Learning Preference Models in Recommender Systems

Giovanni Semeraro, Fedelucio Narducci, Marco de Gemmis, Giovanni Semeraro, Cataldo Musto

Preference ...

Learning Preference Models in Recommender Systems

Marco de Gemmis, Leo Iaquinta, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro

Department of Computer Science
University of Bari "Aldo Moro", Italy
{degemmis,iaquinta,lops,musto,narducci,semeraro}@di.uniba.it
http://www.di.uniba.it/

Abstract. As proved by the continuous growth of the number of web sites which embody recommender systems as a way of personalizing the experience of users with their content, recommender systems represent one of the most popular applications of principles and techniques coming from Information Filtering (IF). As IF techniques usually perform a progressive removal of non-relevant content according to the information stored in a user profile, recommendation algorithms process information about user interests – acquired in an explicit (e.g., letting users express their opinion about items) or implicit (e.g., studying some behavioral features) way – and exploit these data to generate a list of recommended items. Although each type of filtering method has its own weaknesses and strengths, preference handling is one of the core issues in the design of every recommender system: since these systems aim to guide users in a personalized way to interesting or useful objects in a large space of possible options, it is important for them to accurately capture and model user preferences.

The goal of this chapter is to provide a general overview of the approaches to learning preference models in the context of recommender systems. In the first part we will introduce general concepts and terminology of recommender systems, giving a brief analysis of advantages and drawbacks for each filtering approach. Then we will deal with the issue of learning preference models, show the most popular techniques for profile learning and preference elicitation and analyze methods for feedback gathering in recommender systems.

Key words: Recommender Systems, Preference Learning, Machine Learning

1 Introduction

How many times did you find a lot of unwanted mails opening your mailbox? How many times did you search something on the Web and you were not able to find what you were looking for?

The existence of a large quantity of information, in combination with the dynamic and heterogeneous nature of the Web, makes retrieval a hard task for the average user, who is usually overwhelmed by the abundant amount of information.

In this context (we usually refer to this as *Information Overload* problem), the role of user modeling and personalized information access is becoming crucial: although it is too soon to deeply understand the long-term effects of this surplus of information in our habits and in daily life, it is clear that users need a personalized support in sifting through large amounts of available information according to their interests and preferences.

Information Filtering systems, like Recommender Systems, relying on this idea, adapt their behavior to individual users by learning their tastes during the interaction, in order to construct a profile that can be later exploited to select relevant items. Nowadays these systems represent the main solution to the information overload problem, because they are able to gather and exploit heterogeneous information about users, emerging as one of the most useful tools to achieve a more intelligent information access.

In the workflow of a typical recommendation process, *learning user preferences* is a primary step: catching and modeling user interests in an effective way can be a key issue for personalization goals. Gathering user characteristics, acquired through an explicit (e.g., directly asking to the user) or implicit process (e.g., observing the user behavior), can produce a user model to be exploited to enable adaptivity mechanisms during the interaction with an information system.

The problem of recommending items has been studied extensively, and two main paradigms have emerged. *Content-based* recommendation systems try to recommend items similar to those a given user has liked in the past, whereas systems designed according to the *collaborative* recommendation paradigm identify users whose preferences are similar to those of the given user and recommend items they have liked [5]. Further, in the literature we found also other noteworthy paradigms: *demographic recommenders*, whose aim is to categorize the user starting from personal attributes making recommendation based on demographic classes; *knowledge-based* systems, which exploit knowledge about how a particular item meets a particular user need (such as case-based reasoning that solve a problem retrieving a past similar solved one [31]; *hybrid* systems, at last, combine different recommendation techniques trying to exploit their advantages and reducing at the same time their drawbacks. Each of above paradigms has particular methods to elicit user interests and preferences: most of them are related to machine learning area (probabilistic models, bayesian or neural networks, decision trees, association rules), but there are also some other techniques (so-called *heuristics*) which learn user profiles by exploiting preferences expressed by similar users (usually referred to as "neighbours") or processing textual contents describing the items liked. The goal of this chapter is to provide a general overview of the approaches to learning preference models in the context of recommender systems, by showing advantages and drawbacks of each technique and finally giving a brief analysis of the state of the art.

This chapter is organized as follows. Section 2 introduce general concepts and terminology about recommender systems. Preference learning issues in the area of recommender systems is presented in Section 3, where we also introduce the feedback gathering problem and some machine learning techniques used to acquire and infer user preferences. Conclusions are drawn in the last section.

2 Basics of Recommender Systems

Nowadays it is very important for people to be supported in their decisions, due to the exponential increase of available information.

Everyday we get advice from other people: “Hey, check out this Web site”, “I saw this book, you will like it”, “That restaurant is very good!”. When making a choice in the absence of decisive first-hand knowledge, choosing as other like-minded people have chosen in the past may be a good strategy. Recommender systems have the same role as human recommenders: they present information that they perceive to be useful and worth trying out.

These systems are used in several application domains to support users in taking decisions, to help them in managing the exponential increase of information and, in general, to provide a more *intelligent form of information access*.

The creation and management of personalized recommendations require mainly three distinct and important components: a user profile, an algorithm to update the profile given usage/input information, and an adaptive tool that exploits the profile in order to provide personalization.

First, the system needs to be able to store relevant information about users that will be used to infer their preferences and needs. Such information is stored in an individual user profile. Second, if the system has to adapt with the user over time, some mechanism is needed to keep the profile up-to-date. This could happen through explicit data input or implicit recording of user behavior as she interacts with the system, or a combination of them. Third, the system needs some way to exploit the current profile data in making recommendations to the user. The types of information stored in the profile will depend on the goals of the system and the algorithms it employs in order to provide recommendations. Different approaches to recommendation will require different pieces of information about the user, thus the profile structure will differ from system to system.

