Li X, Guo L, Zhao YE (2008) Tag-based social interest discovery. In: WWW'08, Beijing, pp 675–684
Liang H, Xu Y, Li Y (2012) Mining users' opinions based on item folksonomy and taxonomy for personalized recommender systems. In: ICDM'10, Sydney
Linden G, Smith B, York J (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80
Noll MG, Meinel C (2007) Web search personalization via social bookmarking and tagging. In: ISWC'07/ASWC'07, Busan, pp 367–380
van den Oord A, Dieleman, S., Schrauwen, B. (2013) Deep content-based music recommendation, Proceedings advances in Neural Information Processing Systems (NIPS), pp 2643–2651
Ostuni VC, Di Noia T, Di Sciascio E, Mirizzi R (2013) Top-N recommendations from implicit feedback leveraging linked open data. In: Proceedings of the 7th ACM conference on Recommender systems (RecSys '13). ACM, New York, pp 85–92
Pitkow J, Schütze H, Cass T, Cooley R, Turnbull D, Edmonds A, Adar E, Breuel T (2002) Personalized search. Commun ACM 45(9):50–55
Rendle S, Schmidt-Thieme L (2010) Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM'10, New York, pp 81–90
Rendle S, Balby Marinho L, Nanopoulos A, Lars S-T (2009) Learning optimal ranking with tensor factorization for tag recommendation. In: SIGKDD'09, Paris, pp 727–736
Sen S, Vig J, Riedl JT (2009) Tagommenders: connecting users to items through tags. In: WWW'09, Madrid, pp 671–680
Seth A, Zhang J (2008) A social network based approach to personalized recommendation of participatory media content. In: ICWSM'08, Seattle
Shepitsen A, Gemmell J, Mobasher B, Burke R (2008) Personalized recommendation in social tagging systems using hierarchical clustering. In: RecSys'08, Lausanne, pp 259–266
Symeonidis P, Nanopoulos A, Manolopoulos Y (2008) Tag recommendations based on tensor dimensionality reduction. In: RecSys'08, Lausanne, pp 43–50
Tso-Sutter KHL, Marinho LB, Schmidt-Thieme L (2008) Tag-aware recommender systems by fusion of collaborative filtering algorithms. In: SAC'08, Fortaleza, pp 1995–1999
Vatturi PK, Geyer W, Dugan C, Muller M, Brownholtz B (2008) Tag-based filtering for personalized bookmark recommendations. In: CIKM'08, Napa Valley, pp 1395–1396
Vig J, Sen S, Riedl JT (2009) Tagsplanations: explaining recommendations using tags. In: IUI'09, Sanibel Island, pp 47–56
Vig J, Soukup M, Sen S, Riedl JT (2010) Tag expression: tagging with feeling. In: UIST'10, New York, pp 323–332
Wang H, Wang N, Yeung D-Y (2015) Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1235–1244
Xu G, Gu Y, Dolog P, Zhang Y, Kitsuregawa M (2011a) Semrec: a semantic enhancement framework for tag based recommendation. In: AAAI'11, San Francisco, pp 1267–1272
Xu G, Gu Y, Zhang Y, Yang Z, Kitsuregawa M (2011b) Toast: a topic-oriented tag-based recommender system. In: WISE'11, Sydney, pp 158–171
Zanardi V, Capra L (2011) A scalable tag-based recommender system for new users of the Social Web. In: DEXA'11, Toulouse, pp 542–557
Zhen Y, Li W-J, Yeung D-Y (2009) Tagicofi: tag informed collaborative filtering. In: RecSys'09, New York, pp 69–76

Recommended Reading

Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems – an introduction. Cambridge University Press, Leiden
Ricci F, Rokach L, Shapira B, Kantor PB (eds) (2011) Recommender systems handbook. Springer, New York

Recommender Systems: Models and Techniques

Francesco Ricci
Faculty of Computer Science, Free University of Bozen-Bolzano, Bozen-Bolzano, Italy

Synonyms

Advisory systems; Recommendation systems

Glossary

Context	Situational factors influencing the evaluation of a user for an item
Experience	The interaction of a user with an item that is resulting in an evaluation
Evaluation Prediction	The system's prediction of the user's evaluation for an item
Information Filtering	Technique for providing only relevant information to a user

Item	Information content that can be recommended by a RS
Personalization	Providing a user with content adapted or suited to their needs and wants
Preferences	A structured representation of the user preferences for items
Recommendations	System's selected items that are suggested to a user
RSs	Recommender systems
Situation	Conditions under which an item is evaluated by a user
Tag	Metadata in the form of freely chosen keyword

Definition

RSs are information search and filtering tools that provide suggestions for items to be of use to a user. They have become common in a large number of Internet applications, helping users to make better choices while searching for news, music, vacations, or financial investments. RSs exploit data mining and information retrieval techniques to predict to what extent an item suits the user needs and wants and recommend those items with the largest predicted fit score.

