RS Module 1 Notes

Module 1.1: Introduction

Recommender System Functions:

1. **Increase the number of items sold:** This is probably the most important function for a commercial RS, i.e., to be able to sell an additional set of items compared to those usually sold without any kind of recommendation.
2. **Sell more diverse items:** Another major function of a RS is to enable the user to select items that might be hard to find without a precise recommendation. For instance, in a movie RS such as Netflix, the service provider is interested in renting all the DVDs in the catalogue, not just the most popular ones.
3. **Increase the user satisfaction:** A well designed RS can also improve the experience of the user with the site or the application. The user will find the recommendations interesting, relevant and, with a properly designed human-computer interaction, she will also enjoy using the system.
4. **Increase user fidelity:** A user should be loyal to a Web site which, when visited, recognizes the old customer and treats him as a valuable visitor. This is a normal feature of a RS since many RSs compute recommendations, leveraging the information acquired from the user in previous interactions, e.g., her ratings of items.
5. **Better understand what the user wants:** Another important function of a RS, which can be leveraged to many other applications, is the description of the user's preferences, either collected explicitly or predicted by the system. The service provider may then decide to reuse this knowledge for a number of other goals such as improving the management of the item's stock or production.

Data & Knowledge Sources

1. **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.
2. **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.

3. **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.

Recommendations Techniques

1. **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.
2. **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. Collaborative filtering is considered to be the most popular and widely implemented technique in RS.
3. **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.
4. **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.
5. **Constraint-based:** systems are another type of knowledge-based RSs. In terms of used knowledge, both systems are similar: user requirements are collected; repairs for inconsistent requirements are automatically proposed in situations where no solutions could be found; and recommendation results are explained. The major difference lies in the way solutions are calculated. Case-based recommenders determine recommendations on the basis of similarity metrics whereas constraint-based recommenders predominantly exploit predefined knowledge bases that contain explicit rules about how to relate customer requirements with item features.
6. **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 system.

Understanding Ratings

Ratings represent a user's explicit or implicit feedback for an item, generally captured in a matrix format:

1. **Explicit Ratings**: Direct user feedback, like giving a movie 4 stars.
   The main problems with explicit ratings are that such ratings require additional efforts from the users of the recommender system and users might not be willing to provide such ratings as long as the value cannot be easily seen. Thus, the number of available ratings could be too small, which in turn results in poor recommendation quality.
2. **Implicit Ratings**: Inferred from behavior, such as how long a user watches a video or the number of clicks. Although implicit ratings can be collected constantly and do not require additional efforts from the side of the user, one cannot be sure whether the user behavior is correctly interpreted.

Ratings are typically arranged in a **User-Item Matrix**, where rows represent users and columns represent items. The matrix is often sparse because users don’t rate every item.

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:

1. **Numerical ratings** such as the 1-5 stars provided in the book recommender associated with Amazon.com.
2. **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).
3. **Binary ratings** that model choices in which the user is simply asked to decide if a certain item is good or bad, example Likes and Dislikes on a YouTube video.
4. **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).

Example: In a movie recommendation system, a matrix with users as rows and movies as columns will store ratings (from 1 to 5) in the respective cell.

Applications of Recommendation Systems

Recommender systems are widely used across industries for:

1. **E-commerce**: Amazon uses collaborative filtering to recommend products based on

users' past purchases and browsing history.
2. **Entertainment**: Netflix and YouTube recommend videos or movies based on your watching history and ratings.
3. **Social Media**: Facebook and Instagram suggest friends or posts by analyzing user interactions and connections.
4. **Content Discovery**: News sites like Google News recommend articles based on reading history.

Issues with Recommender Systems

Some of the key issues faced by recommender systems include:

1. **Cold Start Problem**: When a new user or item enters the system, there is no previous data available to generate recommendations.
2. **Scalability**: Large datasets with millions of users and items require efficient algorithms for real-time recommendations.
3. **Sparsity**: The user-item matrix is often sparse, meaning most users rate only a small subset of available items, making accurate predictions harder.
4. **Bias**: Recommender systems can be biased towards popular items or those with many ratings, potentially leading to a lack of diversity in recommendations.