In this section we will provide a complete overview of the latter step, showing the main recommending approaches and explaining the benefits and weaknesses of each one, while in section sec:feedback we will analyze thoroughly the techniques used to acquire and infer user preferences through explicit or implicit feedbacks.

2.1 Content-based Recommender Systems

The core of the content-based approach is the processing of the contents describing the items to be recommended. The items can be very different depending on the number and type of attributes used to describe them. Each item can be described by the same small number of attributes with known sets of values, but this is not appropriate for items, such as Web pages, news or documents, described by means of unstructured text. In this case there are no attributes with well-defined values and the use of document modeling techniques with roots in information retrieval [47, 4] and information filtering [6] research is desirable.

A method to represent unstructured data is the Vector Space Model (VSM). The VSM [52] is a spatial representation of text documents. In this model, each document is represented by a vector in a n-dimensional space, where each dimension corresponds to a term from the overall vocabulary of a given document collection. Formally, every document is represented as a vector of term weights, where each weight indicates the degree of association between the document and the term.

The content-based approach can be applied only in the domains where we can provide some textual source describing the items: for example, text recommendation systems like the newsgroup filtering system *NewsWeeder* [27] uses the words of their texts as features. Otherwise, in the domain of movies, the attributes can be movie genre (comedy, horror, drama, etc), main actor and actress, producer, director, etc.

A content-based recommender learns a profile of the user interests based on the features present in the objects the user rated. For example, if a feature (e.g. Inter, or football) occurs in some news the user previously liked, we can expect that he will like other news where this feature often occurs. In this case, a text document may be recommended based on a comparison between the content of the document and the user profile. The system exploits the user profile to suggest relevant items by matching the profile representation against that of items to be recommended. The result of this matching is a binary or continuous relevance judgment, the latter case resulting in a ranked list of potentially interesting items. If data are represented by the VSM, the matching might be realized by computing the cosine similarity between the prototype vector and the item vectors. In some cases, the user is asked for feedback after the document has been shown to her. If the user likes the recommendation, the weights of the words extracted from the document are increased. This process is called *relevance feedback*.

The adoption of the content-based recommendation paradigm has several advantages when compared to the collaborative one:

- User independence - Content-based recommenders exploit solely ratings provided by the active user to build her own profile. Instead, collaborative filtering methods need ratings from other users in order to find the "nearest neighbors" of the active user, i.e., users that have similar tastes (rated the same items similarly). Then, only the items that are most liked by the neighbors of the active user would be recommended;
- Transparency - Explanations of recommended items can be provided by explicitly listing content features or descriptions that caused an item to be recommended. These features are indicators to consult in order to decide

when to trust a recommendation and when to doubt one. On the other hand, collaborative systems are black boxes since the only motivation for an item recommendation is that users with similar tastes liked that item;
- New item - Content-based recommenders are capable of recommending items not yet rated by any user. As a consequence, they do not suffer from the new user problem, which affects collaborative recommenders relying solely on users' preferences to make recommendations. Therefore, until the new item is rated by a substantial number of users, the system would not be able to recommend it.

On the other hand, content-based systems have several shortcomings:

Limited content analysis - Content-based techniques are limited by the features that are associated either automatically or manually with the objects that these systems recommend. No content-based recommendation system can provide good suggestions if the content does not contain enough information to distinguish items the user likes from items the user does not like. Some representations capture only certain aspects of the content, but there are many others that would influence a user's experience. For instance, there often is not enough information in the word frequency to model the user interests in jokes or poems, while techniques for affective computing would be most appropriate. Again, for Web pages, feature extraction by using techniques for text representation completely ignores aesthetic qualities and all multimedia information.

To sum up, both automatic extraction and manually assignment of features to items could not be sufficient to define the distinguishing aspects of items able to elicit the user interests.

Overspecialization - Content-based recommenders have no inherent method for finding something unexpected. The system recommends only items scoring highly against the user profile, hence the user is limited to being recommended items similar to those already rated. This drawback is also called *serendipity* problem. To give an example, when a user has only rated movies based on novels by Stephen King, she will be recommended just this kind of movies. A "perfect" content-based technique would never find anything *novel*, limiting the range of applications for which it would be useful.

New user - Enough ratings have to be collected before a content-based recommender system can really understand user preferences and provide accurate recommendations. Therefore, when few ratings are available, such as for a new user, the system would not be able to provide reliable recommendations.

2.2 Collaborative Recommender Systems

We can think of the Collaborative Filtering (CF) paradigm as a computerized process of *word of mouth*. For instance, when looking for a restaurant we usually rely on friend advices, or when looking for a book to read we ask friends who have the same taste. Similarly in collaborative recommender systems *user opinions* are used to choose what items the user likes. In CF systems recommendations are based on evaluations of users who share similar interests among them. The idea behind these systems is that a set of users which liked the same items in the past probably share the same preferences. Thus, picking a user from this set, we can suggest her all the unseen items which other users with similar tastes showed to like in the past.

Opinions on items can be expressed as explicit user ratings on some scale ranging from bad to good, or as implicit ratings given by logging user actions. As an example of the latter, viewing or skipping items could be interpreted as positive and negative ratings respectively.

CF systems analyze opinions of other users on items, thus they provide a liking degree not based on the nature of the item, but on human judgment. Because of this characteristic, CF systems are generally perceived to be more useful than IF based systems [20].

The main advantage of collaborative methods is that items in different product categories can be recommended. Movies, images, art and text items are all represented by opinions of users and thus they can be recommended by the same system.