Introduction

The explosive growth and variety of information available on the Web and the rapid introduction of new e-business and social services (buying products, product comparison, auction, forums, social networking, multimedia fruition) have created such a richness of choices that, instead of producing a benefit, this overabundance risks to backfire. While choice is good, more choice is not always better. Indeed, choice, with its implications of freedom, autonomy, and self-determination, can become excessive, creating a sense that freedom may come to be regarded as a kind of misery-inducing tyranny (Schwartz 2004). Moreover, if dozens of different types of jams are likely to confuse and paralyze a buyer, as it is illustrated in Schwartz (2004), thousands or even millions of songs are simply impossible to scan if the ultimate goal is just to play some of them.

Such a scenario motivated the introduction of recommender systems (RSs) (Ricci et al. 2011; Konstan and Riedl 2012). RSs are information search and filtering tools that provide suggestions for items to be of use to a user. They have become common in a large number of Internet applications, helping users to make better choices while searching for news, music, vacations, or financial investments. "Item" is the general term used to denote what the system recommends to its users, and a specific RS normally focuses on one type of items (e.g., movies or news). Accordingly, its core algorithmic component and its graphical user interface are customized to provide useful and effective suggestions for that specific type of items. Recommender systems play an important role in highly rated Internet sites, such as Amazon.com, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, and IMDb. More recently, social networks, such as Linkedin and Facebook, have introduced recommender systems to suggest groups to join or people to relate with.

In their simplest form, personalized recommendations are offered as customized lists of items. In performing this selection the system tries to predict what the most suitable products or services (items) are, based on the user's characteristics and preferences. In order to complete such a computational task, a RS must elicit from users such characteristics and preferences, either along the full history of previous interactions with the users or exploiting information entered by the users at the time the recommendation is requested. Moreover, such information either can be explicitly expressed, e.g., as ratings for products, or can be inferred by interpreting user actions. For instance, the navigation to a particular page can be interpreted as an implicit sign of preference for the items shown on that page.

The study of recommender systems is relatively new compared to research in other classical information system tools and techniques (e.g., databases or search engines). Recommender systems emerged as an independent research area in

the mid-1990s (Goldberg et al. 1992; Resnick et al. 1994), and it is still fast growing. Research works on RSs are published in major conferences on machine learning (ICML, KDD, NIPS), information retrieval (SIGIR, WISDM, CIKM), intelligent user interfaces (IUI), and personalization (UMAP). A specific ACM conference on recommender systems has been launched on 2007, and every year it attracts more and more submissions and attendees. In total, thousands of papers are published every year on this subject.

In this paper we provide a description of the general computational model of a recommender system. We aim at modeling in a compact but rigorous presentation the core functionality of a recommender system. We will decompose it into three fundamental tasks: user's preferences elicitation, prediction of user's evaluations for items, and recommendations generation and presentation. In the last section we will conclude this short article with a discussion of some challenges for RSs.

Recommendation Model and Techniques

A general computational model for recommender systems was previously described in Adomavicius and Tuzhilin (2005). In this model, a RS is defined as a machinery implementing a real-valued function defined on the product space of the users and items $r^*: U \times I \rightarrow \mathcal{R}$ that predicts how a pair consisting of a user $u \in U$ and an item $i \in I$ is mapped to the evaluation $r^*(u, i)$ of the user u for the item i. They call this number $r^*(u, i)$ the predicted "utility" of the item for the user. We prefer here not to use the term "utility," as in the RS literature no special assumption is made on the characteristics of the evaluation function, while a utility function does satisfy specific constraints. For that reason we call it "predicted evaluation." Then, having predicted evaluations of users for items, a RS recommends to a user u the items i with the largest predicted evaluations $r^*(u, i)$. Evaluations are called ratings of users for items in Collaborative Filtering RSs (see next section on Evaluation Prediction Techniques). A RS computes the prediction $r^*(u, i)$ on the base of a collection of observations: these are interactions between users and items, and they provide the system with information about the users' preferences. In many cases these interactions produce explicit evaluations performed by some users on some items. In some other cases, more complex types of relationships are observed, for instance, the relative preference of a user for an item when it is compared to another item.

In this survey we introduce a generalization of this two-dimensional model (Users × Items) that is inspired by the multidimensional model introduced in Adomavicius et al. (2005), and it is motivated by recent researches on social networks and tagging recommender systems (Marinho et al. 2011). Moreover, we delineate a model for RSs that decomposes its behavior into three fundamental tasks: preference elicitation, item evaluation prediction, and recommendation generation.

