DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Development of a Recommender System that addresses the diversity principle to improve user satisfaction.

Ozan Pekmezci

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Development of a Recommender System that addresses the diversity principle to improve user satisfaction.

Entwicklung eines Empfehlungdienst unter Berücksichtigung der Benutzerzufriedenheit zur Diversität der Empfehlungen.

Author:	Ozan Pekmezci
Supervisor:	Prof. Dr-Ing. Klaus Diepold
Advisor:	Julian Wörmann, M.Sc.
Submission Date:	15.05.2019

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.05.2019 Ozan Pekmezci

Acknowledgments

Abstract

Contents

1 Introduction

Recommender Systems (RSs) are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user [RRS15].

Suggestions and items depend on the field that recommder system is applied. For example, for the topic of news article recommenders, the aim will most likely be suggesting news to the readers. In the field of job recommenders though, these suggestions can be bidirectional. Meaning that, job postings can be suggested to applicants or resumes can be recommended to the human resources team of a company.

This chapter of the thesis will focus on explaining the basic terminology of recommender systems so that readers who are new to the topic can understand the rest easily.

1.1 Background of Classical Recommender Systems

Although the history of the recommender systems go back to mid-1990s [Par+12], the real boom happened after e-commerce services became mainstream [1]. Since there were too many items to choose from for users, such service was needed. Users of websites were becoming overloaded with the information and the developers of recommender systems had aim of reducing the information to be only relevant to users.

This section contains brief information about classical recommender systems.

1.1.1 Why Recommender Systems?

As mentioned in the last paragraph, recommender systems have the general function of suggesting items to users. However, why do recommender systems get developed? What kind of benefit do they have for both companies and users?

First of all, recommender systems increase the number of items sold [RRS15]. Also, most recommenders suggest personalized results, which means that users will see content that fits their desires. This will also increase users buying more items.

Recommenders also increase the coverage of items that user see. Coverage denotes number of recommended unique items divided by the number of all items. Therefore, users can iteract with items that they wouldn't even see without recommenders, that improves chances of buying more items.

Another important point is definitely increasing the user satisfaction. This function of recommenders is the foundation of the thesis at hand. Unfortunately, most of the researchers don't take into account that the user satisfaction does not solely depend on simple evaluation metrics like accuracy, precision or recall but it can also depend on privacy, data security, diversity, serendipity, labeling, and presentation [Bee+16]. When recommender systems take those factors into account, they clearly increase user satisfaction. [Gerekirse iki madde daha var]

1.1.2 Functions

To achieve the goals in the previous subsection, recommender systems need to fulfill some functions. Common functions include but not limited to [RRS15]:
[kopi - peyst]

- Find Some Good Items: Recommend to a user some items as a ranked list along with predictions of how much the user would like them (e.g., on a scale of one-to-five stars). This is the main recommendation task that many commercial systems address (see, for instance, Chap. 11). Some systems do not show the predicted rating.
- Find all good items: Recommend all the items that can satisfy some user needs. In such cases it is insufficient to just find some good items. This is especially true when the number of items is relatively small or when the RS is mission-critical, such as in medical or financial applications. In these situations, in addition to the benefit derived from carefully examining all the possibilities, the user may also benefit from the RS ranking of these items or from additional explanations that the RS generates.
- Recommend a sequence: Instead of focusing on the generation of a single recommendation, the idea is to recommend a sequence of items that is pleasing as a whole. Typical examples include recommending a TV series, a book on RSs after having recommended a book on data mining, or a compilation of musical tracks
- Recommend a bundle: Suggest a group of items that fits well together. For instance, a travel plan may be composed of various attractions, destinations, and accommodation services that are located in a delimited area. From the point of view of the user, these various alternatives can be considered and selected as a single travel destination

[kopi - peyst]

1.1.3 Applications

[kopi - peyst]

- Entertainment—recommendations for movies, music, games, and IPTV.
- Content—personalized newspapers, recommendation for documents, recommendations of webpages, e-learning applications, and e-mail filters.
- E-commerce—recommendations of products to buy such as books, cameras, PCs etc. for consumers.
- Services—recommendations of travel services, recommendation of experts for consultation, recommendation of houses to rent, or matchmaking services.
- Social—recommendation of people in social networks, and recommendations of content social media content such as tweets, Facebook feeds, LinkedIn updates, and others.

1.1.4 Objects

Data used by typical recommender systems refer to three types of objects [RRS15]: users, items and interactions.

[kopi - peyst] 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 to the user, or negative if the item is not appropriate and the user made the wrong decision when selecting it. We note that when a user is acquiring an item, one 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.

Users Users of an 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.

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 recommen- dation. If available, that transaction may also include explicit feedback that the user has provided, such as the rating for the selected item. Implicit, explicit unutma ve uzat, ornekle

[kopi - peyst]

1.1.5 Types

[gelistir, buyut]

- Collaborative-Filtering: Recommendations based on how other users rated items. Similar users are identified and the items that they rated well are recommended. Has cold-start, sparsity and scalibility issues. First research paper released in the mid-1990s, still used widely [8].
- Content-based filtering: The requirements for this approach are features and the ratings of the items by users. A classifier for users' 'profile' is built and similar items are recommended based on it. It has the problem of recommending only very similar results that the user already is aware of. It was first mentioned on an academic paper on 1998 [8].
- Knowledge-based filtering: This approach also requires features of the items and explicit description of what the user needs or wants. Then, items that match those needs are recommended. The advantage of this approach is the fact that the system doesn't need data from different users. Unfortunately, the suggestion ability is rather static [4]. Research about this topic was first released on 1999.
- Hybrid Recommender Systems: Since all of the methods have some drawbacks, applications have shifted to combining more than one of the previous approaches. Hybrid recommender systems are still widely used by big companies like Amazon [1],Spotify [2] and Netflix [3].