In collaborative filtering, a user profile simply consists of the data the user has specified. These data are compared to those of other users to find overlaps in interests among users. For example, the nearest neighbor approach, used in some collaborative recommender system [30], represents the preferences by the items rated (or purchased) by the user. The profile is represented by the user-item matrix [35] where for each cell (u,i) we have the rating of the user u on the item i. Thus, the recommender algorithm exploits the matrix to identify for each user the set of nearest neighbors. In this case the recommender algorithm performs three tasks: it finds similar users, creates the nearest neighbors set for each user, infers the like degree for an unseen item based on the nearest neighbors behavior. For example, in an e-commerce scenario when the user u puts the 'Shining' movie into her basket, the system recommends her the 'Silence of the lambs' book because more users (who share similar tastes with her, i.e. the nearest neighbors of u) purchased them together.

Terveen and Hill [60] claim three essentials are needed to support collaborative filtering: many people must participate (increasing the likelihood that any one person will find other users with similar preferences), there must be an easy way to represent the user interests in the system, and the algorithms must be able to match people with similar interests. These three elements are not that easy to develop, and produce the main shortcoming of collaborative filtering systems. Following the main limitations of collaborative systems [5, 28].

New user problem - In order to make accurate recommendations, the system must first learn the preferences of the user from her ratings. Several techniques have been proposed to address this problem. Most of them use a hybrid recommendation approach, which combines content-based and collaborative techniques.

New item problem (early rater) - New items are added regularly to recommender systems. Collaborative systems rely only on users preferences to make recommendations. Therefore, until the new item is rated by a substantial number of users, the recommender system would not be able to recommend it. As extreme case of the early rater problem, when a collaborative filtering system first begins, every user suffers from the early rater problem for every item. This problem can also be addressed using hybrid recommendation approaches.

Sparsity problem - In any recommender system, one of the biggest problem to find recommendations is the extreme sparsity of data in the database. The number of ratings obtained is usually very small compared to the number of ratings to be predicted. Effective prediction of ratings from a small number of examples is important. Also, the success of the collaborative recommender system depends on the availability of a critical mass of users. For example, in a movie recommendation system there might be many movies that have been rated only by few people and these movies would be recommended very rarely, even if those few users gave high ratings to them. One way to overcome the problem of rating sparsity is to use user profile information when calculating user similarity. That is, two users could be considered similar not only if they similarly rated the same items, but also if they belong to the same demographic segment. For example, Pazzani uses gender, age, area code, education, and employment information of users in the restaurant recommendation application [40]. This extension of traditional collaborative filtering techniques is sometimes called demographic filtering.

Grey sheep problem (unusual user) - In a small or even medium community of users, there are individuals who would not benefit from pure collaborative filtering systems because their opinions do not consistently agree or disagree with any group of people. These individuals will rarely, if ever, receive accurate predictions, even after the initial start up phase for the user and the system [14].

The majority of users falls into the class of the so-called white sheep, those who have high correlation with many other users and who will therefore, in theory, be easy to find recommendations for. The opposite type of people are the black sheep, those for whom there are no or few people who they correlate with. This makes it very difficult to make recommendations for them. On the positive side, for statistical reasons, as the number of users of a system increases the chance of finding other people with similar tastes increases and so better recommendations can be provided.

Scalability problem - Collaborative filtering systems require a lot of computational resources with the increasing number of users and items. Collaborative filtering systems require data from a large number of users before being effective as well as requiring a large amount of data from each user. The critical dependency on the size and composition of the user population also influences a users group of nearest neighbors. In a situation in which feedback fails to cause this group of nearest neighbors to change, expressing dislike for an item will not necessarily prevent the user from receiving similar items in the future. Furthermore, the lack of access to the content of items prevents similar users from being matched unless they have rated the exact same items.

2.3 Demographic Recommender Systems

These systems aim to categorize the user starting from personal attributes making recommendation based on demographic classes. *Grundy* [49], for example, recommends books by gathering personal information through an interactive dialogue matching users responses against a library of manually assembled user stereotypes. *LifeStyle Finder* [26] tries to identify to which cluster a user belongs tailoring recommendations exploiting preferences of the other users in the cluster. Pazzani [40] uses machine learning techniques to obtain a classifier based on demographic data. The representation of demographic information in a user model can vary greatly. *Grundy* system uses hand-crafted attributes with numeric confidence values, while Pazzani extracts features from users' home pages.

The benefit of a demographic approach is that it may not require a history of user ratings of the type needed by collaborative and content-based techniques. However, up to our knowledge, there are not many recommender systems using demographic data because this form of information is difficult to collect: till some years ago, indeed, users were reluctant to share a big amount of personal information with a system. Nowadays with the exponential growth of social network and the continuous expansion of Web 2.0 platforms like Flickr and YouTube, the situation is changed: people's point of view is evolving towards a more open perspective, with users more trustful to sharing of information. Despite this, still today demographic approaches notice less success than others.

2.4 Knowledge-based Recommender Systems

These systems uses a knowledge-based approach to generate recommendations.

All recommendation techniques make some kind of inference. Knowledge-based approaches are distinguished in that they have functional knowledge: they have knowledge about how a particular item meets a particular user need, and can therefore reason about the relationship between a need and a possible recommendation [12]. The user profile can be any knowledge structure that supports this inference. In the simplest case, as in Google, it may simply be the query that the user has formulated. In others, it may be a more detailed representation of the user needs [61].