Preference Elicitation

The first task that a recommender system must implement is the elicitation of user preferences; this means, in RS terminology, that the system must be able to collect from users a set of evaluations for items.

More formally, let us assume that there exist three sets U, I, and M. U is the set of users, I is the set of items, and M is the set of possible situations under which the items can be experienced. For instance, M can be the user's location when the item, e.g., a restaurant, was searched and selected or the location of the user when a recommendation for a restaurant is required (more on this is discussed later on).

Then the experience of a user for an item is a quadruple $(u, i, m, r(u, i, m)) \in U \times I \times M \times \mathcal{R}$ where u is the user who interacted with item i in the situation m and evaluated the item $r(u, i, m) \in \mathcal{R}$. We use here the function notation $r(u, i, m)$ to stress that the evaluation is a function of the user, the item, and the situation. $\mathcal{R}$ is an evaluation scale containing the possible (ordered) evaluation values. Evaluations are commonly called ratings, and a popular evaluation scale is the finite set $\{1, 2, \ldots, 5\}$. This scale is used, for instance, by Amazon.com. But, different evaluation scales are

possible and have been used both in the scientific literature and in some deployed RSs; for instance, the simpler evaluation schema provided by positive ("like," +1) vs. negative ("not like," −1) evaluation is used in YouTube.com. It is interesting to note that rating scales are not neutral and they influence the user evaluation process (Kuflik et al. 2012) and consequently the RS behavior.

A recommender system, in order to generate recommendations, must first collect a set of experiences from the users in U, for items in I, in some situations M, i.e., to acquire a set $E(D) = \{(u, i, m, r(u, i, m)): (u, i, m) \in D \subset U \times I \times M\}$. The evaluation $r(u, i, m)$ can be collected either explicitly or implicitly. By "explicitly" we mean that the user is explicitly entering in some form her evaluation $r(u, i, m)$ on the presented item. In "implicit" models the user is not entering evaluations but is acting on the presented items, e.g., is watching a recommended video, or is browsing the presented information about an item. The system is then inferring the user evaluation (a value in $\mathcal{R}$) from the user action. We first discuss the "explicit" approach and then the "implicit" one.

Many recommender systems allow users to "explicitly" evaluate (rate) items as they encounter them while interacting with the system. In addition, many RSs explicitly request the user to evaluate a certain number of items (e.g., 20 books) before providing recommendations (for new books). This may happen in the sign up stage, i.e., when the user registers to the system and obtains the credential to access it. Another popular approach for collecting evaluations is to let the user to "correct" the system predictions by entering evaluations for items that have been recommended, hence possibly fixing erroneous predictions. In this case the system may learn that some of the recommended items are not good options, i.e., the user may enter low evaluations for them.

In more sophisticated approaches the system may implement a precise preference elicitation strategy by applying active learning techniques (Rubens et al. 2011). So, for instance, the system may identify the most popular items and request the user to rate them, with the objective of maximizing the probability that the user knows these items and can really evaluate them. Or, in a completely different approach, the system may ask the user to evaluate the items that have received so far the most diverse evaluations, since the opinion of the user on these items can better reveal the specific users' preferences.

In "implicit" feedback approaches the user is not directly entering evaluations for items by choosing values from the evaluation scale $\mathcal{R}$ considered by the system. So, for instance, the user is not entering a star-based score for the books she has read. The underlying assumption is that the user may not want to spend time for this task; hence, the system must infer the evaluations in the target $\mathcal{R}$ scale from another measure in another scale $\mathcal{R}'$. For instance, in the music recommender system presented in Moling et al. (2012) the system is measuring the percentage of a suggested track that is actually listened to by the user to decide what type of track to recommend next. Another popular implicit evaluation scale is the number of times a user visited the item (played a track or browsed a page). Yet another example of the "implicit" approach is offered by systems that do not ask the user to evaluate individual items, but allow them to compare two items or to criticize them (McGinty and Reilly 2011). In this case the system first presents some recommended items. These are primarily generated for letting the user to specify the characteristics of the preferred items. In response, the user can inform the system that the presented items do not completely conform to her preferences and select one item that is almost good but is still lacking a preferred feature (e.g., it should be cheaper). In these cases the system uses the user input, which is composed by the selected item and the "critique," to update the evaluation prediction function for that user $r^*(u, \cdot, \cdot)$.

