AIW & Information Retrieval (UE18CS322)
Unit 5

Aronya Baksy

May 2021

1 Recommender Systems

- Software agents that elicit customer preferences, and make recommendations accordingly, while improving the quality of decisions made by the customer.
- Recommendations are provided to *users*. The product being recommended is called an *item*.
- Recommendation analysis is based on interactions (past or present) between item and user (except for Knowledge-based RS where user supplies constraints on items).
- Principle behind recommender system: learn **dependencies** between user-centric and item-centric **behaviours**. Such a model is called a *neighbourhood model*, belonging to the family of *collaborative filtering models*.
- In *content-based RS*, the user ratings are modelled on the basis of the attributes of the individual items they have rated/accessed in the past.
- In *knowledge-based RS*, users interactively specify their interests, and the user specification is combined with domain knowledge to provide recommendations
- The Recommender System problem can be formulated in either of 2 ways:
  - **Prediction**: Predicting a rating value for a particular user-item combination. ($m \times n$ matrix of m user ratings for n items, complete the missing values in the matrxi)
  - **Ranking**: Recommend top k users to target for an item, or recommend top k items to a user.

1.1 Goals of a recommender system

- **Relevance**: Generate interesting and relevant recommendations to the user. This is the primary operational goal, but not useful in isolation
- **Novelty**: Items recommended must be new to user, not seen before. (leads to increase in sales diversity)
- **Serendipity**: aka the *luck* factor, the items recommended are *surprising* (not only novel) to the user. (e.g.: I like Indian, but RS recommends Ethiopian which is similar but surprising to me). Increases sales value, diversity, opens new scopes up needs to be balanced with relevance.
- **Increased Diversity**: Items recommended must be diverse (increases likelihood of user selecting atleast one item, hence increase business value)
- **Soft Goals**: Increased usability, increased customer retention

2 Types of Recommender System Models

2.1 Collaborative Filtering

- Given a matrix of ratings, the CF model tries to fill in the missing rating values using the *collaborative power* of the specified ratings.
- Main challenge is sparse nature of ratings matrix (most values are unspecified)
- Most CF models leverage item-item or user-user correlations, or both.
- **Memory-based**:
  - Ratings of user-item combinations are predicted using neighbourhoods.
  - Simple and easy to explain, but do not work well with sparse matrices.
  - Neighbourhoods are defined as:
    1. **User-based**: Determine users similar to the target user, and based on these similar users' preferences determine the missing rating values
    2. **Item-based**: Find items most similar to the target item. The target user's ratings on these similar items is used to calculate the unknown rating for the item.
- **Model-based**:
  - Using machine learning models in a predictive context.
  - Models that involve parameters learn these parameter values in a optimization framework
  - e.g.: Rule-based models, Decision Trees, Bayesian methods
- In a CF problem, there is no distinction between dependent and independent variables, or between training and testing rows in the data.
- The CF problem is therefore said to be *transductive*. A CF model will not be able to predict ratings for a new user if their ratings are added to the matrix after the model is trained.

2.1.1 Rating Scales

- Ratings can be either continuous (rare) or discrete. Discrete ratings can be in the form of numbers (0-5 or -2 to 2, etc., also called an *interval scale*), or categorical values (strongly disagree, disagree, neutral, agree, strongly agree), also called the *ordinal scale.*
- In an **unbalanced** rating scale, there are more positive ratings than negative ratings within the scale (e.g. Netflix rating system)
- In a **binary rating scale**, users express only like or dislike behaviour, and no other information (e.g. Stanford course feedback form).
- In a **unary rating scale**, user express only their preference for a product, but not any dislike (e.g. buying a product at a retail store indicates preference for it, but not buying does not indicate dislike, hence cannot use binary scale)
- Ratings matrices generated by unary rating scales are also called *positive preference utility matrices*, or *implicit feedback matrices* as they are generated by user *actions* and not ratings.
- Substituting missing values in either explicit or implicit ratings is not recommended as it introduces bias.