A particular kind of knowledge-based recommender systems implement case-based reasoning (CBR). This recommender solves a new problem by retrieving a known solution to a similar problem. In [31], four main steps of a CBR recommender are identified: *retrieve, reuse, adaptation*, and *retain*. The first step looks in the knowledge-base for a case similar to the new problem, then reuse the retrieved solution (making some adaptation, if necessary). Finally the new adapted case is stored in the case-library. In this system there is not a user preference elicitation because the main task of the recommendation algorithm is to retrieve the case most similar to the problem to solve. From the point of view of the system, the search for a product to recommend is similar to diagnose a disease. In the first case the system retrieves a product with particular requirements, in the latter case it retrieves a disease with particular symptoms.

A knowledge-based recommender system avoids some of the drawbacks of other recommendation techniques. It does not have a *ramp-up* problem (early rater problem and the sparse ratings problem) since its recommendations do not depend on a base of user ratings. As stated above, it does not have to gather information about a particular user because system judgments are independent of individual tastes. These characteristics make knowledge-based recommenders not only valuable systems on their own, but also highly complementary to other types of recommender systems [11].

2.5 Hybrid Recommender Systems

They combine two or more recommender algorithms (the more frequent approach is to combine collaborative filtering with content-based filtering) in order to emphasize their strengths and to level out their corresponding weaknesses.

Robin Burke proposed a very analytical classification of hybrid systems [12], listing a number of hybridization methods to combine pairs of recommender algorithms.

- Weighted - In weighted hybrid recommenders the score (or votes) of a recommended item is computed from the results of all of the available recommendation techniques present in the system. This means that the scores of several recommendation techniques are combined together to produce a single recommendation. The simplest combined hybrid would be a linear combination of recommendation scores.
- Switching - A switching hybrid uses some criterion to switch between recommendation techniques. Switching hybrids introduce additional complexity into the recommendation process since the switching criteria must be determined, and this introduces another level of parameterization.
- Mixed - Recommendations from several different recommenders are presented at the same time. This may be possible where it is practical to make large number of recommendations simultaneously.
- Feature Combination - Features from different recommendation sources are thrown together into a single recommendation algorithm. For example content and collaborative techniques might be merged treating collaborative information as simply additional feature data associated with each example and using content-based techniques over this augmented data set.

– Cascade - The cascade hybrid involves a staged process because one recommender refines the recommendations given by another one. This means that, one recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation from among the candidate set.
– Feature Augmentation - Output from one technique is used as an input feature to another. This means that one technique is employed to produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique.
– Meta-level - The model learned by one recommender is used as input to another. This differs from feature augmentation: in an augmentation hybrid, we use a learned model to generate features for input to a second algorithm; in a meta-level hybrid, the entire model becomes the input.

In order to complete the survey we should also mention some hybrid recommender systems combining collaborative and content-based methods, such as *Fab* [5], *WebWatcher* [22], *P-Tango* [14], *ProfBuilder*, [62], *PTV* [58], *Content-boosted Collaborative Filtering* [32] and *CinemaScreen* [51].

3 Learning User Preferences in Recommender Systems

Before entering in technical details concerning methods for preference acquisition and especially techniques for learning user profiles, we need to take a step backward to thoroughly analyze other important issues: *what are preferences?*

As stated by [10], a *preference* is an ordering relation between two or more items that lets us to characterize which, among a set of possible choices, is the one that best fits our tastes. *Preferences* are something able to guide our choices, discriminating items we like from those we don't like (or we like the least). In other terms, learning user preferences is a way to find the solution of a research (or optimization, in some cases) problem whose space of possible solutions is represented by the set of the items the user can enjoy (namely, in recommender systems, the set of items that can be recommended). Although the semantics of the concept of preference is pretty clear, acquiring user preferences and working with them is a more difficult task. Indeed, the complexity of the problem of preference learning is strictly related to the number of dimensions we used to represent the set of possible choices. We can think at a simple example: choosing a mobile phone. If the only feature to consider is its price, ordering the set of phones and suggesting user the preferred one (namely, the cheaper one) is a simple task. However, if we also add only one more feature (e.g. camera zoom) ordering process becomes more complex and hardly to manage by users in an effective way. When we choose a restaurant, in the same way, we need to find a trade-off between a lot of aspects like price, service, distance, available time, quality of food and so on.

So, in order to generate a user profile we need to *gather user feedback* in order to catch information about user prefences and *model* them using a specific representation. Next, this information can be processed (e.g. through machine learning-related approaches) in order to *learn user profiles* to be exploited in the recommendation process.

3.1 Feedback Gathering

The information filtering and information retrieval systems rely on relevance feedback (RF) to capture an appropriate snapshot of user information needs in order to allow the user to directly express her notion of relevance with respect to individual documents [6]. RF has been employed in several classes of personalization systems. Driven by the need for better representation of information needs, RF was initially introduced to support basic query expansion [50]. However, its success in inferring the user's notion of relevance on a per-document basis has lead to a subsequent adoption by information filtering and recommendation systems. RF approaches are based on a feedback gathering scheme, either explicit or implicit. In the former, object ratings of predefined scale are provided explicitly by users, while implicit feedback gathering techniques infer object relevance in a transparent fashion, by monitoring user interaction with the system.

Explicit Ratings. The use of explicit ratings is common in everyday life; ranging from grading students' work to assessing competing consumer goods (see Alton-Scheidl et al. [3] for a review). Although some forms of rating are made in free text form (e.g. book reviews), it is frequently the case that ratings are made on an agreed discrete scale (e.g. star ratings for restaurants, marks out of ten for films, etc). Ratings made on these scales allow these judgments to be processed statistically to provide averages, ranges, distributions, etc.

