and *Duine Toolkit* are hybrid recommendation engines. *RACOFI* adjusts a collaborative filter prediction with mechanisms coming from content-based approaches. *Aura* uses collaborative recommendation but uses a

[3] http://www.opentaste.net/

mechanism that assigns and processes a set of tags to items to improve the recommendation. *Duine Toolkit* uses collaborative and content-based techniques.

GRSK is an hybrid recommendation engine that employs different basic and hybrid recommendation techniques. The purpose of including these different recommendation techniques is to make GRSK able to work with any application domain, independently from the number of users, the available user information, etc. On the other hand, it is based on the semantic description of the items in the domain.

7 Conclusions and Further Work

The GRSK, a *Generalist Recommender System Kernel*, relies on a specification of the domain represented within an ontology, thus being able to apply to any domain as long as there exists such an ontological representation. This makes the GRSK be *generalist* or domain-independent. The GRSK uses four Basic Recommendation Techniques (demographic, content-based, collaborative and general likes filtering), and two different Hybrid Techniques (mixed and weighed) to create the final ranked list of recommended items. The GRSK allows selecting the techniques and parameters that best suit the particular application domain. We have also shown the results obtained when applying the GRSK to two different application domains, e-Tourism and e-Movies, highlighting the adaptability of the GRSK, and showing the strengths and weaknesses on each domain.

Currently, we are working on the extension of GRSK to group recommendation [6]. Different innovative techniques like Incremental Intersection and Collaborative Incremental Intersection are being developed to compute the group preference model. Additionally, we are also working on the use of agreement techniques for group recommendations. The members of the group are modeled as agents who attempt achieving a reconciled solution for the whole group maximizing the user satisfaction. The inclusion of this technique will allow us to incorporate more sophisticated human-like behaviors into the group.

Acknowledgements. Partial support provided by Consolider Ingenio 2010 CSD2007-00022, Spanish Government Project MICINN TIN2008-6701-C03-01 and Valencian Government Project Prometeo 2008/051.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749 (2005)
2. Anderson, M., Ball, M., Boley, H., Greene, S., Howse, N., Lemire, D., McGrath, S.: Racofi: Rule-applying collaborative filtering systems. In: IEEE WIC COLA (2003)
3. Brozovsky, L.: Recommender system for a dating service. Master's thesis, KSI, MFF UK, Prague, Czech Republic (2006)
4. Burke, R.: The Adaptive Web, chapter Hybrid web recommender systems, pp. 377–408. Springer, Heidelberg (2007)
5. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transactions on Information Systems 22(1), 143–177 (2004)

6. Garcia, I., Sebastia, L., Onaindia, E., Guzman, C.: A Group Recommender System for Tourist Activities. In: Di Noia, T., Buccafurri, F. (eds.) EC-Web 2009. LNCS, vol. 5692, pp. 26–37. Springer, Heidelberg (2009)
7. Jameson, A.: More than the sum of its members: Challenges for group recommender systems. In: Proceedings of the International Working Conference on Advanced Visual Interfaces, pp. 48–54. ACM, New York (2004)
8. Lamere, P., Green, S.: Project aura - recommendation for the rest of us. JavaOne (2008)
9. Lemire, D., McGrath, S.: Implementing a rating-based item-to-item recommender system in php/sql. D-01, On delette.com (2005)
10. Ogston, E., Bakker, A., van Steen, M.: On the value of random opinions in decentralized recommendation. In: Eliassen, F., Montresor, A. (eds.) DAIS 2006. LNCS, vol. 4025, pp. 84–98. Springer, Heidelberg (2006)
11. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artificial Intelligence Review 13, 393–408 (1999)
12. Plua, C., Jameson, A.: Collaborative preference elicitation in a group travel recommender system. In: Proceedings of the AH Workshop on Recommendation and Personalization in eCommerce, Malaga, Spain, pp. 148–154 (2002)
13. Resnick, P., Varian, H.: Recommender systems. Communications of the ACM 40(3), 56–58 (1997)
14. Sebastia, L., Garcia, I., Onaindia, E., Guzman, C.: e-Tourism: a tourist recommendation and planning application. International Journal on Artificial Intelligence Tools (WSPC-IJAIT) 18(5), 717–738 (2009)
15. Li, T., Anand, S.S.: Exploiting Domain Knowledge by Automated Taxonomy Generation in Recommender Systems. In: Di Noia, T., Buccafurri, F. (eds.) EC-Web 2009. LNCS, vol. 5692, pp. 120–131. Springer, Heidelberg (2009)
16. van Setten, M., Reitsma, J., Ebben, P.: Duine toolkit - user manual. Technical report, Telematica Instituut (2006)