1.1.6 Evaluation Metrics

Evaluation metrics are bla

Offline Evaluation

- Accuracy
- Precision
- Recall

Online Evaluation

Off-line experiments can measure the quality of the chosen algorithm in fulfilling its recommendation task. However, such evaluation cannot provide any insight about the user satisfaction, acceptance or experience with the system. The algorithms might be very accurate in solving the core recommendation problem, i.e., predicting user ratings, but for some other reason the system may not be accepted by users, for example, because the performance of the system was not as expected. Therefore, a user-centric evaluation is also required. It can be performed online after the system has been launched, or as a focused user study. During on-line evaluation, real users interact with the system without being aware of the full nature of the experiment running in the background. It is possible to run various versions of the algorithms on different groups of users for comparison and analysis of the system logs in order to enhance system performance. In addition, most of the algorithms include parameters, such as weight thresholds, the number of neighbors, etc., requiring constant adjustment and calibration.

[kopi - peyst]

More information in research part

1.1.7 User Satisfaction

[proposaldan]

- The recommendation performance is mainly evaluated in terms of accuracy (i.e. difference between true and predicted rating). - This might not be a desired goal of the talent/project recommendations (e.g. simply recommending a 'top-N' list of talents that best match the requirements might result in similar talents whose skills only vary in a small extent). - Besides the desired diversity, there might be other properties necessary for adequate recommendations (e.g. privacy, data security, diversity, serendipity, labeling, and presentation, and group recommendation).

1.2 Motivation

- Thus, the aim of this thesis is to investigate how these properties transfer to the talent/project matching (i.e. are they necessary or not) and how they can be algorithmically achieved.

[Also explain job recommender if we do that]

1.3 N/a

In recent years, the interest in recommender systems has dramatically increased, as the following facts indicate: 1. Recommender systems play an important role in highly-rated Internet sites such as Amazon.com, YouTube, Netflix, Spotify, LinkedIn, Facebook, 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. For example, Netflix, the online provider of on-demand streaming media, awarded a million dollar prize to the team that first succeeded in substantially improving the performance of its recommender system [31]. 2. There are conferences and workshops dedicated specifically to the field, namely the Association of Computing Machinery's (ACM) Conference Series on Recommender Systems (RecSys), established in 2007. This conference stands as the premier annual event in recommender technology research and applications. In addition, sessions dedicated to RSs are frequently included in more traditional conferences in the area of databases, information systems and adaptive systems. Additional noteworthy conferences within this scope include: ACM's Special Interest Group on Information Retrieval (SIGIR); User Modeling, Adaptation and Personalization (UMAP); Intelligent User Interfaces (IUI); World Wide Web (WWW); and ACM's Special Interest Group on Management Of Data (SIGMOD). 3. At institutions of higher education around the world, undergraduate and graduate courses are now dedicated entirely to RSs, tutorials on RSs are very popular at computer science conferences, and a book introducing RSs techniques has been published as well [27]. Springer is publishing several books on specific topics in recommender systems in its series: Springer Briefs in Electrical and Computer Engineering. A large, new collection of articles dedicated to recommender systems applications to software engineering has also recently been published [46]. 4. There have been several special issues in academic journals which cover research and developments in the RSs field. Among the journals that have dedicated issues to RSs are: AI Communications (2008); IEEE Intelligent Systems (2007); International Journal of Electronic Commerce (2006); International Journal of Computer Science and Applications (2006); ACM Transactions on Computer Human Interaction (2005); ACM Transactions on Information Systems (2004); User Modeling and User-Adapted Interaction (2014, 2012); ACM Transactions on Interactive Intelligent Systems (2013); and ACM Transactions on Intelligent Systems and Technology (2015). [RRS15]

2 Review of Literature and Research

2.1 Types of Recommender Systems

[kopi peyst] Item recommendation approaches can be divided in two broad categories: per- sonalized and non-personalized. Among the personalized approaches are content-based and collaborative filtering methods, as well as hybrid techniques combining these two types of methods. The general principle of content-based (or cognitive) methods [4, 8, 42, 54] is to identify the common characteristics of items that have received a favorable rating from a user, and then recommend to this user new items that share these characteristics. Recommender systems based purely on content generally suffer from the problems of limited content analysis and over- specialization [63]. Limited content analysis occurs when the system has a limited amount of information on its users or the content of its items. For instance, privacy issues might refrain a user from providing personal information, or the precise content of items may be difficult or costly to obtain for some types of items, such as music or images. Another problem is that the content of an item is often insufficient to determine its quality. Over-specialization, on the other hand, is a side effect of the way in which content-based systems recommend new items, where the predicted rating of a user for an item is high if this item is similar to the ones liked by this user. For example, in a movie recommendation application, the system may recommend to a user a movie of the same genre or having the same actors as movies already seen by this user. Because of this, the system may fail to recommend items that are different but still interesting to the user.

Instead of depending on content information, collaborative (or social) filtering approaches use the rating information of other users and items in the system. The key idea is that the rating of a target user for a new item is likely to be similar to that of another user, if both users have rated other items in a similar way. Likewise, the target user is likely to rate two items in a similar fashion, if other users have given similar ratings to these two items. Collaborative approaches overcome some of the limitations of content-based ones. For instance, items for which the content is not available or difficult to obtain can still be recommended to users through the feedback of other users. Furthermore, collaborative recommendations are based on the quality of items as evaluated by peers, instead of relying on content that may be a bad indicator of quality. Finally, unlike content-based systems, collaborative filtering ones can recommend items with very different content, as long as other users have already shown interest for these different items.

Collaborative filtering approaches can be grouped in the two general classes of neighborhood and model-based methods. In neighborhood-based (memory-based [10] or heuristic-based [2]) collaborative filtering [14, 15, 27, 39, 44, 48, 57, 59, 63], the user-item ratings stored in the system are directly used to predict ratings for new items. This can be done in two ways known as user-based or item-based recommendation. User-based systems, such as GroupLens [39], Bellcore video [27], and Ringo [63], evaluate the interest of a target user for an item using the ratings for this item by other users, called neighbors, that have similar rating patterns. The neighbors of the target user are typically the users whose ratings are most correlated to the target user's ratings. Item-based approaches [15, 44, 59], on the other hand, predict the rating of a user for an item based on the ratings of the user for similar items. In such approaches, two items are similar if several users of the system have rated these items in a similar fashion.