Experimental Settings:

1. **Offline Experiments:** An offline experiment is performed by using a pre-collected data set of users choosing or rating items. Using this data set we can try to simulate the behavior of users that interact with a recommendation system. In doing so, we assume that the user behavior when the data was collected will be similar enough to the user behavior when the recommender system is deployed, so that we can make reliable decisions based on the simulation. Offline experiments are attractive because they require no interaction with real users, and thus allow us to compare a wide range of candidate algorithms at a low cost. The downside of offline experiments is that they can answer a very narrow set of questions, typically questions about the prediction power of an algorithm.
2. **User Studies:** A user study is conducted by recruiting a set of test subjects, and asking them to perform several tasks requiring an interaction with the recommendation system. While the subjects perform the tasks, we observe and record their behavior, collecting any number of quantitative measurements, such as what portion of the task was completed, the accuracy of the task results, or the time taken to perform the task. In many cases we can ask qualitative questions before, during, and after the task is completed. User studies can perhaps answer the widest set of questions of all three experimental settings that we survey here. Unlike offline experiments this setting allows us to test the behavior of users when interacting with the recommendation system, and the influence of the recommendations on user behavior. Primarily, user studies are very expensive to conduct; collecting a large set of subjects and asking them to perform a large enough set of tasks is costly.
3. **Online Evaluation:** In many realistic recommendation applications the designer of the system wishes to influence the behavior of users. We are therefore interested in measuring the change in user behavior when interacting with different recommendation systems. Thus, the experiment that provides the strongest evidence as to the true value of the system is an online evaluation, where the system is used by real users that perform real tasks. It is most trustworthy to compare a few systems online, obtaining a ranking of alternatives, rather than absolute numbers that are more difficult to interpret. In some cases, such experiments are risky. For example, a test system that provides irrelevant recommendations, may discourage the test users from using the real system ever again. Thus, the experiment can have a negative effect on the system, which may be unacceptable in commercial applications.

Recommendation System Properties

As different applications have different needs, the designer of the system must decide on the important proper- ties to measure for the concrete application at hand. Some of the properties can be traded-off, the most obvious example perhaps is the decline in accuracy when other properties (e.g. diversity) are improved. It is important to understand and evaluate these trade-offs and their effect on the overall performance.

1. **User Preferences**: When we wish to improve a system, it is important to know why people favor one system over the other. Typically, it is easier to understand that when com- paring specific properties. Therefore, while user satisfaction is important to mea-
8 Evaluating Recommendation Systems 273
sure, breaking satisfaction into smaller components is helpful to understand the system and improve it.
2. **Prediction Accuracy**: At the base of the vast majority of recommender systems lie a prediction engine. This engine may predict user opinions over items (e.g. ratings of movies) or the probability of usage (e.g. purchase). A basic assumption in a recommender system is that a system that provides more accurate predictions will be preferred by the user. Thus, many researchers set out to find algorithms that provide better predictions. Prediction accuracy is typically independent of the user interface, and can thus be measured in an offline experiment.
3. **Coverage**: As the prediction accuracy of a recommendation system, especially in collaborative filtering systems, in many cases grows with the amount of data, some algorithms may provide recommendations with high quality, but only for a small portion of the items where they have huge amounts of data. It is mainly of 2 types: item-space coverage and user-space coverage.
4. **Confidence**: Confidence in the recommendation can be defined as the system's trust in its recom- mendations or predictions. As we have noted above, collaborative filtering recommenders tend to improve their accuracy as the amount of data over items grows. Similarly, the confidence in the predicted property typically also grows with the amount of data. In many cases the user can benefit from observing these confidence scores. When the system reports a low confidence in a recommended item, the user may tend to further research the item before making a decision. For example, if a system recommends a movie with very high confidence, and another movie with the same rating but a lower confidence, the user may add the first movie immediately to the watching queue, but may further read the plot synopsis for the second movie, and perhaps a few movie reviews before deciding to watch it.
5. **Trust**: While confidence is the system trust in its ratings, in trust we refer here to the user's trust in the system recommendation5. For example, it may be beneficial for the sys- tem to recommend a few items that the user already knows and likes. This way, even though the user gains no value from this recommendation, she observes that the system provides reasonable recommendations, which may increase her trust in the system recommendations for unknown items. Another common way of enhancing trust in the system is to explain the recommendations that the system provides. Trust in the systems is also called the credibility of the system.
6. **Novelty**: Novel recommendations are recommendations for items that the user did not know about. In applications that require novel recommendation, an obvious and easy to implement approach is to filter out items that the user already rated or used. However, in many cases users will not report all the items they have used in the past. Thus, this simple method is insufficient to filter out all items that the user already knows.
7. **Serendipity**: Serendipity is a measure of how surprising the successful recommendations are. For example, if the user has rated positively many movies where

a certain star actor appears, recommending the new movie of that actor may be novel, because the user may not know of it, but is hardly surprising. Of course, random recommendations may be very surprising, and we therefore need to balance serendipity with accuracy. One can think of serendipity as the amount of relevant information that is new to the user in a recommendation. For example, if following a successful movie recommendation the user learns of a new actor that she likes, this can be considered as serendipitous.