Several online systems have adopted the explicit ratings approach. For instance, Grouplens [47] provides the collaborative filtering of Internet news. Grouplens users rate articles after having read them and the system aggregates ratings and analyses for future use. MovieLens [34] follows a similar technique to provide movie recommendation services to their members by creating a user profile based on subjective rating of films. Since both these systems originate directly from user explicit judgments, they lead to an accurate estimation of information requirements.

A central feature of explicit ratings is that the evaluator has to examine an item and assign it a value on the rating scale. This imposes a cognitive cost on the evaluator to assess the performance of an object [38]. Indeed, the act of rating alters the user behavior from her normal interaction pattern and, consequently, even less noticeable explicit feedback approaches are considered expensive. Since the results may not become immediately apparent, users tend to skip the rating task [19].

Also, explicit RF techniques disregard user knowledge on the current topic. Users are often unclear about their search interests. They browse for more information to clarify their need and re-formulate their query accordingly. The uncertainty in their search episodes increases the cognitive load during explicit

RF, as users must decide on the relevance of a document possibly with a lack of confidence.

Finally, the use of explicit ratings imposes privacy issues that have to be resolved [23]. Irrespective of the underlying reason, users are not always comfortable in providing direct indications of their interests. Due to the obtrusive nature of explicit ratings, not many users are willing to provide them. Hence, the performance of profile capturing and recommendation algorithms of such systems degrades, due to the dearth of ratings. In social filtering systems based on explicit feedback gathering policies, the sparsity of RF judgments can often render such systems unusable, since there are few previous assessments to learn from.

Explicit RF can relying also on critiquing examples. For instance, Smart-Client [44] is a tool for planning travel arrangements. Users are required to criticize examples of possible solutions. For instance, "the arrival time of this flight leg is too late." The interaction is cyclical: (1) the system provides example solutions, (2) the user examines any of them and may state a critique on any aspect of it, (3) the critique becomes an additional preference in the model, and (4) the system refines the solution set. Ricci and Nguyen [48] propose a similar critiquing interaction to provide recommendations of restaurants in a mobile context.

As discussed in Pu and Chen [43], the motivation for this methodology is that people usually cannot state preferences in advance but construct their preferences as they see the available options. However, because the critiques come from the user in response to the shown examples, the current solutions can hinder the user from refocusing the search in another direction (the anchoring effect). A complete preference model can be acquired only if the system is able to stimulate the user by showing diverse examples.

Implicit Ratings. Implicit RF gathering techniques are proposed as unobtrusive alternative or supplement to explicit ratings in order to state (indirect) assessment about usefulness of any individual item. Such techniques passively monitor user interactions with the system in order to estimate user interests [37]. Click-throughs, time spent viewing a document and mouse gestures are among the possible sources of implicit feedback [24]. The main benefits of implicit feedback, over explicit ratings, are that they remove the cognitive cost of providing relevance judgments explicitly and they can be gathered in large quantities and aggregated to infer item relevance. Since implicit judgments are derived transparently, they contain less indicative value than explicit ratings. Although the accuracy of implicit approaches has been questioned [38], recent studies have shown that they can be effectively adopted to state relevance feedback [63].

There are several types of feedback that can be implicitly captured. For instance, whether a message was read or ignored, whether it was saved or deleted, and whether or not a follow up message was posted are utilized as an implicit feedback source in conjunction with explicit rating by InfoScope [59] to filter Internet discussion groups. Monitoring the reading time of a document could also be used. Morita & Shinoda [37] concluded that the time spent reading documents on the web is closely related to the degree it suits the needs of each user. An alternative measure of implicit feedback is to assume that all printed documents are relevant and therefore try to detect the user profile from this kind of behavior [25].

Nichols [38] presented a list of potential types of user behaviors that could be exploited as sources for implicit feedback. Kelly & Teevan [24] extended a classification of observable feedback behaviors according to two axes, *Behavior Category*[1] and *Minimum Scope*[2] to categorize actions that can be observed during user information seeking episodes. Their work has also focused on classifying existing scientific literature on implicit feedback according to Behavior Category and Minimum Scope. Unsurprising, a lot of analyzed works concerns examination with object scope, i.e. click-through or scrolling measures are largely investigated and exhibit a strong positive correlation with the explicit ratings. Such data can be easily captured in realtime at no considerable computational cost, while user behaviors that fall in the "Reference", "Annotate" and "Create" require a more precise control over individual services and applications and, thus, are hard to capture and benefits for estimating user interests are not fully clear.

3.2 Modeling User Preferences

Feedback gathering techniques let us collect as much information is possible about user tastes and interests. However, before this information can be exploited as input to learn preferences models, this data need to be modeled following a specific representation. Techniques for modeling information (we usually refer to this as *items*, in recommender systems) can be split depending on the kind of data which will be stored in the user profile. If we have to handle unstructured data (the ones usually exploited by content-based recommenders) it is necessary to process them through some information retrieval-related techniques (such as stemming, lemmatization, indexing, and so on) which allow us to shift from a textual source to a structured one. For structured data, like generic ratings or some well-defined attribute-value pairs (e.g. demographic data), instead, it is possible to represent them through a matrix, as it usually happens in collaborative recommender systems. In both cases all the information provided by the user, apart from their nature, can be also represented in a more complex way (semantic or neural networks, probabilistic models, etc.) so that we can use them as input for learning user profiles.

In the next section we will focus our attention on machine learning techniques showing how we can learn an user profile and adapt it gathering user feedback on recommended items. We will complete the analysis trying to give also a complete overview of the state of the art in this area, showing how each approach was implemented in a real recommender system.

---

[1] The Behavior Category (Examine, Retain, Reference, Annotate and Create), refers to the underlying purpose of the observed behavior.

