PANIMALAR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF CSE

R-2017

ACADEMIC YEAR	:2023-2024
BATCH	:2020-2024
YEAR/SEM	:IV/VIII
SUBJECTCODE/TITLE	:CS8080/INFORMATION RETRIEVAL TECHNIQUES

UNIT-5

RECOMMENDER SYSTEM

UNIT V
RECOMMENDER SYSTEM

Recommender Systems Functions – Data and Knowledge Sources – Recommendation Techniques – Basics of Content-based Recommender Systems – High Level Architecture – Advantages and Drawbacks of Content-based Filtering – Collaborative Filtering – Matrix factorization models – Neighborhood models.

RECOMMENDER SYSTEMS

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read.

"Item" is the general term used to denote what the system recommends to users. A RS normally focuses on a specific type of item (e.g., CDs, or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item.

RSs are primarily directed towards individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternative items that a Web site

In their simplest form, personalized recommendations are offered as ranked lists of items. In performing this ranking, RSs try to predict what the most suitable products or services are, based on the user's preferences and constraints. In order to complete such a computational task, RSs collect from users their preferences, which are either explicitly expressed, e.g., as ratings for products, or are inferred by interpreting user actions. For instance, a RS may consider the

navigation to a particular product page as an implicit sign of preference for the items shown on that page.

Depending on the recommendation approach, by the user's context and need, RSs generate recommendations using various types of knowledge and data about users, the available items, and previous transactions stored in customized databases. The user can then browse the recommendations. She may accept them or not and may provide, immediately or at a next stage, an implicit or explicit feedback. All these user actions and feedbacks can be stored in the recommender database and may be used for generating new recommendations in the next user-system interactions.

Recommender systems emerged as an independent research area in themid-1990s. In recent years, the interest in recommender systems has dramatically increased, as the following facts indicate:
Recommender systems play an important role in such highly rated Internet sites as Amazon.com, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, and IMDb. Moreover many media companies are now developing and deploying RSs as part of the services they provide to their subscribers.

RECOMMENDER SYSTEMS FUNCTION

In the previous section we defined RSs as software tools and techniques providing users with suggestions for items a user may wish to utilize. Now we want to refine this definition illustrating a range of possible roles that a RS can play. First of all, we must distinguish between the role played by the RS on behalf of the service provider from that of the user of the RS. For instance, a travel recommender system is typically introduced by a travel intermediary (e.g., Expedia.com) or a destination management organization (e.g., Visitfinland.com) to increase its turnover (Expedia), i.e., sell more hotel rooms, or to increase the number of tourists to the destination [86]. Whereas, the user's primary motivations for accessing the

two systems is to find a suitable hotel and interesting events/attractions when visiting a destination.

In fact, there are various reasons as to why service providers may want to exploit this technology:

- Increase the number of items sold.
- Sell more diverse items.
- Increase the user satisfaction.
- Increase user fidelity.
- Better understand what the user wants.

DATA AND KNOWLEDGE SOURCES

RSs are information processing systems that actively gather various kinds of data in order to build their recommendations. Data is primarily about the items to suggest and the users who will receive these recommendations. But, since the data and knowledge sources available for recommender systems can be very diverse, ultimately, whether they can be exploited or not depends on the recommendation technique.

In general, there are recommendation techniques that are knowledge poor, i.e., they use very simple and basic data, such as user ratings/evaluations for items. Other techniques are much more knowledge dependent, e.g., using ontological descriptions of the users or the items, or constraints, or social relations and activities of the users. In any case, as a general classification, data used by RSs refers to three kinds of objects: items, users, and transactions, i.e., relations between users and items.

Items.

Items are the objects that are recommended. Items may be characterized by their complexity and their value or utility. The value of an item may be positive if the

item is useful for the user, or negative if the item is not appropriate and the user made a wrong decision when selecting it. We note that when a user is acquiring an item she will always incur in a cost, which includes the cognitive cost of searching for the item and the real monetary cost eventually paid for the item.