In contrast to neighborhood-based systems, which use the stored ratings directly in the prediction, model-based approaches use these ratings to learn a predictive model. Salient characteristics of users and items are captured by a set of model parameters, which are learned from training data and later used to predict new ratings. Model-based approaches for the task of recommending items are numerous and include Bayesian Clustering [10], Latent Semantic Analysis [28], Latent Dirich- let Allocation [9], Maximum Entropy [72], Boltzmann Machines [58], Support Vector Machines [23], and Singular Value Decomposition [6, 40, 53, 68, 69]. A survey of state-of-the-art model-based methods can be found in Chap. 3 of this book.

Finally, to overcome certain limitations of content-based and collaborative filtering methods, hybrid recommendation approaches combine characteristics of both types of methods. Content-based and collaborative filtering methods can be combined in various ways, for instance, by merging their individual predictions into a single, more robust prediction [8, 55], or by adding content information into a collaborative filtering model [1, 3, 51, 65, 71]. Several studies have shown hybrid recommendation approaches to provide more accurate recommendations than pure content-based or collaborative methods, especially when few ratings are available [2].

2.2 Neighborhood-Based Approach

2.2.1 Introduction

[kopi peyst] While recent investigations show state-of-the-art model-based approaches superior to neighborhood ones in the task of predicting ratings [40, 67], there is also an emerging understanding that good prediction accuracy alone does not guarantee users an effective and satisfying experience.

Model-based approaches excel at characterizing the preferences of a user with latent factors. For example, in a movie recommender system, such methods may determine that a given user is a fan of movies that are both funny and romantic, without having to actually define the notions “funny” and “romantic”. This system would be able to recommend to the user a romantic comedy that may not have been known to this user. However, it may be difficult for this system to recommend a movie that does not quite fit this high-level genre, for instance, a funny parody of horror movies. Neighborhood approaches, on the other hand, capture local associations in the data. Consequently, it is possible for a movie recommender system based on this type of approach to recommend the user a movie very different from his usual taste or a movie that is not well known (e.g. repertoire film), if one of his closest neighbors has given it a strong rating. This recommendation may not be a guaranteed success, as would be a romantic comedy, but it may help the user discover a whole new genre or a new favorite actor/director. -> Model based suck at serendipity

- Simplicity: Neighborhood-based methods are intuitive and relatively simple to implement. In their simplest form, only one parameter (the number of neighbors used in the prediction) requires tuning.

- Justifiability: Such methods also provide a concise and intuitive justification for the computed predictions. For example, in item-based recommendation, the list of neighbor items, as well as the ratings given by the user to these items, can be presented to the user as a justification for the recommendation. This can help the user better understand the recommendation and its relevance, and could serve as basis for an interactive system where users can select the neighbors for which a greater importance should be given in the recommendation [6].

- Efficiency: One of the strong points of neighborhood-based systems are their efficiency. Unlike most model-based systems, they require no costly training phases, which need to be carried at frequent intervals in large commercial applications. These systems may require pre-computing nearest neighbors in an offline step, which is typically much cheaper than model training, providing near instantaneous recommendations. Moreover, storing these nearest neighbors requires very little memory, making such approaches scalable to applications having millions of users and items.

- Stability: Another useful property of recommender systems based on this approach is that they are little affected by the constant addition of users, items and ratings, which are typically observed in large commercial applications. For instance, once item similarities have been computed, an item-based system can readily make recommendations to new users, without having to re-train the system. Moreover, once a few ratings have been entered for a new item, only the similarities between this item and the ones already in the system need to be computed.

While neighborhood-based methods have gained popularity due to these advantages, they are also known to suffer from the problem of limited coverage, which causes some items to be never recommended. Also, traditional methods of this category are known to be more sensitive to the sparseness of ratings and the cold-start problem, where the system has only a few ratings, or no rating at all, for new users and items. Section 2.5 presents more advanced neighborhood-based techniques that can overcome these problems.

2.2.2 User-Based Rating Prediction

[Kitapta 2.3.1] 2 Sayfa kadar

2.2.3 User-Based Classification

[Kitapta 2.3.2] 0.5 Sayfa kadar

2.2.4 Regression vs Classification

[kopi -peys, genel daha uzun bahsedilebilir ve belki resim eklenebilir] The choice between implementing a neighborhood-based regression or classifica- tion method largely depends on the system's rating scale. Thus, if the rating scale is continuous, e.g. ratings in the Jester joke recommender system [20] can take any value between -10 and 10, then a regression method is more appropriate. On the contrary, if the rating scale has only a few discrete values, e.g. "good" or "bad", or if the values cannot be ordered in an obvious fashion, then a classification method might be preferable. Furthermore, since normalization tends to map ratings to a continuous scale, it may be harder to handle in a classification approach. Another way to compare these two approaches is by considering the situation where all neighbors have the same similarity weight. As the number of neighbors used in the prediction increases, the rating rui predicted by the regression approach will tend toward the mean rating of item i. Suppose item i has only ratings at either end of the rating range, i.e. it is either loved or hated, then the regression approach will make the safe decision that the item's worth is average. This is also justified from a statistical point of view since the expected rating (estimated in this case) is the one that minimizes the RMSE. On the other hand, the classification approach will predict the rating as the most frequent one given to i. This is more risky as the item will be labeled as either "good" or "bad". However, as mentioned before, taking risks may be desirable if it leads to serendipitous recommendations.

2.2.5 Item-Based Recommendation

[Kitapta 2.3.2] 1 Sayfa kadar

2.2.6 Item-Based vs User-Based Recommendation

[Kitapta 2.3.3] 1 Sayfa kadar

2.2.7 Steps of Neighborhood Methods

three very important considerations in the implementation of a neighborhood-based recommender system are (1) the normalization of ratings, (2) the computation of the similarity weights, and (3) the selection of neighbors. This section reviews some of the most common approaches for these three components, describes the main advantages and disadvantages of using each one of them, and gives indications on how to implement them.