[2] Minimum Scope (Segment, Object and Class), refers to the smallest possible scope of the item being acted upon.

3.3 Techniques for Learning User Profiles

Most systems learn user profiles using an online learning approach, building and updating the model in order to make recommendations in real-time. Offline learning methods, instead, fit better in domains where, as stated in [35], user preferences change slowly with respect to the time needed to build the model.

The application of machine learning techniques is a typical way to fulfil the task of learning user profiles in model-based recommender systems. A common approach is to learn the user profile by building a *classifier*. In [35] a classifier is defined as a model able to assign a category to a specific input.

In the machine learning approach to categorization, an inductive process automatically builds a classifier by learning from a *training set* (items labeled with the categories they belong to) the features of the categories. In this approach the problem of learning user profiles is considered as a binary categorization task: each item has to be classified as interesting or not with respect to the user preferences. Therefore, the set of categories is $C = \{c_+, c_-\}$, where c_+ is the positive class (user-likes) and c_- the negative one (user-dislikes).

Classifiers may be implemented using many different machine learning strategies including probabilistic approaches, neural networks, decision trees, association rules and Bayesian networks. In this section we will provide a general overview of these techniques.

Naïve Bayes. It is the most used probabilistic algorithm and belongs to the general class of Bayesian classifiers.

These approaches generate a probabilistic model based on previously observed data. It is usually used in content-based recommender systems where the items to recommend are represented by textual documents. Thus, the model estimates the *a posteriori* probability, $P(c|d)$, of document d belonging to class c. This estimation is based on the a priori probability, $P(c)$, the probability of observing a document in class c, $P(d|c)$, the probability of observing the document d given c and, $P(d)$, the probability of observing the instance d. Using these probabilities, the Bayes theorem is applied to calculate $P(c|d)$:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \tag{1}$$

To classify the document d, the class with the highest probability is chosen:

$$c = argmax_{c_j} \frac{P(c_j)P(d|c_j)}{P(d)}$$

$P(d)$ is generally removed as it is equal for all c_j. As we do not know the value for $P(d|c)$ and $P(c)$, we estimate them by observing the training data. However, estimating $P(d|c)$ in this way is problematic, as it is very unlikely to see the same document more than once: the observed data is generally not enough to be able to generate good probabilities. The naïve Bayes classifier overcomes this problem by simplifying the model by making the independence assumption: all the words or tokens in the observed document d are conditionally independent of each other given the class. Individual probabilities for the words in a document are estimated one by one rather than the complete document as a whole. The conditional independence assumption is clearly violated in real-world data, however, despite these violations, empirically the naïve Bayes classifier does a good job of classifying text documents [29, 7].

Although naïve Bayes classifiers are not as good as probability estimators, it has been shown that they can perform surprisingly well in the classification tasks where the computed probability is not important [18]. Another advantage of the naïve Bayes approach is that it is very efficient and easy to implement compared to other learning methods.

The naïve Bayes classifier has been used in several content-based recommendation systems, such as *Syskill & Webert* [42, 39], *NewsDude* [8], *Daily Learner* [9], *LIBRA* [36] and *ITR* [17, 56].

Rocchio's method. Some linear classifiers consist of an explicit profile (or prototypical document) of the category [55]. The Rocchio's method is used for inducing linear, profile-style classifiers. It relies on an adaptation to text categorization of the well-known Rocchio's formula for relevance feedback in the VSM [50].

This algorithm represents documents as vectors so that documents with similar content have similar vectors. Each component of such a vector corresponds to a term in the document, typically a word. The weight of each component is computed using the TF-IDF [53] term weighting scheme. Learning is achieved by combining document vectors (of positive and negative examples) into a prototype vector for each class in the set of classes C. To classify a new document d, the similarity between the prototype vectors and the corresponding document vector representing d are calculated for each class (for example by using the cosine similarity measure), then d is assigned to the class with which its document vector has the highest similarity value. More formally, Rocchio's method computes a classifier $\overrightarrow{c_i} = \langle \omega_{1i}, \ldots, \omega_{|T|i} \rangle$ for category c_i (T is the *vocabulary*, that is the set of distinct terms in the training set) by means of the formula:

$$\omega_{ki} = \beta \cdot \sum_{\{d_j \in POS_i\}} \frac{\omega_{kj}}{|POS_i|} - \gamma \cdot \sum_{\{d_j \in NEG_i\}} \frac{\omega_{kj}}{|NEG_i|} \tag{2}$$

where ω_{kj} is the TF-IDF weight of the term t_k in document d_j, POS_i and NEG_i are the set of positive and negative examples in the training set for the specific class c_j, β and γ are control parameters that allow setting the relative importance of *all* positive and negative examples.

To assign a class $\widetilde{c}$ to a document d_j, the similarity between each prototype vector $\overrightarrow{c_i}$ and the document vector $\overrightarrow{d_j}$ is computed and $\widetilde{c}$ will be the c_i with the highest value of similarity. Relevance feedback has been used in several content-based recommendation systems, such as *YourNews* [2], *Fab* [5] and *NewT* [57].

Decision trees learners. Decision trees are trees in which internal nodes are labeled by terms, branches departing from them are labeled by tests on the weight that the term has in the test document, and leafs are labeled by categories. Decision trees are built by recursively partitioning training data, that is text documents, into subgroups, until those subgroups contain only instances of a single class. The test for partitioning data is run on the weights that the terms labeling the internal nodes have in the document. The choice of the term on which to operate the partition is generally made according to an information gain or entropy criterion [64]. Decision trees are used in the *Syskill & Webert* [42, 39] recommender system. The most widely-used decision tree learner applied to profiling is ID3 [45].