For instance, the designer of a news RS must take into account the complexity of a news item, i.e., its structure, the textual representation, and the time-dependent importance of any news item. But, at the same time, the RS designer must understand that even if the user is not paying for reading news, thereis always a cognitive cost associated to searching and reading news items. If a selected item is relevant for the user this cost is dominated by the benefit of having acquired a useful information, whereas if the item is not relevant the net value ofthat item for the user, and its recommendation, is negative. In other domains, e.g., cars, or financial investments, the true monetary cost of the items becomes an important element to consider when selecting the most appropriate recommendation approach.

Items with low complexity and value are: news, Web pages, books, CDs, movies. Items with larger complexity and value are: digital cameras, mobile phones, PCs, etc. The most complex items that have been considered are insurance policies, financial investments, travels, jobs.

RSs, according to their core technology, can use a range of properties and features of the items. For example in a movie recommender system, the genre (such as comedy, thriller, etc.), as well as the director, and actors can be used to describe a movie and to learn how the utility of an item depends on its features. Items can be represented using various information and representation approaches, e.g., in a minimalist way as a single id code, or in a richer form, as a set of attributes, but even as a concept in an ontological representation of the domain.

Users.

Users of a RS, as mentioned above, may have very diverse goals and characteristics. In order to personalize the recommendations and the human-computer interaction, RSs exploit a range of information about the users. This information can be structured in various ways and again the selection of what information to model depends on the recommendation technique.

Users can also be described by their behavior pattern data, for example, site browsing patterns (in a Web-based recommender system), or travel search patterns (in a travel recommender system). Moreover, user data may include relations between users such as the trust level of these relations between users. A RS might utilize this information to recommend items to users that were preferred by similar or trusted users.

Transactions.

We generically refer to a transaction as a recorded interaction between a user and the RS. Transactions are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using. For instance, a transaction log may contain a reference to the item selected by the user and a description of the context (e.g., the user goal/query) for that particular recommendation. If available, that transaction may also include an explicit feedback the user has provided, such as the rating for the selected item.

In fact, ratings are the most popular form of transaction data that a RS collects. These ratings may be collected explicitly or implicitly. In the explicit collection of ratings, the user is asked to provide her opinion about an item on a rating scale. Ratings can take on a variety of forms:

Numerical ratings such as the 1-5 stars provided in the book recommender associated with Amazon.com.

- **Ordinal ratings**, such as "strongly agree, agree, neutral, disagree, strongly disagree"

  where the user is asked to select the term that best indicates her opinion regarding an item (usually via questionnaire).
- **Binary ratings** that model choices in which the user is simply asked to decide if a certain item is good or bad.
- **Unary ratings** can indicate that a user has observed or purchased an item, or otherwise rated the item positively. In such cases, the absence of a rating indicates that we have no information relating the user to the item (perhaps she purchased the item somewhere else).

RECOMMENDATION TECHNIQUES

In order to implement its core function, identifying the useful items for the user, a RS must predict that an item is worth recommending. In order to do this, the system must be able to predict the utility of some of them, or at least compare the utility of some items, and then decide what items to recommend based on this comparison. The prediction step may not be explicit in the recommendation algorithm but we can still apply this unifying model to describe the general role of a RS. Here our goal is to provide the reader with a unifying perspective rather than an account of all the different recommendation approaches that will be illustrated in this handbook.

To provide a first overview of the different types of RSs, we want to quote a taxonomy provided by that has become a classical way of distinguishing between recommender systems and referring to them.

Distinguishes between six different classes of recommendation approaches:

Content-based: The system learns to recommend items that are similar to the ones that the user liked in the past. The similarity of items is calculated based on the features associated with the compared items. For example, if a user has positively

rated a movie that belongs to the comedy genre, then the system can learn to recommend other movies from this genre.

Collaborative filtering: The simplest and original implementation of this approach recommends to the active user the items that other users with similar tastes liked in the past. The similarity in taste of two users is calculated based on the similarity in the rating history of the users. This is the reason why refers to collaborative filtering as "people-to-people correlation." Collaborative filtering is considered to be the most popular and widely implemented technique in RS.

Demographic: This type of system recommends items 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. While these approaches have been quite popular in the marketing literature, there has been relatively little proper RS research into demographic systems.