Normalization of Ratings

[2.4.1.1 ve 2.4.1.2' i de ekle] In some cases, rating normalization can have undesirable effects. For instance, imagine the case of a user that gave only the highest ratings to the items he has purchased. Mean-centering would consider this user as "easy to please" and any rating below this highest rating (whether it is a positive or negative rating) would be considered as negative. However, it is possible that this user is in fact "hard to please" and carefully selects only items that he will like for sure. Furthermore, normalizing on a few ratings can produce unexpected results. For example, if a user has entered a single rating or a few identical ratings, his rating standard deviation will be 0, leading to undefined prediction values. Nevertheless, if the rating data is not overly sparse, normalizing ratings has been found to consistently improve the predictions [25, 29].

Comparing mean-centering with Z-score, as mentioned, the second one has the additional benefit of considering the variance in the ratings of individual users or items. This is particularly useful if the rating scale has a wide range of discrete values or if it is continuous. On the other hand, because the ratings are divided and multiplied by possibly very different standard deviation values, Z-score can be more sensitive than mean-centering and, more often, predict ratings that are outside the rating scale. Lastly, while an initial investigation found mean-centering and Z-score to give comparable results [25], a more recent one showed Z-score to have more significant benefits [29].

Finally, if rating normalization is not possible or does not improve the results, another possible approach to remove the problems caused by the rating scale variance is preference-based filtering. The particularity of this approach is that it focuses on predicting the relative preferences of users instead of absolute rating values. Since an item preferred to another one remains so regardless of the rating scale, predicting relative preferences removes the need to normalize the ratings. More information on this approach can be found in [12, 18, 32, 33].

Computation of Similarity Weights

The similarity weights play a double role in neighborhood-based recommendation methods: (1) they allow to select trusted neighbors whose ratings are used in the prediction, and (2) they provide the means to give more or less importance to these neighbors in the prediction. The computation of the similarity weights is one of the most critical aspects of building a neighborhood-based recommender system, as it can have a significant impact on both its accuracy and its performance.

[2.4.2.1, 2.4.2.2] cosine vector, person correlation, adjusted cosine, mean squared prediction

There are also some considerations: [en iyisi bu kismi ayri chapter yapip bunu subsubsection yap]

Significance of weights: 1 paragraph Variance of Ratings: 1 paragraph, mesela godfather'i herkes seviyor => IDF yap

Neigborhood Selection

The number of nearest-neighbors to select and the criteria used for this selection can also have a serious impact on the quality of the recommender system. The selection of the neighbors used in the recommendation of items is normally done in two steps: (1) a global filtering step where only the most likely candidates are kept, and (2) a per prediction step which chooses the best candidates for this prediction.

Prefiltering[aslinda subsubsec]: In large recommender systems that can have millions of users and items, it is usually not possible to store the (non-zero) similarities between each pair of users or items, due to memory limitations. Moreover, doing so would be extremely wasteful as only the most significant of these values are used in the predictions. The pre- filtering of neighbors is an essential step that makes neighborhood-based approaches practicable by reducing the amount of similarity weights to store, and limiting the number of candidate neighbors to consider in the predictions. There are several ways in which this can be accomplished:

- Top-n filtering: For each user or item, only a list of the N nearest-neighbors and their respective similarity weight is kept. To avoid problems with efficiency or accuracy, N should be chosen carefully. Thus, if N is too large, an excessive amount of memory will be required to store the neighborhood lists and predicting ratings will be slow. On the other hand, selecting a too small value for N may reduce the coverage of the recommendation method, which causes some items to be never recommended.

- Threshold filtering: Instead of keeping a fixed number of nearest-neighbors, this approach keeps all the neighbors whose similarity weight's magnitude is greater than a given threshold wmin. While this is more flexible than the previous filtering technique, as only the most significant neighbors are kept, the right value of wmin may be difficult to determine.

- Negative filtering: In general, negative rating correlations are less reliable than positive ones. Intuitively, this is because strong positive correlation between two users is a good indicator of their belonging to a common group (e.g., teenagers, science-fiction fans, etc.). However, although negative correlation may indicate membership to different groups, it does not tell how different are these groups, or whether these groups are compatible for some other categories of items.

Actual Prediction[aslinda subsubsec]: Once a list of candidate neighbors has been computed for each user or item, the prediction of new ratings is normally made with the k-nearest-neighbors, that is, the k neighbors whose similarity weight has the greatest magnitude. The choice of k can also have a significant impact on the accuracy and performance of the system.

As shown in Table 2.3, the prediction accuracy observed for increasing values of k typically follows a concave function. Thus, when the number of neighbors is restricted by using a small k (e.g., $k < 20$), the prediction accuracy is normally low. As k increases, more neighbors contribute to the prediction and the variance introduced by individual neighbors is averaged out. As a result, the prediction accuracy improves. Finally, the accuracy usually drops when too many neighbors are used in the prediction (e.g., $k > 50$), due to the fact that the few strong local relations are "diluted" by the many weak ones. Although a number of neighbors between 20 to 50 is most often described in the literature, see e.g. [24, 26], the optimal value of k should be determined by cross-validation.

On a final note, more serendipitous recommendations may be obtained at the cost of a decrease in accuracy, by basing these recommendations on a few very similar users. For example, the system could find the user most similar to the active one and recommend the new item that has received the highest rated from this user.

2.2.8 Conclusion

One of the earliest approaches proposed for the task of item recommendation is neighborhood-based recommendation, which ranks among the most popular methods for this problem. Although quite simple to describe and implement, this recommendation approach has several important advantages, including its ability to explain a recommendation with the list of the neighbors used, its computational and space efficiency which allows it to scale to large recommender systems, and its marked stability in an online setting where new users and items are constantly added. Another of its strengths is its potential to make serendipitous recommendations that can lead users to the discovery of unexpected, yet very interesting items.