Item Evaluation Prediction

The second task, item evaluation prediction, uses a data set of user-provided evaluations to predict new evaluations. In fact, this is the most important task of a recommender system: exploiting a data set of experiences, its background knowledge, $E(D) = \{(u, i, m, r(u, i, m)): (u, i, m) \in D \subset U \times I \times M\}$ to generate predictions of the user evaluations for

other experiences E^* (D^*) = {(u, i, m, $r^*(u, i, m)$): (u, i, m) $\in D^* \subset U \times I \times M$}, where $r^*(\cdot, \cdot, \cdot)$ is the evaluation prediction function that is estimated with a specific recommendation technology. In the next section, we will present some techniques, e.g., collaborative filtering or content-based, which have been introduced to compute the evaluation prediction function $r^*(\cdot, \cdot, \cdot)$. D^* is the set containing the user, item, and situation combinations for which the recommender system can generate evaluation predictions. D^* is disjoint from D and may be equal to $(U \times I \times M) \setminus D$ or smaller. It is smaller when the RS is not able to generate evaluation predictions for all the items and situations combinations that the user has not evaluated yet. In fact, recommender systems find it difficult, for instance, to make recommendations for new users and new items. These are users who never entered any evaluation in the system or items that were never evaluated by any user (users and items not present in D) (Ricci et al. 2011; Konstan and Riedl 2012).

We note that I, U, and M are sets of objects and may, or may not, have an internal structure, i.e., they may have features. For instance, in plain collaborative filtering systems, which will be described later on, items' and users' features, even if available, are not exploited, and the situation space is ignored. For that reason the evaluation function is modeled as a matrix $\mathbf{R} = [r_{u,i}]_{m \times n}$, where u and i are simple indexes ranging from 1 to m and n, respectively, and $r_{u,i} \in \mathcal{R}$. Whereas, in content-based systems, items have an internal structure and are described with features. Items' features are used to generate user-specific classifiers that can predict the user evaluation for (unseen) items (Lops et al. 2011).

It is worth noting that some recommender systems do not collect user evaluations for items in order to make predictions, i.e., they can make recommendations even though $E(D)$ is empty. For instance, case-based recommender systems let the user to enter a partial description of the preferred item q, as query, and then, using this input, they generate predictions $r^*(u, i, q)$ of the user evaluation for item i. They accomplish this task by exploiting the similarity of q and i, based on some description of i. Hence, in CBR RSs, the situations set M is the space of all possible user queries (Bridge et al. 2006).

Evaluation Prediction Techniques

In order to implement the item evaluation prediction function, RSs can exploit a range of techniques. This has been the major topic of research in RSs. We will here briefly indicate the most important types of techniques, referring the interested reader to Ricci et al. (2011) for more examples, references, and details.

Recommendation techniques vary in terms of the addressed domain, the knowledge used, and the recommendation algorithm, i.e., essentially how the item evaluation prediction is actually computed. We provide here an overview of the different types of RS techniques by quoting a taxonomy introduced by Burke (2007) that has become classical for distinguishing between recommender systems and referring to them. Burke (2007) lists six different types of recommendation approaches.

Content-Based

The system implements for each user a "classifier" that learns to evaluate (classify) higher the items that are similar to the ones that the user evaluated higher in the past. The similarity of items, or more in general the item classification rule, is calculated based on the features associated with the compared items. For example, if a user has systematically positively rated movies that belong to the action genre, then the system can learn that other movies from this genre should have a high value for that user (Lops et al. 2011).

Collaborative Filtering

The simplest and original implementation of this approach predicts that the active user, i.e., the user asking for recommendations, will evaluate higher the items that other users with similar tastes liked in the past (Desrosiers and Karypis 2011). The similarity in taste of two users is calculated based on the similarity of the evaluations' history of the users. Collaborative filtering is probably the most popular and widely implemented technique in RSs. The latest approaches to CF use latent factor models, such as matrix factorization (e.g.,

using Singular Value Decomposition, SVD). These methods map both items and users to the same latent factor space. Then the predicted evaluation of a user for an item is basically computed by the dot multiplication of their representative vectors, which gives a kind of similarity between the user and the item in this common representation space (Koren and Bell 2011).

Demographic

These techniques predict item evaluations based on the demographic profile of the user. The assumption is that different recommendations should be generated for different demographic niches. Many Web sites adopt simple and effective personalization solutions based on demographics. For example, users are dispatched to particular Web sites based on their language or country. Or suggestions may be customized according to the age of the user.

Knowledge-Based

Knowledge-based systems predict item evaluations based on specific domain knowledge about how certain item features meet user's needs and preferences and ultimately how the item is useful for the user. Notable knowledge-based recommender systems are case-based (Bridge et al. 2006). In these systems a similarity function estimates how much the user needs (problem description) match the recommendations (solutions of the problem). Here the similarity score can be directly interpreted as the predicted item evaluation of the user. Another group of knowledge-based systems uses constraints, to represent user preferences and to find relevant items (Jannach et al. 2010).