8. **Diversity**: Diversity is generally defined as the opposite of similarity. In some cases suggesting a set of similar items may not be as useful for the user, because it may take longer to explore the range of items. Consider for example a recommendation for a vacation [55], where the system should recommend vacation packages. Presenting a list with 5 recommendations, all for the same location, varying only on the choice of hotel, or the selection of attraction, may not be as useful as suggesting 5 different locations. The user can view the various recommended locations and request more details on a subset of the locations that are appropriate to her.

Other Properties Include:

1. **Utility**
2. **Risk**
3. **Robustness**
4. **Privacy**
5. **Adaptivity**
6. **Scalability**

The Cold-Start Problem

The cold-start problem arises when there is insufficient data for new users or items, making it difficult to generate accurate recommendations. It affects three areas:

- **New User Problem**: No historical data exists for new users, making it hard to recommend relevant items.
- **New Item Problem**: New items don’t have user interactions, so recommending them is challenging.
- **Cold Start Solution**: Gathering user preferences via onboarding surveys, or using content-based filtering to recommend items based on item attributes.

Example: Spotify asks new users about their favorite genres and artists to overcome the cold-start problem.

Evaluation Metrics

Evaluation metrics are critical to measure how well a recommender system performs. Commonly used metrics include:

1. **Precision**: Measures the proportion of recommended items that are relevant.

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

2. **Recall**: Measures the proportion of relevant items that were recommended.

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

3. **F1-Score**: Harmonic mean of precision and recall, providing a single metric that balances both.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4. **Root Mean Squared Error (RMSE)**: Evaluates the accuracy of rating predictions. It measures how far the predicted ratings are from actual ratings.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

5. **Mean Absolute Error (MAE)**: Another metric for evaluating prediction accuracy by calculating the average of the absolute differences between predicted and actual ratings.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Other Metrics

1. **Fairness**: Ensures that the recommender system doesn't favor specific groups of users or items, providing equal representation. **Example**: A music recommendation system should not favor popular genres like pop at the expense of less popular genres like jazz.
2. **Coverage**: Measures the proportion of items recommended by the system compared to the total number of items available.

$$Coverage = \frac{\text{Recommended Items}}{\text{Total Items in the Dataset}}$$

3. **Diversity**: Ensures that the system recommends a wide variety of items, preventing a concentration on a small set of popular items. **Example**: Recommending different types of movies (comedy, drama, action) rather than only recommending action movies.
4. **Novelty**: Measures how often new or previously unknown items are recommended to users, promoting discovery of new content. **Example**: A news recommendation system that suggests recent or less-known articles.

5. **Serendipity**: Ensures that the recommended items pleasantly surprise the user. It refers to the system's ability to suggest items that are unexpected but liked. **Example**: Recommending a documentary to a user who usually watches action movies but enjoys the new suggestion.

Ranking Measures

Ranking measures evaluate how well a system orders items in the list of recommendations, focusing on the position of relevant items in the ranked list.

1. **NDPM (Normalized Distance-based Performance Measure)**: Focuses on how far the relevant items are from the top of the recommendation list. Lower distances mean better performance.

$$NDPM = \frac{1}{N} \sum_{i} \frac{d(i)}{|RankedList|}$$

2. **Spearman's ρ (Spearman's Rank Correlation)**: Measures how well the predicted rankings of items match the true rankings. A score of +1 indicates perfect correlation, while -1 indicates a reverse order.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

3. **R-Score (Recall at K)**: Measures recall at a specific position K in the ranked recommendation list. It focuses on whether relevant items appear within the top K recommendations.
4. **MAP (Mean Average Precision)**: Measures the precision at different cutoff points in the ranked list, averaged across all users.

$$MAP = \frac{1}{|U|} \sum_{u} \frac{1}{N_u} \sum_{k=1}^{N_u} P(k)$$

5. **NDCG (Normalized Discounted Cumulative Gain)**: Measures the relevance of items, with higher weights given to top-ranked items. It discounts the relevance score of items appearing lower in the ranking.

$$DCG = \sum \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

6. **MRR (Mean Reciprocal Rank)**: The average of reciprocal ranks of the first relevant item in each ranked list. Higher MRR means the relevant items tend to appear earlier in the recommendation lists.

$$MRR = \frac{1}{|U|} \sum_{i=1}^{|U|} \frac{1}{rank_i}$$

Summary

Recommender systems aim to personalize user experiences by leveraging data sources like user-item interactions and ratings. Key approaches such as content-based filtering, collaborative filtering, demographic methods, and hybrid techniques are explored. Critical challenges like cold-start issues, sparsity, and scalability, importance of fair and unbiased recommendations. Evaluation methods, including precision, recall, RMSE, and ranking measures like NDCG. Real-world examples demonstrate the practical application of these concepts.