In the implementation of a neighborhood-based approach, one has to make several important decisions. Perhaps the one having the greatest impact on the accu- racy and efficiency of the recommender system is choosing between a user-based and an item-based neighborhood method. In typical commercial recommender systems, where the number of users far exceeds the number of available items, item-based approaches are typically preferred since they provide more accurate recommendations, while being more computationally efficient and requiring less frequent updates. On the other hand, user-based methods usually provide more original recommendations, which may lead users to a more satisfying experience. Moreover, the different components of a neighborhood-based method, which include the normalization of ratings, the computation of the similarity weights and the selection of the nearest-neighbors, can also have a significant influence on the quality of the recommender system. For each of these components, several different alternatives are available. Although the merit of each of these has been described in this document and in the literature, it is important to remember that the "best" approach may differ from one recommendation setting to the next. Thus, it is important to evaluate them on data collected from the actual system, and in light of the particular needs of the application.

Finally, when the performance of a neighborhood-based approach suffers from the problems of limited coverage and sparsity, one may explore techniques based on dimensionality reduction or graphs. Dimensionality reduction provides a compact representation of users and items that captures their most significant features. An advantage of such approach is that it allows to obtain meaningful relations between pairs of users or items, even though these users have rated different items, or these items were rated by different users. On the other hand, graph-based techniques exploit the transitive relations in the data. These techniques also avoid the problems of sparsity and limited coverage by evaluating the relationship between users or items that are not "directly connected". However, unlike dimensionality reduction, graph-based methods also preserve some of the "local" relations in the data, which are useful in making serendipitous recommendations.

3 Collaborative Filtering

3.1 Introduction

[kopi peyst] Collaborative filtering recommender system (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 researchcommunity 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.

Because explicit feedback is not always available, some recommenders infer user preferences from the more abundant implicit feedback, which indirectly reflects opinion through observing user behavior [20]. Types of implicit feedback include purchase history, browsing history, search patterns, or even mouse movements. For example, a user who purchased many books by the same author probably likes that author. This chapter focuses on models suitable for explicit feedback. Nonetheless, we recognize the importance of implicit feedback, an especially valuable information source for users who do not provide much explicit feedback. Hence, we show how to address implicit feedback within the models as a secondary source of information.

In order to establish recommendations, CF systems need to relate two funda- mentally different entities: items and users. There are two primary approaches to facilitate such a comparison, which constitute the two main techniques of CF: the neighborhood approach and latent factor models. Neighborhood methods focus on relationships between items or, alternatively, between users. An item-item approach models the preference of a user to an item based on ratings of similar items by the same user. Latent factor models, such as matrix factorization (aka, SVD), comprise an alternative approach by transforming both items and users to the same latent factor space. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback.

Producing more accurate prediction methods requires deepening their founda- tions and reducing reliance on arbitrary decisions. In this chapter, we describe a variety of recent improvements to the primary CF modeling techniques. Yet, the quest for more accurate models goes beyond this. At least as important is the identification of all the signals, or features, available in the data. Conventional techniques address the sparse data of user-item ratings. Accuracy significantly improves by also utilising other sources of information. One prime example includes all kinds of temporal effects reflecting the dynamic, time-drifting nature of user-item interactions. No less important is listening to hidden feedback such as which items users chose to rate (regardless of rating values). Rated items are not selected at random, but rather reveal interesting aspects of user preferences, going beyond the numerical values of the ratings. Section 3.3 surveys matrix factorization techniques, which combine implementa- tion convenience with a relatively high accuracy. This has made them the preferred technique for addressing the largest publicly available dataset—the Netflix data.

3.2 Baseline

We are given ratings for m users (aka customers) and n items (aka products). We reserve special indexing letters to distinguish users from items: for users u; v, and for items i; j; l. A rating rui indicates the preference by user u of item i, where high values mean stronger preference. For example, values can be integers ranging from 1 (star) indicating no interest to 5 (stars) indicating a strong interest. We distinguish predicted ratings from known ones, by using the notation rOui for the predicted value of rui.

The scalar tui denotes the time of rating rui. One can use different time units, based on what is appropriate for the application at hand. For example, when time is measured in days, then tui counts the number of days elapsed since some early time point. Usually the vast majority of ratings are unknown. For example, in the Netflix data 99% of the possible ratings are missing because a user typically rates only a small portion of the movies. The (u,i) pairs for which rui is known are stored in the set $\mathcal{K} = \{(u,i)|r_{ui} isknown\}$ is knowng. Each user u is associated with a set of items denoted by R.u/, which contains all the items for which ratings by u are available. Likewise, R.i/ denotes the set of users who rated item i. Sometimes, we also use a set denoted by λ, which contains all items for which u provided an implicit preference (items that he rented/purchased/watched, etc.).

3.2.1 Baseline Predictors

CF models try to capture the interactions between users and items that produce the different rating values. However, much of the observed rating values are due to effects associated with either users or items, independently of their interaction. A principal example is that typical CF data exhibit large user and item biases—i.e., systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others. We will encapsulate those effects, which do not involve user-item interaction, within the baseline predictors (also known as biases). Because these predictors tend to capture much of the observed signal, it is vital to model them accurately. Such modeling enables isolating the part of the signal that truly represents user-item interaction, and subjecting it to more appropriate user preference models. Denote by μ the overall average rating. A baseline prediction for an unknown rating rui is denoted by bui and accounts for the user and item effects:

$$b_{ui} = \mu + b_u + b_i$$

The parameters bu and bi indicate the observed deviations of user u and item i, respectively, from the average. For example, suppose that we want a baseline predictor for the rating of the movie Titanic by user Joe. Now, say that the average rating over all movies, μ , is 3.7 stars. Furthermore, Titanic is better than an average movie, so it tends to be rated 0.5 stars above the average. On the other hand, Joe is a critical user, who tends to rate 0.3 stars lower than the average. Thus, the baseline predictor for Titanic's rating by Joe would be 3.9 stars by calculating 3.7 - 0.3 + 0.5. In order to estimate bu and bi one can solve the least squares problem

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right)$$

Here, the first term $\sum_{(u,i)\in\mathcal{X}} (r_{ui} - \mu + b_u + b_i)^2$ the given ratings. The regularizing term $\lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right)$ avoids overfitting by penalizing the magnitudes of the parameters. This least square problem can be solved fairly efficiently by the method of stochastic gradient descent.