Knowledge-based: Knowledge-based systems recommend items based on specific domain knowledge about how certain item features meet users needs and preferences and, ultimately, how the item is useful for the user. Notable knowledge based recommender systems are case-based. 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 utility of the recommendation for the user.

Community-based: This type of system recommends items based on the preferences of the users friends. This technique follows the epigram "Tell me who your friends are, and I will tell you who you are". 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, as or as they usually referred to, social recommender systems. This type of RSs models and acquires information about the social relations of the users and the preferences of the user's friends. The recommendation is based on ratings that were provided by the user's friends. In fact these RSs are following the rise of social-networks and enable a simple and comprehensive acquisition of data related to the social relations of the users.

Hybrid recommender systems: These RSs are based on the combination of the above mentioned 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 recommend 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 RSs techniques, several ways have been proposed for combining them to create a new hybrid.

The aspects that apply to the design stage include factors that might affect the choice of the algorithm. The first factor to consider, the application's domain, has a major effect on the algorithmic approach that should be taken. Based on the specific application domains, we define more general classes of domains for the most common recommender systems applications:

- **Entertainment** - recommendations for movies, music, and IPTV.
- **Content** - personalized newspapers, recommendation for documents, recommendations of Web pages, e-learning applications, and e-mail filters.
- **E-commerce** - recommendations for consumers of products to buy such as books, cameras, PCs etc.

- **Services** - recommendations of travel services, recommendation of experts for consultation, recommendation of houses to rent, or matchmaking services.

CONTENT-BASED RECOMMENDER SYSTEMS

Recommender systems have the effect of guiding users in a personalized way to interesting objects in a large space of possible options. Content-based recommendation systems try to recommend items similar to those a given user has liked in the past. Indeed, the basic process performed by a content-based recommender consists in matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of a content object (item), in order to recommend to the user new interesting items. This chapter provides an overview of content-based recommender systems, with the aim of imposing a degree of order on the diversity of the different aspects involved in their design and implementation.

The first part of the chapter presents the basic concepts and terminology of content based recommender systems, a high level architecture, and their main advantages and drawbacks. The second part of the chapter provides a review of the state of the art of systems adopted in several application domains, by thoroughly describing both classical and advanced techniques for representing items and user profiles. The most widely adopted techniques for learning user profiles are also presented. The last part of the chapter discusses trends and future research which might lead towards the next generation of systems, by describing the role of User Generated Content as a way for taking into account evolving vocabularies, and the challenge of feeding users with serendipitous recommendations, that is to say surprisingly interesting items that they might not have otherwise discovered.

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.

BASICS OF CONTENT-BASED RECOMMENDER SYSTEMS

Systems implementing a content-based recommendation approach analyze a set of documents and/or descriptions of items previously rated by a user, and build a model or profile of user interests based on the features of the objects rated by that user. The profile is a structured representation of user interests, adopted to recommend new interesting items. The recommendation process basically consists in matching up the attributes of the user profile against the attributes of a content object. The result is a relevance judgment that represents the user's level of interest in that object. If a profile accurately reflects user preferences, it is of tremendous advantage for the effectiveness of an information access process. For instance, it could be used to filter search results by deciding whether a user is interested in a specific Web page or not and, in the negative case, preventing it from being displayed.

A HIGH LEVEL ARCHITECTURE OF CONTENT-BASED SYSTEMS

Content-based Information Filtering (IF) systems need proper techniques for representing the items and producing the user profile, and some strategies for comparing the user profile with the item representation. The high level architecture of a content based recommender system is depicted in Figure 3.1. The recommendation process is performed in three steps, each of which is handled by a separate component:

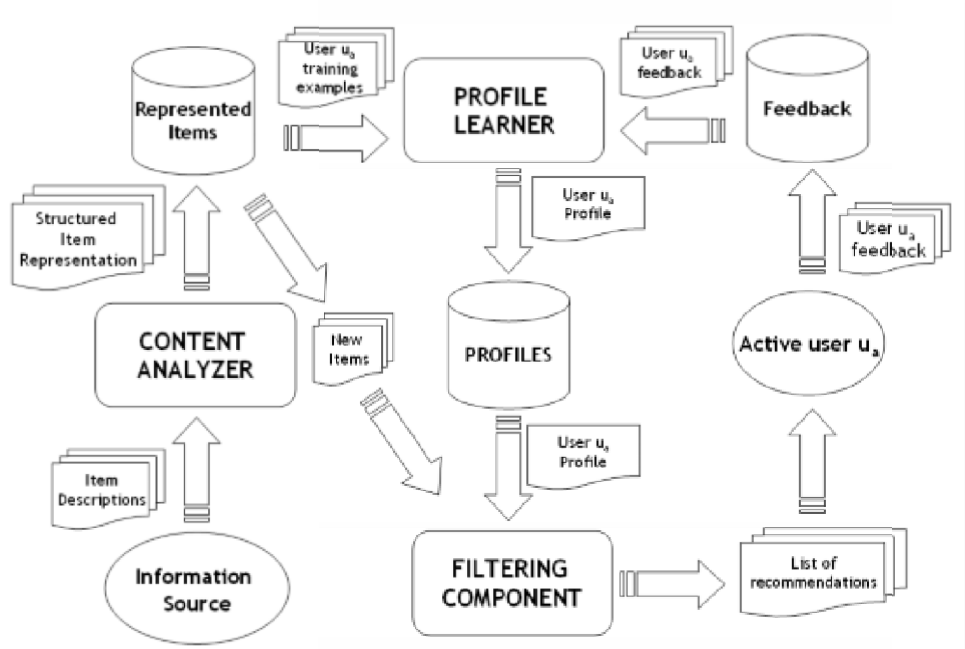

Fig. 3.1: High level architecture of a Content-based Recommender

• **CONTENT ANALYZER** – When information has no structure (e.g. text), some kind of pre-processing step is needed to extract structured relevant information. The main responsibility of the component is to represent the content of items (e.g. documents, Web pages, news, product descriptions, etc.) coming from information sources in a form suitable for the next processing steps. Data items are analyzed by feature extraction techniques in order to shift item representation from the original information space to the target one (e.g. Web pages represented as keyword vectors). This representation is the input to the PROFILE LEARNER and FILTERING COMPONENT;

• **PROFILE LEARNER** – This module collects data representative of the user preferences and tries to generalize this data, in order to construct the user profile. Usually, the generalization strategy is realized through machine learning techniques, which are able to infer a model of user interests starting from items liked or disliked in the past. For instance, the PROFILE LEARNER of a Web page

recommender can implement a relevance feedback method in which the learning technique combines vectors of positive and negative examples into a prototype vector representing the user profile. Training examples are Web pages on which a positive or negative feedback has been provided by the user;

• **FILTERING COMPONENT** – This module exploits the user profile to suggest relevant items by matching the profile representation against that of items to be recommended. The result is a binary or continuous relevance judgment (computed using some similarity metrics [42]), the latter case resulting in a ranked list of potentially interesting items. In the above mentioned example, the matching is realized by computing the cosine similarity between the prototype vector and the item vectors.

The first step of the recommendation process is the one performed by the CONTENT ANALYZER, that usually borrows techniques from Information Retrieval system. Item descriptions coming from Information Source are processed by the CONTENT ANALYZER, that extracts features (keywords, n-grams, concepts, . . .) from unstructured text to produce a structured item representation, stored in the repository Represented Items. In order to construct and update the profile of the active user u (user for which recommendations must be provided) her reactions to items are collected in some way and recorded in the repository Feedback. These reactions, called annotations or feedback, together with the related item descriptions, are exploited during the process of learning a model useful to predict the actual relevance of newly presented items. Users can also explicitly define their areas of interest as an initial profile without providing any feedback. Typically, it is possible to distinguish between two kinds of relevance feedback:

- positive information (inferring features liked by the user) and
- negative information

Two different techniques can be adopted for recording user's feedback. When a system requires the user to explicitly evaluate items, this technique is usually referred to as "explicit feedback"; the other technique, called "implicit

feedback", a does not require any active user involvement, in the sense that feedback is derived from monitoring and analyzing user's activities. Explicit evaluations indicate how relevant or interesting an item is to the user. There are three main approaches to get explicit relevance feedback:

like/dislike – items are classified as "relevant" or "not relevant" by adopting a simple binary rating scale; ratings – a discrete numeric scale is usually adopted to judge items. Alternatively, symbolic ratings are mapped to a numeric scale, such as in Syskill & Webert, where users have the possibility of rating a Web page as hot, lukewarm, or cold; text comments – Comments about a single item are collected and presented to the users as a means of facilitating the decision-making process. For instance, customer's feedback at Amazon.com or eBay.com might help users in deciding whether an item has been appreciated by the community. Textual comments are helpful, but they can overload the active user because she must read and interpret each comment to decide if it is positive or negative, and to what degree.