Community-Based

In this type of systems, item evaluation predictions are based on the preferences of the user's friends. Evidence suggests that people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals. This observation, combined with the growing popularity of open social networks, is generating a rising interest in community-based systems or social recommender systems (Golbeck 2006). This type of RS techniques acquires and exploits information about the social relations of the users and the preferences of the user's friends. The item evaluation predictions are based on ratings that were provided by the user's friends.

Hybrid Recommender Systems

These RSs are based on the combination of the abovementioned techniques. A hybrid system combining techniques A and B tries to use the advantages of A to fix the disadvantages of B. For instance, CF methods suffer from new-item problems, i.e., they cannot generate evaluation predictions for items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available. Given two (or more) basic RS techniques, several ways have been proposed for combining them to create a new hybrid system (see Burke 2007 for the precise descriptions).

Recommendation Generation

RSs, after having generated evaluation predictions for items, can generate recommendations, i.e., absolve their primary role. The classical and common approach for recommendation generation is to recommend to user u in situation m the N items i that have the largest predicted evaluation in that situation, i.e., the set $TopN(u, m) \subset I$, where $|TopN(u, m)| = N$, and if $i \in TopN(u, m)$, then $r^*(u, i, m) \geq r^*(u, j, m)$, for all $j \notin TopN(u, m)$ (Adomavicius and Tuzhilin 2005). N is normally a small value, such as 5 or 10. Reducing the number of recommendations is essential to address the main goal of a recommender system, as we mentioned in the Introduction, namely, to filter irrelevant items and simplify the user's decision-making process.

Similarly, the system can rank all the items for which the recommender can generate an evaluation prediction and present to the user the ranked list of these items, ordered by decreasing value of the predicted evaluation $r^*(u, i, m)$. It is worth noting that this ranked list is not generally equal to the full set of items I, since, as we mentioned above, the RS may not be able to generate an

evaluation prediction for all the items (in all the situations). Hence, in practice this ranked list contains the $TopN(u, m)$ items for a large value of N.

In fact, there is a growing understanding that this apparently obvious design choice may not be the most appropriate in many cases. For instance, the top N items may be all very similar; hence, it would be more useful to introduce in the recommendation list those items, which may have a lower predicted evaluation but are, all together, providing a more useful information to the user. The essential point that we raise here is that, while the item evaluation prediction function estimates to what extent the user likes an item, the user decides what item to select by browsing the recommendation list. Hence, items presented in the recommendation list must fulfil two, possibly conflicting, criteria: be interesting to the user and help the user to make a decision. As the diversity issue points out, items that can better help the user to make a decision may not be just the best top five items that he may select.

Another related issue is pertaining to the information that is provided together with the recommendations. In practice this has been dealt by including explanations for the recommendations (Tintarev and Masthoff 2012). Explanations may be directed to enhance the transparency or the scrutability of the system, i.e., giving to the users hints about the system internal behavior. Or they may increase the trust, the persuasiveness, and the subjective satisfaction of the user. Or ultimately explanations can improve the efficiency and effectiveness of the decision-making process supported by the recommender.

Key Applications

As we mentioned above, the illustrated recommender system model focuses on three types of entities: users U, items I, and situations M. In the following sections we will illustrate three key applications of this model, where the situation space is representing, the context, or the group of users, or a tag. We must stress that the simpler and more popular model, which is described in Adomavicius and Tuzhilin (2005), does not consider any situation space M. In that case the recommendations are generated for a user, independently from any other additional information that may specify the recommendation situation.

Context-Aware Recommender Systems

The first application of the model described in the previous section refers to context-aware recommender systems (Adomavicius et al. 2011). In this case M contains the possible alternative contextual situations that may be concurrent with the experience that a user makes of an item and can have an impact on the item evaluation. The goal is to make predictions of user evaluations for items in a particular target contextual condition m.

For instance, Baltrunas et al. (2012) describe a place of interests (POIs) recommender system for the tourism domain, where 14 contextual factors are considered. To mention some examples, there are factors that describe the time of the travel, the weather condition, the mood of the user, or the type of group that is accompanying the traveler. We refer the reader to Baltrunas et al. (2012) for a detailed description of these contextual factors. For each factor a finite set of possible values is defined. For instance, the weather factor can take the values: snowing, clear sky, sunny, and rainy. Hence, in this recommender system M is the space of all the possible combinations of the values for these 14 contextual factors: $M = F_1 \times \cdots F_{14}$, where, for instance, $F_9 = \{\text{snowing, clear sky, sunny, rainy}\}$ is the weather factor that we mentioned above. This system uses a collection of users' evaluations for a collection of places of interests in Bolzano (items), in different contextual situations, to predict the users' evaluations for POIs not yet experienced in some possibly new contextual situations.