For the Netflix data the mean rating μ is 3.6. As for the learned user biases (bu), their average is 0.044 with standard deviation of 0.41. The average of their absolute values (|bu|) is: 0.32. The learned item biases (bi) average to -0.26 with a standard deviation of 0.48. The average of their absolute values (|bi|) is 0.43.

An easier, yet somewhat less accurate way to estimate the parameters is by decoupling the calculation of the bi's from the calculation of the bu's. First, for each item i we set

$$b_i = \frac{\sum_{u \in \mathbb{R}(i)} (r_{ui} - \mu)}{\lambda_2 + |\mathrm{R}(i)|}$$

Then, for each user u we set:

$$b_u = \frac{\sum_{i \in \mathrm{R}(u)} (r_{ui} - \mu - b_i)}{\lambda_3 + |\mathrm{R}(u)|}$$

For Netflix data, the accuracy can be easily evaluated using RMSE:

$$\sqrt{\sum_{(u,i) \in Testset} (r_{ui} - \hat{r}_{ui})^2 \,/\, |TestSet|}$$

3.2.2 Matrix Factorization

Latent factor models approach collaborative filtering with the holistic goal to uncover latent features that explain observed ratings; examples include pLSA [14], neural networks [22], Latent Dirichlet Allocation [7], 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 [9]. However, applying SVD to explicit ratings in the CF domain raises dif- ficulties 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 [15, 24], 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 [4, 6, 10, 16, 21, 22, 26] 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 increas- ing 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

[kitapta 3.3.1, 1.5 sayfa ve gerekirse SVD++(with implicit feedback)]

Summary

In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from patterns of item ratings. High correspondence between item and user factors leads to recommendation of an item to a user. These methods deliver prediction accuracy superior to other published collaborative filtering techniques. At the same time, they offer a memory efficient compact model, which can be trained relatively easy. Those advantages, together with the implementation ease of gradient based matrix factorization model (SVD), made this the method of choice within the Netflix Prize competition.

What makes these techniques even more convenient is their ability to address several crucial aspects of the data. First, is the ability to integrate multiple forms of user feedback. One can better predict user ratings by also observing other related actions by the same user, such as purchase and browsing history. The proposed SVD++ model leverages multiple sorts of user feedback for improving user profiling.

Another important aspect is the temporal dynamics that make users' tastes evolve over time. Each user and product potentially goes through a distinct series of changes in their characteristics. A mere decay of older instances cannot adequately identify communal patterns of behavior in time changing data. The solution we adopted is to model the temporal dynamics along the whole time period, allowing us to intelligently separate transient factors from lasting ones. The inclusion of temporal dynamics proved very useful in improving quality of predictions, more than various algorithmic enhancements.

3.2.3 Neighborhood Models

The most common approach to CF is based on neighborhood models. Chapter 2 provides an extensive survey on this approach. Its original form, which was shared by virtually all earlier CF systems, is user-user based; see [13] for a good analysis. User-user methods estimate unknown ratings based on recorded ratings of like- minded users.

Later, an analogous item-item approach [18, 25] 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 [2, 25, 26]. 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; see also Sect. 3.5.2.2.

In general, latent factor models offer high expressive ability to describe various aspects of the data. Thus, they tend to provide more accurate results than neighborhood models. However, most literature and commercial systems (e.g., those of Amazon [18] and TiVo [1]) are based on the neighborhood models. The prevalence of neighborhood models is partly due to their relative simplicity. However, there are more important reasons for real life systems to stick with those models. First, they naturally provide intuitive explanations of the reasoning behind recommendations, which often enhance user experience beyond what improved accuracy may achieve. Second, they can provide immediate recommendations based on newly entered user feedback.

The structure of this section is as follows. First, we describe how to estimate the similarity between two items, which is a basic building block of most neighborhood techniques. Then, we move on to the widely used similarity-based neighborhood method, which constitutes a straightforward application of the similarity weights. We identify certain limitations of this similarity based approach. As a consequence, in Sect. 3.4.3 we suggest a way to solve these issues, thereby improving prediction accuracy at the cost of a slight increase in computation time.

[3.4.1 ve 3.4.2 banko] [3.4.3'te yeni bir neighborhood modeli anlatiyor, belki o eger kullanilirsa]

Summary

Collaborative filtering through neighborhood-based interpolation is probably the most popular way to create a recommender system. Three major components characterize the neighborhood approach: (1) data normalization, (2) neighbor selection, and (3) determination of interpolation weights.

Normalization is essential to collaborative filtering in general, and in particular to the more local neighborhood methods. Otherwise, even more sophisticated methods are bound to fail, as they mix incompatible ratings pertaining to different unnormalized users or items. We described a suitable approach to data normalization, based around baseline predictors.

Neighborhood selection is another important component. It is directly related to the employed similarity measure. Here, we emphasized the importance of shrinking unreliable similarities, in order to avoid detection of neighbors with a low rating support.

Finally, the success of neighborhood methods depends on the choice of the interpolation weights, which are used to estimate unknown ratings from neighboring known ones. Nevertheless, most known methods lack a rigorous way to derive these weights. We showed how the interpolation weights can be computed as a global solution to an optimization problem that precisely reflects their role.

[kullanirsan ve yer kalirsa 3.5]

[yer kalirsa ve matrix factorization kullanirsan 3.6]

[taxonomy vs olayina girersen 4]

4 Content-Based Filtering

4.1 Introduction

[kopi peyst from semantics aware cbf, 4]

Content-based recommender systems (CBRSs) rely on item and user descriptions (content) to build item representations and user profiles to suggest items similar to those a target user already liked in the past. The basic process of producing content-based recommendations consists in matching up the attributes of the target user profile, in which preferences and interests are stored, with the attributes of the items. The result is a relevance score that predicts the target user's level of interest in those items. Usually, attributes for describing an item are features extracted from metadata associated to that item, or textual features extracted directly from the item description. The content extracted from metadata is often too short and not sufficient to correctly define the user interests, while the use of textual features involves a number of complications when learning a user profile due to natural language ambiguity. Polysemy, synonymy, multi-word expressions, named entity recognition and disambiguation are inherent problems of traditional keyword-based profiles, which are not able to go beyond the usage of lexical/syntactic structures to infer the user interest in topics.