The literature proposes advanced techniques from the affective computing research area to make content-based recommenders able to automatically perform this kind of analysis.

ADVANTAGES AND DRAWBACKS OF CONTENT-BASED FILTERING

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 since they rated the same items similarly. Then, only the items that are most liked by the neighbors of the active user will be recommended;

• **TRANSPARENCY** - Explanations on how the recommender system works can be provided by explicitly listing content features or descriptions that caused an item to occur in the list of recommendations. Those features are indicators to consult in order to decide whether to trust a recommendation. Conversely, collaborative systems are black boxes since the only explanation for an item recommendation is that unknown 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 first-rater problem, which affects collaborative recommenders which rely 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.

Nonetheless, content-based systems have several shortcomings:

• **LIMITED CONTENT ANALYSIS** - Content-based techniques have a natural limit in the number and type of features that are associated, whether automatically or manually, with the objects they recommend. Domain knowledge is often needed, e.g., for movie recommendations the system needs to know the actors and directors, and sometimes, domain ontologies are also needed. No content-based recommendation system can provide suitable suggestions if the analyzed content does not contain enough information to discriminate 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, often there 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 techniques from text completely ignore aesthetic qualities and additional multimedia information.

To sum up, both automatic and manually assignment of features to items could not be sufficient to define distinguishing aspects of items that turn out to be necessary for the elicitation of user interests.

OVER-SPECIALIZATION - Content-based recommenders have no inherent method for finding something unexpected. The system suggests items whose scores are high when matched against the user profile, hence the user is going to be recommended items similar to those already rated. This drawback is also called serendipity problem to highlight the tendency of the content-based systems to produce recommendations with a limited degree of novelty. To give an example, when a user has only rated movies directed by Stanley Kubrick, she will be recommended just that kind of movies. A "perfect" content-based technique would rarely 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, as for a new user, the system will not be able to provide reliable recommendations.

COLLABORATIVE FILTERING

The collaborative filtering (CF) approach to recommenders has recently enjoyed much interest and progress. The fact that it played a central role within the recently completed Netflix competition has contributed to its popularity. This chapter surveys the recent progress in the field. Matrix factorization techniques, which became a first choice for implementing CF, are described together with recent innovations. We also describe several extensions that bring competitive accuracy into neighborhood methods, which used to dominate the field. The chapter demonstrates how to utilize temporal models and implicit feedback to extend models accuracy. In passing, we include detailed descriptions of some the central methods developed for tackling the challenge of the Netflix Prize competition.

Collaborative filtering (CF) methods produce user specific recommendations of items based on patterns of ratings or usage (e.g., purchases) without need for exogenous information about either items or users. While well established methods work adequately for many purposes, we present several recent extensions available

to analysts who are looking for the best possible recommendations.

The Netflix Prize competition that began in October 2006 has fueled much recent progress in the field of collaborative filtering. For the first time, the research community gained access to a large-scale, industrial strength data set of 100 million movie ratings attracting thousands of scientists, students, engineers and enthusiasts to the field. The nature of the competition has encouraged rapid development, where innovators built on each generation of techniques to improve prediction accuracy.

Because all methods are judged by the same rigid yardstick on common data, the evolution of more powerful models has been especially efficient. Recommender systems rely on various types of input. Most convenient is high quality explicit feedback , where users directly report on their interest in products. For example, Netflix collects star ratings for movies and TiVo users indicate their preferences for TV shows by hitting thumbs-up/down buttons.