Context-aware RSs are now an active research area and we refer the reader to Adomavicius et al. (2011) for more examples and techniques. The major technical difficulties are related to the following: understanding the impact of the contextual factors on the personalization process; selecting (dynamically) the right factors, i.e., relevant in a particular personalization task; obtaining sufficient and reliable data describing the user preferences in context; and embedding the contextual information in a more classical recommendation computation technique.

Group Recommender Systems

The second application refers to group recommender systems (Jameson and Smyth 2007). In this case M represents the possible groups of users that u may belong to, and the ultimate system goal is to offer the same set of recommendations for items to the users belonging to a group. The underlying assumption is that the items will be experienced together, e.g., in a travel recommender system, the users will travel together to the recommended place. The data set of the users' experiences $E(D)$ in this case contains quadruples of the type $(u, i, g, r(u, i, g))$, where g is a subset of users in U that contains u, and $r(u, i, g)$ is the evaluation provided by the user u, for item i, when the item was experienced together with the other users in g. The idea is that the user evaluation is influenced by the presence of other users (Masthoff 2011), i.e., $r(u, i, g)$ is in principle different from $r(u, i, h)$ if $h \neq g$.

A group recommender system with such background knowledge must predict the evaluation of u for other items, let us say i' when she is together with the users in g'. It is worth noting that no group recommender system is actually making predictions for the user evaluations as a function of the group g the user belongs to. Conversely, the current leading approach is to first predict the individual evaluations independently from the group, using a standard two-dimensional model, i.e., neglecting the situation in M, which means that $r^*(u, i, h) = r^*(u, i, g) = r^*(u, i)$, for all groups $h, g \in M$. Then, in a second step, group RSs compute the "group evaluation prediction" for an item by aggregating the various evaluation predictions for that item for the groups members. So, for instance, if the average aggregation method is used, the predicted evaluation of the group g for the item i is $AVG_{u \in g} \{ r^*(u, i)\}$, and the items with the largest group aggregated evaluations are recommended.

Other approaches, instead of aggregating evaluation predictions, aggregate the input evaluations of the users u belonging to a group g, hence generating (fictitious) group evaluations $r(g, i, g)$. These group evaluations, which constitute the group model, are again computed by, for instance, averaging the evaluations of the users in the group for a given item. Then, the group evaluation predictions are computed for these fictitious users considering them as normal users. One can describe this approach in the model proposed in this article, by introducing new fictitious users that represent groups g, and then the evaluation prediction function for a group g is $r^*(g, \cdot, g)$.

The group recommendation application is stressing once more the fundamental, but so far neglected, difference between the evaluation prediction and the recommendation generation tasks of a recommender system. As we mentioned above, the best recommendations may not be for the items having the largest predicted evaluation. For group recommendations, a better approach is instead to really try to predict $r(u, i, g)$, i.e., the evaluation of u for item i when experienced together with the other users in g. Then the system must compute recommendations for u when she is together with the users in g not simply by taking the highest predicted items, but trying to generate a recommendation list that is really useful for the group, e.g., by combining items that would help the group to make a joint and agreed decision.

Tag-Based Recommender Systems

The last application refers to social tagging recommender systems (Marinho et al. 2011). In social tagging systems, such as Delicious, BibSonomy, and Last.fm, users can search and browse the items managed by the system, which in these three examples are bookmarks, bibliographic references, and music, respectively. But, in addition to this basic functionality, users can tag the items with tags, i.e., metadata in the form of freely chosen keywords. On top of these systems, various kinds of recommendations are possible. One can recommend items to user, but also tags to be assigned by a user to an item, or even users to users. In tag-based recommender systems, M, the space of situations, is a tag vocabulary V. An evaluation $r(u, i, t)$ in these systems is taking values just in the set $\{1\}$, where $r(u, i, t) = 1$ means that the user u has tagged the item i with the tag t. Hence, in this scenario, a set of collected experiences $E(D) = \{(u, i, t, 1): (u, i, t) \in D \subset U \times I \times V\}$ represents tag assignments performed by users to items. This set of

experiences, actually considering only the triples (u, i, t), i.e., discarding the redundant 1, is called in social tag systems a Folksonomy. So, given a Folksonomy, the evaluation prediction task of a tag-based recommender system can be specialized by saying that it predicts whether a user u will assign the tag t to the item i, or in other words, if the triple (u, i, t) not yet included in the Folksonomy should be added to it.