The ever increasing interest in semantic technologies and the availability of several open knowledge sources, such as Wikipedia, DBpedia, Freebase, and BabelNet have fueled recent progress in the field of CBRSs. Novel research works have introduced semantic techniques that shift from a keyword-based to a concept- based representation of items and user profiles. These observations make very relevant the integration of proper techniques for deep content analytics borrowed from Natural Language Processing (NLP) and Semantic Technologies, which is one of the most innovative lines of research in semantic recommender systems [61].

We roughly classify semantic techniques into top-down and bottom-up approaches. Top-down approaches rely on the integration of external knowledge, such as machine readable dictionaries, taxonomies (or IS-A hierarchies), thesauri or ontologies (with or without value restrictions and logical constraints), for annotating items and representing user profiles in order to capture the semantics of the target user information needs. The main motivation behind top-down approaches is the challenge of providing recommender systems with the linguistic knowledge and common sense knowledge, as well as the cultural background which characterize the human ability of interpreting documents expressed in natural language and reasoning on their meaning.

On the other side, bottom-up approaches exploit the so-called geometric metaphor of meaning to represent complex syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. According to this metaphor, each word (and each document as well) can be represented as a point in a vector space. The peculiarity of these models is that the representation is learned by analyzing the context in which the word is used, in a way that terms (or documents) similar to each other are close in the space. For this reason bottom-up approaches are also called distributional models. One of the great virtues of these approaches is that they are able to induce the semantics of terms by analyzing their use in large corpora of textual documents using unsupervised mechanisms, as evidenced by the recent advances of machine translation techniques [52, 83].

This chapter describes a variety of semantic approaches, both top-down and bottom-up, and shows how to leverage them to build a new generation of semantic CBRSs that we call semantics-aware content-based recommender systems.

xd

This section reports an overview of the basic principles for building CBRSs, the main techniques for representing items, learning user profiles and providing recommendations. The most important limitations of CBRSs are also discussed, while the semantic techniques useful to tackle those limitations are introduced in the next sections.

The high level architecture of a content-based recommender system is depicted in Fig. 4.1. The recommendation process is performed in three steps, each of which is handled by a separate component:

[Figure 4.1 ya da benzeri ekle]

- 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 [86], 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 [113] 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 [57]), 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 systems [6, 118]. 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 ua (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 anno- tations [51] 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 (i.e., inferring features the user is not interested in [58]). 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", 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 [111]. Explicit feedback has the advantage of simplicity, albeit the adoption of numeric/symbolic scales increases the cognitive load on the user, and may not be adequate for catching user's feeling about items. Implicit feedback methods are based on assigning a relevance score to specific user actions on an item, such as saving, discarding, printing, bookmarking, etc. The main advantage is that they do not require a direct user involvement, even though biasing is likely to occur, e.g. interruption of phone calls while reading.

In order to build the profile of the active user ua, the training set TRa for ua must be defined. TRa is a set of pairs <Ik, rk> i, where rk is the rating provided by ua on the item representation Ik. Given a set of item representation labeled with ratings, the PROFILE LEARNER applies supervised learning algorithms to generate a predictive model—the user profile—which is usually stored in a profile repository for later use by the FILTERING COMPONENT. After the user profile has been learned, the FILTERING COMPONENT predicts whether a new item is likely to be of interest for the active user, by comparing features in the item representation to those in the representation of user preferences (stored in the user profile).

User tastes usually change in time, therefore up-to-date information must be maintained and provided to the PROFILE LEARNER in order to automatically update the user profile. Further feedback is gathered on generated recommendations by letting users state their satisfaction or dissatisfaction with items in La. After gathering that feedback, the learning process is performed again on the new training set, and the resulting profile is adapted to the updated user interests. The iteration of the feedback-learning cycle over time enables the system to take into account the dynamic nature of user preferences.

4.1.1 Text to Vector Space Model

[4.2.1 tf-idf, cosine similarity falan]

4.1.2 Methods for Learning User Profile

[4.2.2.1 eger naive bayes kullanirsan 4.2.2.1, 4.2.2.2 bilmiyorum, 4.2.2.3 kesin]

4.1.3 Advantages and Drawbacks of Content-Based Filtering

The adoption of the content-based recommendation paradigm has several advan- tages 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. Furthermore, CBRSs based on a string matching approach suffer from problems of:
  - POLYSEMY, the presence of multiple meanings for one word;
  - SYNONYMY, multiple words with the same meaning;
  - MULTI-WORD EXPRESSIONS, the difficulty to assign the correct properties to a sequence of two or more words whose properties are not predictable from the properties of the individual words;
  - ENTITY IDENTIFICATION or NAMED ENTITY RECOGNITION, the difficulty to locate and classify elements in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, etc.

  - ENTITY LINKING or NAMED ENTITY DISAMBIGUATION, the difficulty of determining the identity (often called the reference) of entities mentioned in text.

- 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 lack of 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.

[4.3 Eger textten anlam cikarma falan yaparsan eklersin] [4.4 Eger textten anlam cikarma falan yaparsan eklersin, bir oncekinin daha basiti ama iste insan olmadan halletme gibi] [4.5 ayni sekilde]

4.2 Conclusion

[Biraz 4.6 dan alabilirsin ama hepsi alakali degil]

5 Evaluation Metrics

5.1 Introduction

[kopi peyst] In order to give a formal definition of the item recommendation task, we introduce the following notation. The set of users in the recommender system will be denoted by U, and the set of items by I. Moreover, we denote by R the set of ratings recorded in the system, and write S the set of possible values for a rating (e.g., S = [1..5] or S = like; dislike). Also, we suppose that no more than one rating can be made by any user u ∈ U for a particular item i ∈ I and write $r_u i$ this rating. To identify the subset of users that have rated an item i, we use the notation $U_i i$. Likewise, I_u represents the subset of items that have been rated by a user u. Finally, the items that have been rated by two users u and v, i.e. $I_u \cap I_v$, is an important concept in our presentation, and we use Iuv to denote this concept. In a similar fashion, $U_i j$ is used to denote the set of users that have rated both items i and j.