MATRIX FACTORIZATION MODELS

Latent factor models approach collaborative filtering with the holistic goal to uncover latent features that explain observed ratings; examples include pLSA, neural networks, Latent Dirichlet Allocation, and models that are induced by factorization of the user-item ratings matrix (also known as SVD-based models). Recently, matrix factorization models have gained popularity, thanks to their attractive accuracy and scalability. In information retrieval, SVD is well established for identifying latent semantic factors. However, applying SVD to explicit ratings in the CF domain raises difficulties due to the high portion of missing values.

Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting. Earlier works relied on imputation, which fills in missing ratings and makes the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, the data may be considerably distorted due to inaccurate imputation. Hence, more recent

works suggested modeling directly only the observed ratings, while avoiding overfitting through an adequate regularized model.

In this section we describe several matrix factorization techniques, with increasing complexity and accuracy. We start with the basic model – "SVD". Then, we show how to integrate other sources of user feedback in order to increase prediction accuracy, through the "SVD++ model". Finally we deal with the fact that customer preferences for products may drift over time. Product perception and popularity are constantly changing as new selection emerges. Similarly, customer inclinations are evolving, leading them to ever redefine their taste. This leads to a factor model that addresses temporal dynamics for better tracking user behavior.

SVD

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f , such that user-item interactions are modeled as inner products in that space. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback. For example, when the products are movies, factors might measure obvious dimensions such as comedy vs. drama, amount of action, or orientation to children; less well defined dimensions such as depth of character development or "quirkiness"; or completely uninterpretable dimensions.

SVD++

Prediction accuracy is improved by considering also implicit feedback, which provides an additional indication of user preferences. This is especially helpful for those users that provided much more implicit feedback than explicit one. As explained earlier, even in cases where independent implicit feedback is absent, one can capture a significant signal by accounting for which items users rate, regardless of their rating value. This led to several methods that modeled a user factor by the identity of the items he/she has rated. Here we focus on the SVD++ method, which was shown to offer accuracy superior to SVD.

Time-aware factor model

The matrix-factorization approach lends itself well to modeling temporal effects, which can significantly improve its accuracy. Decomposing ratings into distinct terms allows us to treat different temporal aspects separately. Specifically, we identify the following effects that each vary over time:

(1) user biases b(t),

(2) item biases bi(t), and

(3) user preferences p(t).

On the other hand, we specify static item characteristics, q_i, because we do not expect significant temporal variation for items, which, unlike humans, are static in nature. We start with a detailed discussion of the temporal effects that are contained within the baseline predictors.

NEIGHBORHOOD MODELS

The most common approach to CF is based on neighborhood models. Its original form, which was shared by virtually all earlier CF systems, is user-user based; see for a good analysis. User-user methods estimate unknown ratings based on recorded ratings of likeminded users. Later, an analogous item-item approach became popular. In those methods, a rating is estimated using known ratings made by the same user on similar items. Better scalability and improved accuracy make the item-item approach more favorable in many cases. In addition, item-item methods are more amenable to explaining the reasoning behind predictions. This is because users are familiar with items previously preferred by them, but do not know those allegedly like-minded users. We focus mostly on item-item approaches, but the same techniques can be directly applied within a user-user approach;

A global neighborhood model

In this subsection, we introduce a neighborhood model based on global optimization. The model offers an improved prediction accuracy, by offering the

aforementioned merits of the model. with additional advantages that are summarized as follows:

1. No reliance on arbitrary or heuristic item-item similarities. The new model is cast as the solution to a global optimization problem.
2. Inherent overfitting prevention or "risk control": the model reverts to robust baseline predictors, unless a user entered sufficiently many relevant ratings.
3. The model can capture the totality of weak signals encompassed in all of a user's ratings, not needing to concentrate only on the few ratings for most similar items.
4. The model naturally allows integrating different forms of user input, such as explicit and implicit feedback.
5. A highly scalable implementation allows linear time and space complexity, thus facilitating both item-item and user-user implementations to scale well to very large datasets.
6. Time drifting aspects of the data can be integrated into the model, thereby improving its accuracy.