As we mentioned above, while the basic evaluation prediction task in this case is predicting if $r(u, i, t)$ is 1, i.e., if a user will assign a tag to an item, various specific recommendation generation tasks have been considered. The two more popular are to recommend a set of items to a user and to recommend a set of tags to a user for tagging an item. The first one is the classical (two-dimensional) recommendation task of a RS, i.e., where the recommendations are not supposed to be dependent on the tag that the user is giving to the target item (the situation). In fact, tag assignments are used to generate item recommendations, e.g., by using tag assignments to identify similar items (tagged in a similar way) or similar users (tagging common items) in collaborative filtering techniques. But in this case, the recommendations offered to a user are not depending on the tags that the user may have selected for the recommended item. In this case, the original three-dimensional matrix of observed tag assignments (experiences) $r(u, i, t) = 1$ can be projected into the two classical two-dimension space of users and items evaluations $r(u, i)$. This is performed by simply assigning to $r(u, i)$ the value 1 if there exists a tag t such that $r(u, i, t) = 1$. Then, on this two-dimensional space of $U \times I$, standard recommendation techniques, such as collaborative filtering, can be applied to compute evaluation predictions (i.e., if a user likes or not the items) and generate the final recommendation sets.

Conversely, if the goal is to recommend tags to a user for tagging an item, one must consider the original three-dimensional model. After having identified the triples (u, i, t) that are predicted, i.e., such that $r^*(u, i, t) = 1$, the system simply recommends the tag t for tagging the item i to the user u. In practice, prediction functions are scoring the triples not yet observed, e.g., by assigning a predicted evaluation in [0, 1], and then the N tags, for a given pair (u, i), that are predicted to score higher are recommended to the user u for tagging the item i. A large number of techniques have been proposed to compute this score, and we refer to Marinho et al. (2011) for a survey in this fast-growing research area.

Future Directions

The research on RSs is still very active, and numerous issues and challenges are still open. We want to list here some of them, with the obvious caveat that this list cannot be complete and is influenced by our personal vision. The reader is also referred to Konstan and Riedl (2012) for another discussion on the future of recommender systems.

Group Recommenders

Group recommenders deal with situations when it would be good if the system could recommend information or items that are relevant to a group of users rather than to an individual (Jameson and Smyth 2007). For instance, a RS may select television programs for a group to view or a sequence of songs to listen to, based on preference models of all group members (Masthoff 2011). Recommending to groups is clearly more complicated than recommending to individuals. Assuming that we know precisely what is good for individual users, the issue is how to combine individual user preferences or aggregate recommendations for the group members into group-specific recommendations (Jameson and Smyth 2007). Even if several techniques have been proposed so far (Masthoff 2011), very few experiments have been conducted in live-user studies (Senot et al. 2010), and it is challenging to define measures for assessing the quality of group recommendations.

Proactive Recommender Systems

Proactive recommenders can decide to push recommendations even if not explicitly requested. The largest majority of the recommender systems developed so far follow the "pull" model, where

the user originates the request for a recommendation. In the scenarios emerging today, where computers are ubiquitous and users are always connected, it seems natural to imagine that a RS can detect needs even if they are not explicitly stated by the user with a request. In this scenario, the system therefore must predict what to recommend but also when and how to "push" its recommendations. By accurately estimating when the RS can become proactive without being perceived as disturbing, the perceived utility of the recommendations may greatly increase.

Active Learning

RSs need to actively look for new data during the operational life (Rubens et al. 2011) (Active Learning). This issue is normally neglected on the assumption that there is not much space for controlling what data (e.g., ratings) the system can collect, because these decisions are autonomously taken by the users when visiting the system. Actually, a RS provokes the users with its recommendations. In fact, many systems (e.g., MovieLens.org) explicitly ask for user ratings during the recommendation process. Hence, by tuning the process, users can be pushed to enter into the system a range of different information. Specifically, they can be requested to rate particular items, because the knowledge of the user's opinions about these items is estimated as beneficial for improving a particular aspect of the system performance, e.g., in order to generate more diverse recommendations or just to improve the prediction accuracy. Some recent works have addressed these issues (Harpale and Yang 2008; Rashid et al. 2008), but more research activity is required to assess the proposed techniques in live-user studies and to design more adaptive solutions to the dynamics of the data managed by the system (Elahi et al. 2011; Golbandi et al. 2011).

Privacy Preserving

RSs exploit user data to generate personalized recommendations. In the attempt to build increasingly better recommendations, they collect as much user data as possible. This can clearly have a negative impact on the privacy of the users, and the users may start feeling that the system knows too much about their true preferences (Kobsa 2008). Therefore, there is a need to design solutions that will parsimoniously, sensibly, and cooperatively collect user data. At the same time these solutions will ensure that the acquired knowledge about the users cannot be freely accessed by malicious users.