Decision rule classifiers. Are similar to decision trees, because they operates in a similar way to the recursive data partitioning approach described above. An advantage of rule learners is that they tend to generate more compact classifiers than decision trees learners. Rule learning methods usually attempt to select from all the possible covering rules (i.e. rules that correctly classify all the training examples) the "best" one according to some minimality criterion. Some examples of inductive learning techniques are Ripper [15], Slipper [16], CN2 [13] and C4.5rules [46].

Neural networks. As the approaches seen above, neural networks has a training phase to learn the user profile. These networks model complex relationships between input and output cells. The user interests are represented by the output cells and each of them are achievable by a specific pattern in the network. When an error occurs, there is a backward propagation until the responsible cell is achieved, so the cell parameters are adjusted. Jennings and Higuchi employed a neural network for constructing a users profile [21].

Bayesian network. It represents a probability distribution by a direct acyclic graph. There are random variables (nodes) and relations among them (arcs). The nodes represent attributes and the arcs represent probability correlations. In [54] a method integrating Case Based Reasoning and Bayesian Network for the user profiling task is shown. Bayesian Network is employed to model quantitative and qualitative relationships between items that users have liked. Bayesian Network is generally used in those situations where user interests change slowly.

Nearest neighbor algorithms. These algorithms, also called lazy learners, simply store training data in memory, and classify a new unseen item by comparing it to all stored items by using a similarity function. The "nearest neighbor" or the "k-nearest neighbors" items are determined, and the class label for the unclassified item is derived from the class labels of the nearest neighbors. A similarity function is needed, for example the cosine similarity measure is adopted when items are represented using the VSM. Nearest neighbor algorithms are quite effective, albeit the most important drawback is their inefficiency at classification time, since they do not have a true training phase and thus defer all the computation to classification time. *Daily Learner* [9] and *Quickstep* [33] use the nearest neighbor algorithm to create a model of the user short term interest and for associating semantic annotation of papers with class names within the ontology, respectively.

4 Conclusions

In this chapter we surveyed the issue of learning user preferences in recommender systems area. Firstly, we introduced the topic of recommender systems by showing the main approaches presented in literature: we described the content-based and collaborative approaches, showing also the features of some other models such as demographic, knowledge-based and hybrid ones. Although each kind of recommender system has its own weaknesses and strengths, all of them are joint by a common goal: filtering a set of items, identifying which ones the user will like more by exploiting the information stored in her profile.

In the second part of the chapter we shift our attention on the core of the recommendation process investigating the issues of learning and modeling user preferences. Several manners to gather user information are exposed, in particular we focus on techniques to get user feedback in implicit and explicit way. Thus, techniques for learning user profiles are analyzed. In particular we presented machine learning methods such as Naïve Bayes, Rocchio's method, etc.

We hope that the survey presented in this chapter will contribute to stimulate the research community about the next generation of recommendation technologies and can provide the basis for researches toward new methods for user preferences gathering and modeling in recommender system area.

References

1. G. Adomavicius and A. Tuzhilin. 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. J. Ahn, P. Brusilovsky, J. Grady, D. He, and S. Yeon Syn. Open User Profiles for Adaptive News Systems: Help or Harm? In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web*, pages 11–20. ACM, 2007.
3. R. Alton-Scheidl, R. Schumutzer, P.P. Sint, and G. Tscherteu. *Voting and rating in Web4Groups*, pages 13–103. R. Oldenbourg, 1997.
4. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
5. M. Balabanovic and Y. Shoham. Fab: Content-based, Collaborative Recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
6. N. Belkin and B. Croft. Information filtering and information retrieval:two sides of the same coin? *Communications of the ACM*, 35(12):29–37, 1992.

7. D. Billsus and M. Pazzani. Learning Probabilistic User Models. In *Proceedings of the Workshop on Machine Learning for User Modeling*, Chia Laguna, IT, 1997.
8. D. Billsus and M. J. Pazzani. A Hybrid User Model for News Story Classification. In *Proceedings of the Seventh International Conference on User Modeling.Banff, Canada*, 1999.
9. D. Billsus and M. J. Pazzani. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180, 2000.
10. R. Brafman and C. Domshlak. Preference handling - an introductory tutorial. *AI Magazine*, 30(1):58–86, 2009.
11. R. Burke. Knowledge-based Recommender Systems. In *A. Kent (ed.), Encyclopedia of Library and Information Systems*, volume 69(32). 2000.
12. R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
13. P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
14. M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining Content-Based and Collaborative Filters in an Online Newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.
15. W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 115–123, 1995.
16. W. W. Cohen and Y. Singer. A simple, fast, and efficitive rule learner. In *AAAI/IAAI*, pages 335–342, 1999.
17. M. Degemmis, P. Lops, and G. Semeraro. A Content-collaborative Recommender that Exploits WordNet-based User Profiles for Neighborhood Formation. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research (UMUAI)*, 17(3):217–255, 2007. Springer Science + Business Media B.V.
18. P. Domingos and M. J. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3):103–130, 1997.
19. J. Grudin. Groupware and social dynamics: eight challenges for developers. *Commun. ACM*, 37(1):92–105, 1994.
20. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Theoretical Models, pages 230–237, 1999.
21. A. Jennings and H. Higuchi. A user model neural network for a personal news service. *User Model. User-Adapt. Interact.*, 3(1):1–25, 1993.
22. T. Joachims, D. Freitag, and T. M. Mitchell. Web Watcher: A Tour Guide for the World Wide Web. In *15th International Joint Conference on Artificial Intelligence*, pages 770–777, 1997.
23. K. Keenoy and M. Levene. Personalisation of web search. In Bamshad Mobasher and Sarabjot S. Anand, editors, *Intelligent Techniques for Web Personalization*, volume 3169 of *Lecture Notes in Computer Science*, pages 201–228. Springer, 2003.
24. D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.
25. J. Kim, D. W. Oard, and K. Romanik. Using implicit feedback for user modeling in internet and intranet searching. Technical report, University of Maryland at College Park, 2000.
26. B. Krulwich. Lifestyle finder: Intelligent user profiling using large-scale demographic data. *Artificial Intelligence Magazine*, 18(2):37–45, 1997.