Neighborhood-based Recommendation Methods

Abstract Among collaborative recommendation approaches, methods based on nearest-neighbors still enjoy a huge amount of popularity, due to their simplicity, their efficiency, and their ability to produce accurate and personalized recommendations. This chapter presents a comprehensive survey of neighborhood-based methods for the item recommendation problem. In particular, the main benefits of such methods, as well as their principal characteristics, are described. Furthermore, this document addresses the essential decisions that are required while implementing a neighborhood-based recommender system, and gives practical information on how to make such decisions. Finally, the problems of sparsity and limited coverage, often observed in large commercial recommender systems, are discussed, and a few solutions to overcome these problems are presented.

Building the Model

We gradually construct the various components of the model, through an ongoing refinement of our formulations. Previous models were centered around *user-specific* interpolation weights – θ_{ij}^u in (5.19) or $s_{ij}/\sum_{j\in S^k(i;u)} s_{ij}$ in (5.17) – relating item i to the items in a user-specific neighborhood $S^k(i;u)$. In order to facilitate global optimization, we would like to abandon such user-specific weights in favor of global item-item weights independent of a specific user. The weight from j to i is denoted by w_{ij} and will be learned from the data through optimization. An initial sketch of the model describes each rating r_{ui} by the equation

$$\hat{r}_{ui} = b_{ui} + \sum_{j\in R(u)} (r_{uj} - b_{uj}) w_{ij} . \tag{5.29}$$

This rule starts with the crude, yet robust, baseline predictors (b_{ui}). Then, the estimate is adjusted by summing over *all* ratings by u.

Let us consider the interpretation of the weights. Usually the weights in a neighborhood model represent interpolation coefficients relating unknown ratings to existing ones. Here, we adopt a different viewpoint, that enables a more flexible usage of the weights. We no longer treat weights as interpolation coefficients. Instead, we take weights as part of adjustments, or *offsets*, added to the baseline predictors. This way, the weight w_{ij} is the extent by which we increase our baseline prediction of r_{ui} based on the observed value of r_{uj}. For two related items i and j, we expect w_{ij} to be high. Thus, whenever a user u rated j higher than expected ($r_{uj} - b_{uj}$ is high), we would like to increase our estimate for u's rating of i by adding $(r_{uj} - b_{uj})w_{ij}$ to the baseline prediction. Likewise, our estimate will not deviate much from the baseline by an item j that u rated just as expected ($r_{uj} - b_{uj}$ is around zero), or by an item j that is not known to be predictive on i (w_{ij} is close to zero).

5.5.2.1 Factoring item-item relationships

We factor item-item relationships by associating each item i with three vectors: $q_i, x_i, y_i \in \mathbb{R}^f$. This way, we confine w_{ij} to be $q_i^T x_i$. Similarly, we impose the structure $c_{ij} = q_i^T y_j$. Essentially, these vectors strive to map items into an f-dimensional latent factor space where they are measured against various aspects that are revealed automatically by learning from the data. By substituting this into (5.34) we get the following prediction rule:

$$\hat{r}_{ui} = \mu + b_u + b_i + |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} [(r_{uj} - b_{uj}) q_i^T x_j + q_i^T y_j] \tag{5.37}$$

Computational gains become more obvious by using the equivalent rule

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(|R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} (r_{uj} - b_{uj}) x_j + y_j \right) . \tag{5.38}$$