Diversity

Assuring the diversity of the items recommended to a target user is an important feature of a recommender system (Smyth and McClave 2001). For instance, it is more likely that the user will find a suitable item in a recommendation list, if there is a certain degree of diversity among the recommendations. There is a limited value in having perfect recommendations for a restricted type of products, unless the user has expressed a narrow set of preferences. There are many situations, especially in the early stage of a recommendation process, in which the users want to explore new and diverse directions. In such cases, the user is using the recommender as a knowledge discovery tool. The research on this topic is still in an early stage and is now trying to characterize the nature of this "diversity" (Vargas and Castells 2011), i.e., whether the system must produce diversity among different recommendation sessions, or within a session (Adomavicius and Kwon 2012), and how to achieve simultaneously diversity and accuracy of the recommendations (Vargas and Castells 2011).

Generic User Models

Generic user models (Kobsa 2007) and cross-domain recommender systems are able to mediate user data (item evaluations) through different systems and application domains (Berkovsky et al. 2008). Using generic user model techniques, a single RS can produce recommendations about a variety of items. This is normally not possible for a traditional RS which can combine more techniques in a hybrid approach, but cannot easily benefit from the user preferences collected in one domain to generate recommendations in a different one. Solutions to this problem may further push the adoption

of personalized mobile recommender systems, running on the user personal communication device and offering ubiquitous support in many user's activities (Ricci 2011).

Sequential Recommendations

Recommender Systems may optimize a sequence of recommendations (Shani et al. 2005; Baccigalupo and Plaza 2006), for instance, a sequence of songs broadcast by a personalized radio channel. Sequential recommendations may be generated by systems that manage a structured dialogue between the user and the recommender. These systems are called conversational and have emerged in the attempt to improve the quality of the recommendations that are normally offered by simpler systems based on a onetime request/response. Conversational RSs can be further improved by implementing learning capabilities that can optimize not only the items that are recommended at each conversational step but also how the dialogue between the user and the system must unfold in all possible situations (Mahmood et al. 2009).

Robust Recommender Systems

Finally, the topic of robust recommender systems has become a major issue in the past few years. New research activities have focused on algorithms designed to generate more robust recommendations, i.e., recommendations that are harder for malicious users to influence. In fact, collaborative recommender systems are dependent on the goodwill of their users, i.e., there is an implicit assumption that users will interact with the system with the aim of getting good recommendations for themselves while providing useful data for their neighbors. However, users might have a range of purposes in interacting with RSs, and in some cases, these purposes may be opposed to those of the system owner or those of the majority of its user population. Namely, these users may want to damage the Web site that is hosting the recommender or to bias the recommendations, e.g., to score some items better or worse, rather than to arrive at a fair evaluation (Burke et al. 2011).

Cross-References

- ▶ Collective Intelligence: Overview
- ▶ Computational Trust Models
- ▶ Data Mining
- ▶ Distance and Similarity Measures
- ▶ E-Commerce and Internet Business
- ▶ Folksonomies
- ▶ Group Representation and Profiling
- ▶ Matrix Algebra, Basics of
- ▶ Matrix Decomposition
- ▶ Privacy and Disclosure in a Social Networking Community
- ▶ Recommender Systems using Social Network Analysis: Challenges and Future Trends
- ▶ Social Media
- ▶ Social Recommendation in Dynamic Networks
- ▶ Trust in Social Networks

References

Adomavicius G, Kwon Y (2012) Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans Knowl Data Eng 24(5):896–911

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749

Adomavicius G, Sankaranarayanan R, Sen S, Tuzhilin A (2005) Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans Inf Syst 23(1):103–145

Adomavicius G, Mobasher B, Ricci F, Tuzhilin A (2011) Context-aware recommender systems. AI Mag 32(3):67–80

Baccigalupo C, Plaza E (2006) Case-based sequential ordering of songs for playlist recommendation. In: Roth-Berghofer T, Göker MH, Güvenir HA (eds) Advances in case-based reasoning, Proceedings of the 8th European conference on case-based reasoning, ECCBR 2006, Fethiye. Lecture notes in computer science, vol 4106. Springer, pp 286–300

Baltrunas L, Ludwig B, Peer S, Ricci F (2012) Context relevance assessment and exploitation in mobile recommender systems. Personal Ubiquitous Comput 16(5):507–526

Berkovsky S, Kuflik T, Ricci F (2008) Mediation of user models for enhanced personalization in recommender systems. User Model User Adapt Interact 18(3):245–286

Bridge D, Göker M, McGinty L, Smyth B (2006) Case-based recommender systems. Knowl Eng Rev 20(3):315–320