27. K. Lang. Newsweeder: Learning to filter news. In *Proceedings of the 12th International Conference on Machine Learning, Lake Tahoe, CA*, pages 331–3¡39, 1995.
28. W.S. Lee. Collaborative learning for recommender systems. In *Procedings of the International Conference on Machine Learning*, 2001.
29. D. D. Lewis and M. Ringuette. A Comparison of Two Learning Algorithms for Text Categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.
30. G. Linden, B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
31. F. Lorenzi and F. Ricci. Case-based recommender systems: A unifying view. In Bamshad Mobasher and Sarabjot S. Anand, editors, *ITWP*, volume 3169 of *Lecture Notes in Computer Science*, pages 89–113. Springer, 2003.
32. P. Melville, R. J. Mooney, and R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-02)*, pages 187–192, Menlo Parc, CA, USA, 2002. AAAI Press.
33. S. E. Middleton, N. R. Shadbolt, and D. C. De Roure. Ontological User Profiling in Recommender Systems. *ACM Transactions on Information Systems*, 22(1):54–88, 2004.
34. B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM.
35. M. Montaner, B. Lopez, and J. L. De La Rosa. A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, 19(4):285–330, 2003.
36. R. J. Mooney and L. Roy. Content-Based Book Recommending Using Learning for Text Categorization. In *Proceedings of the 5th ACM Conference on Digital Libraries*, pages 195–204, San Antonio, US, 2000. ACM Press, New York, US.
37. M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 272–281, New York, NY, USA, 1994.
38. D. M. Nichols. Implicit rating and filtering. In *Proceedings of Fifth DELOS Workshop on Filtering and Collaborative Filtering*, pages 31–36. ERCIM, 1997.
39. M. Pazzani and D. Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3):313–331, 1997.
40. M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.
41. M. J. Pazzani and D. Billsus. Content-Based Recommendation Systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341, 2007. ISBN 978-3-540-72078-2.
42. M. J. Pazzani, J. Muramatsu, and D. Billsus. Syskill and Webert: Identifying Interesting Web Sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 54–61, Menlo Park, 1996. AAAI Press / MIT Press.
43. P. Pu and L. Chen. User-involved preference elicitation for product search and recommender systems. *AI Magazine*, 29(4):93–103, 2008.

44. P. Pu and B. Faltings. Enriching buyers' experiences: the smartclient approach. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 289–296, New York, NY, USA, 2000. ACM.
45. J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine Learning. An Artificial Intelligence Approach*, pages 463–482, 1983.
46. J. R. Quinlan. The minimum description length principle and categorical theories. In *ICML*, pages 233–241, 1994.
47. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.
48. F. Ricci and Q. N. Nguyen. Acquiring and revising preferences in a critique-based mobile recommender system. *IEEE Intelligent Systems*, 22(3):22–29, 2007.
49. E. Rich. User Modeling via Stereotypes. *Cognitive Science*, 3:329–354, 1979.
50. J. J. Rocchio. *Relevance Feedback in Information Retrieval.* Prentice Hall, Englewood, Cliffs, New Jersey, 1971.
51. J. Salter and N. Antonoupoulos. CinemaScreen Recommender Agent: Combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21(1):35–41, 2006.
52. G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.
53. Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. Technical report, 1988.
54. S. N. Schiaffino and A. Amandi. User profiling with case-based reasoning and bayesian networks. In *IBERAMIA-SBIA 2000 Open Discussion Track*, pages 12–21, 2000.
55. F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
56. G. Semeraro, P. Basile, M. de Gemmis, and P. Lops. User Profiles for Personalizing Digital Libraries. In Yin-Leng Theng, Schubert Foo, Dion Goh Hoe Lian, and Jin-Cheon Na, editors, *Handbook of Research on Digital Libraries: Design, Development and Impact*, pages 149–158. IGI Global, 2009. ISBN 978-159904879-6.
57. B. Sheth and P. Maes. Evolving Agents for Personalized Information Filtering. In *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*, pages 345–352. IEEE Computer Society Press, 1993.
58. B. Smith and P. Cotter. A Personalized TV Listings Service for the Digital TV Age. *Knowledge-Based Systems*, 13:53–59, 2000.
59. F. C. Stevens. *Knowledge-based assistance for accessing large, poorly structured information spaces.* PhD thesis, Boulder, CO, USA, 1993.
60. L. Terveen and W. Hill. *Human-Computer Collaboration in Recommender Systems.* In J. Carroll (Ed.), HCI on the new Millennium: Addison Wesley, 2001.
61. B. Towle and C. Quinn. Knowledge based recommender systems using explicit user models. In *Papers from the AAAI Workshop, AAAI Technical Report WS-00-04*, pages 74–77. Menlo Park, CA: AAAI Press, 2000.
62. A. M. Wasfi. Collecting User Access Patterns for Building User Profiles and Collaborative Filtering. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 57–64, 1999.
63. R. White, I. Ruthven, and J. M. Jose. The use of implicit evidence for relevance feedback in web retrieval. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research*, pages 93–109, London, UK, 2002. Springer-Verlag.

64. Y. Yang and J. O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.