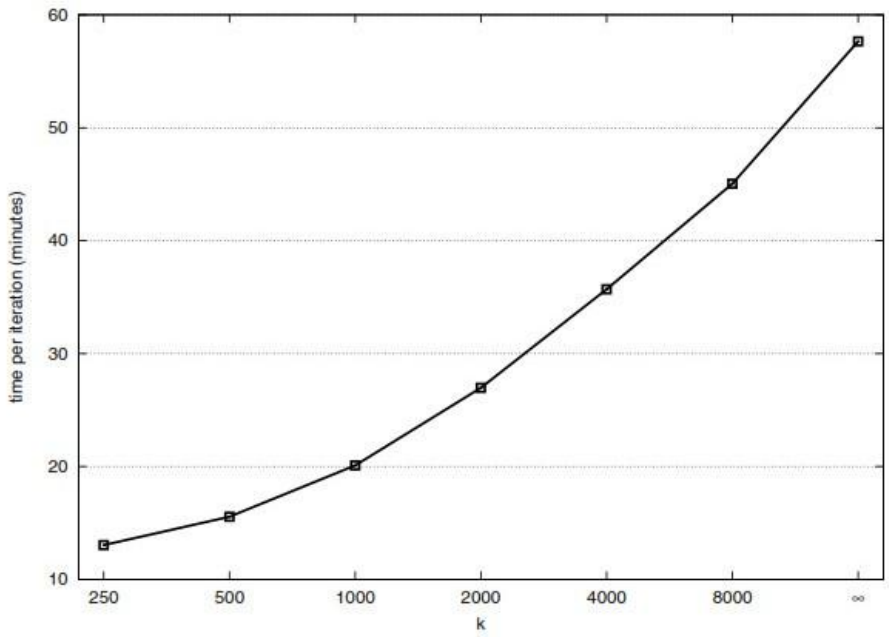

Fig. 5.2: Running time per iteration of the globally optimized neighborhood model, as a function of the parameter k.

Notice that the bulk of the rule $(|\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) x_j + y_j)$ depends only on u while being independent of i. This leads to an efficient way to learn the model parameters. As usual, we minimize the regularized squared error function associated with (5.38)

$$\min_{q_*, x_*, y_*, b_*} \sum_{(u,i) \in \mathcal{K}} \left(r_{ui} - \mu - b_u - b_i - q_i^T \left(|\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) x_j + y_j \right) \right)^2 + \lambda_{11} \left(b_u^2 + b_i^2 + \|q_i\|^2 + \sum_{j \in \mathrm{R}(u)} \|x_j\|^2 + \|y_j\|^2 \right). \quad (5.39)$$

Optimization is done by a stochastic gradient descent scheme, which is described in the following pseudo code:

PART-A

1 Examine the broad classification of Recommendation systems?
2 Justify Content Based recommendation system
3 Classify collaborative filtering system
4 Define knowledge based Recommendation
5 Give the definition Hybrid recommendation
6 Define Meta level and Cascade Recommendation system
7 Examine Knowledge based configuration in detail
8 Explain the types of Recommendation System
9 Point out some advantages of Mobile Recommendation system
10 Demonstrate Used based and Model based Collaborative filtering
11 Describe Recommendation Techniques in detail
12 Examine the advantages of Content Based recommendation system
13 What are the types of Hybrid recommendation system
14 Define web recommendation in detail
15 Describe Hybrid recommendation system.
16 List the advantages of Collaborative Recommendation System
17 Construct Collaborative and Content based recommendation system
18 Estimate some example for content based recommendation system
19 Examine disadvantages of content based recommendation system
20 Describe Weighted Recommenders

PART-B

1 Define Recommendation based on User Ratings using appropriate example. (13)
2 i) Explain Recommender system. (4)
ii) Explain the techniques of Matrix Factorization (9)
3 Explain the different types of recommendation system.
i) Hybrid Recommendation System (3)
ii) Content Based Recommendation System (3)
iii) Collaborative Recommendation System (3)
iv) Knowledge Based Recommendation System (4)
4 Estimate the Content based recommendation system (13)
5 Differentiate collaborative filtering and content based systems. (13)
6 i) Explain about High Level Architecture (6)
ii) Explain the significance of Collaborative Filtering in detail. (7)
7 Illustrate the advantages and disadvantages of Content based and collaborative filtering recommendation system (13)
8 Describe Knowledge based recommendation system in detail (13)
9 i) Detailed the rules of HLA (7)
ii) Difference between Hybrid and Collaborative Recommendation (6)
10 i) Describe common HLA terminologies. (3)
ii) Define the steps involved in Collaborative Filtering (10)
11 i) Describe web based recommendation system (7)
ii) When can Collaborative Filtering be used? (6)
12 Define in detail about Matrix factorization models (13)
13 Discuss Neighbouring model in detail (13)

14 i) Explain is Matrix Factorization? (4)
ii) Discuss the approaches of recommender system. (9)

PART-C

1 Narrate in detail about a model for Recommendation system. (15)
2 Discuss in detail about High Level Architecture and also common Terminologies in HLA. (15)
3 Classify Recommendation techniques with examples (15)
4 i) Design Matrix factorization model (8)
ii) Detail about Neighbouring models in detail (7)