Two of the most important problems associated with recommender systems are the rating prediction and top-N recommendation problems. The first problem is to predict the rating that a user u will give his or her unrated item i. When ratings are available, this task is most often defined as a regression or (multi-class) classification problem where the goal is to learn a function f : U × I → S that predicts the rating f(u, i) of a user u for a new item i. Accuracy is commonly used to evaluate the performance of the recommendation method. Typically, the ratings R are divided into a training set $R_t rain$ used to learn f , and a test set Rtest used to evaluate the prediction accuracy. Two popular measures of accuracy are the Mean Absolute Error (MAE):

$$\mathrm{MAE}(f) = \frac{1}{|\mathcal{R}_{test}|} \sum_{r_{ui} \in \mathcal{R}_{test}} |f(u,i) - r_{ui}|$$

and the Root Mean Squared Error (RMSE):

$$\mathrm{RMSE}(f) = \sqrt{\frac{1}{|\mathcal{R}_{test}|} \sum_{r_{iu}} (f(u,i) - r_{ui})^2}$$

When ratings are not available, for instance, if only the list of items purchased by each user is known, measuring the rating prediction accuracy is not possible. In such cases, the problem of finding the best item is usually transformed into the task of recommending to an active user ua a list L(u_a) containing N items likely to interest him or her [15, 59]. The quality of such method can be evaluated by splitting the items of I into a set Itrain, used to learn L, and a test set Itest. Let T(u) $\in I_u \cap I_test$ be the subset of test items that a user u found relevant. If the user responses are binary, these can be the items that u has rated positively. Otherwise, if only a list of purchased or accessed items is given for each user u, then these items can be used as T(u). The performance of the method is then computed using the measures of precision and recall:

$$\text{Precision}(L) = \frac{1}{|u|} \sum_{u \in \mathcal{U}} |L(u) \cap T(u)| / |L(u)|$$

$$\text{Recall}(L) = \frac{1}{|u|} \sum_{u \in \mathcal{U}} |L(u) \cap T(u)| / |T(u)|$$

A drawback of this task is that all items of a recommendation list L.u/ are considered equally interesting to user u. An alternative setting, described in [15], consists in learning a function L that maps each user u to a list L.u/ where items are ordered by their "interestingness" to u. If the test set is built by randomly selecting, for each user u, a single item iu of Iu, the performance of L can be evaluated with the Average Reciprocal Hit-Rank (ARHR):

$$\text{ARHR}(L) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{\text{rank}\,(i_u, L(u))}$$

where rank(iu ; L(u)) is the rank of item iu in L(u). A more extensive description of evaluation measures for recommender systems can be found in Chap. 8 of this book.

6 Implementation

6.1 Introduction

Recommender Systems (RSs) are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user [RRS15].

Suggestions and items depend on the field that recommder system is applied. For example, for the topic of news article recommenders, the aim will most likely be suggesting news to the readers. In the field of job recommenders though, these suggestions can be bidirectional. Meaning that, job postings can be suggested to applicants or resumes can be recommended to the human resources team of a company.

See Table 6.1, Figure 6.1, Figure 6.2, Figure 6.3.

Table 6.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

$$\begin{tikzcd}
R_1 \arrow[r, no head] \arrow[d, no head] & R_2 \arrow[r, no head] \arrow[d, no head] & R_5 \\
R_3 \arrow[r, no head] & R_4 &
\end{tikzcd}$$

Figure 6.1: An example for a simple drawing.

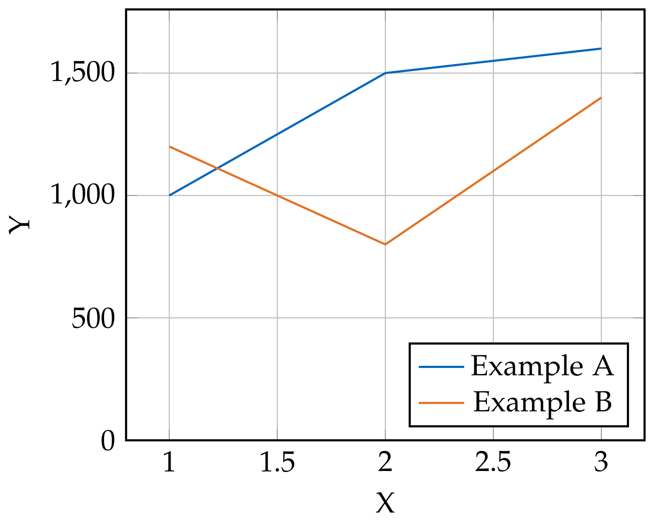

Figure 6.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 6.3: An example for a source code listing.

List of Figures

List of Tables

Bibliography

[Bee+16] J. Beel, B. Gipp, S. Langer, and C. Breitinger. “Research-paper recommender systems: a literature survey.” In: *International Journal on Digital Libraries* 17.4 (Nov. 2016), pp. 305–338. ISSN: 1432-1300. DOI: `10.1007/s00799-015-0156-0`.

[Par+12] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim. “A literature review and classification of recommender systems research.” In: *Expert Systems with Applications* 39.11 (2012), pp. 10059–10072. ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2012.02.038`.

[RRS15] F. Ricci, L. Rokach, and B. Shapira. “Recommender Systems: Introduction and Challenges.” In: *Recommender Systems Handbook*. Ed. by F. Ricci, L. Rokach, and B. Shapira. Boston, MA: Springer US, 2015, pp. 1–34. ISBN: 978-1-4899-7637-6. DOI: `10.1007/978-1-4899-7637-6_1`.