2.2 Content-based Recommender System

- Content refers to the description of items in the database.
- CBRS is useful when no info about user ratings is available.
- In content-based methods, the item descriptions, which are labeled with ratings, are used as training data to create a user-specific modeling problem.
- For each user, the training documents correspond to the descriptions of the items they have bought or rated
- Content based methods work well even when there is no past history of ratings for an item. A CBRS leverages ratings made for similar items along with info from the item description to make recommendations.
- **Disadvantages**:
  - CBRS tend to provide **obvious** recommendations, reducing diversity. (e.g. if a user has never consumed an item with a particular set of keywords, such an item has no chance of being recommended)
  - Not effective at providing recommendations for new users. This isbecause the model for the target user needs to use the history of their ratings.

2.3 Knowledge-based Recommender Systems

- KBRS are used for infrequently bought products where ratings are not available in large quantities (i.e. the cold start problem)
- KBRS performs recommendations not on the basis of ratings, but on the basis of similairites between user requirements and item description.
- A *Knowledge base* is a set of rules and similarity functions that govern how retrieval of items is to be done given the user requirements.
- Based on the interface, types of KBRS:
  1. **Constraint-based**: Users specify constraints on the item attributes. Rules may include these user-specified ones, as well as others created by the system itself that link user and item attributes. Users iteratively modify their requirements until the desired result is obtained.
  2. **Case-based**: Specific cases are specified by the user as targets or anchor points, and similarity metrics are used to retrieve similar items to these cases. Similarity metrics comprise the domain knowledge of the system.
- Based on the level of interactivity, types of KBRS:
  1. **Conversational**: In this case, the user preferences are determined iteratively in the context of a feedback loop. Used for complex domains
  2. **Search-based**: User preferences are elicited using a fixed sequence of questions.
  3. **Navigation-based**: aka critiquing systems, where the user supplies a set of change requests that are used to modify the currently recommended item. With an iterative sequence of change requests the user can arrive at the correct item.

2.4 Utility-based Recommender Systems

- The utility function is a function of the item attributes that calculates the probability of an user liking the item.
- The difference between utility-based and other types is that the utility function is known before hand (*a priori*) and is static.

2.5 Demographic Recommender Systems

- The demographic information about the user is leveraged to learn classifiers that can map specific demographics to ratings or buying behaviours.
- In many cases, demographic information can be combined with additional context to guide the recommendation process to give rise to *context-based* recommender systems.
- Rule-based classifiers are often used to relate the demographic profile to buying behavior in an interactive way
- Although demographic recommender systems do not usually provide the best results on their own, they add significantly to the power of other recommender systems as a component of hybrid or ensemble models.

2.6 Hybrid and Ensemble Systems

- In many cases where a wider variety of inputs is available, different types of recommender systems may be used for the same task.
- Ensemble-based recommender systems are able to combine not only the power of multiple data sources, but they are also able to improve the effectiveness of a particular class of recommender systems by combining multiple models of the same type.

3 Domain-specific Challenges

- **Context-based** or **Context-aware** recommender systems take into account various types of context information (location, season, time of day, etc.) while making recommendations
- Item ratings may either evolve with time, or be dependent on a specific time of day/month/year.
- **Temporal** recommender systems take into account time as a factor when making recommendations. This is done either by introducing time as an explicit parameter in a CF system, or by viewing them as a special case of context-aware recommender systems.
- Click-streams on the internet are a form of time series data that can be used to make recommendations about user activity. Discrete sequential pattern mining and HMMs are some models used in such cases.
- **Location-based recommendations** can be of two main types:
  1. *User-specific location*, also called preference locality
  2. *Item-specific location*, also referred to as travel locality
- Location-based recommender systems are designed using various heuristics, as compared to context-aware recommender systems.
- **Social recommender systems** are based on network structures, social cues and tags, or a combination of these various network aspects.
- Social RS that recommend nodes and links using the network structure, make use of various ranking algorithms (such as PageRank) that are personalized to a particular query or user.
- Link prediction is another problem prevalent in social RS (the aim is to find friends, i.e. potential links of interest for one given node in the network)
- Collective classificaiton is the problem wherein the aim is, given a collection of nodes of interest, to use these as training data and find other nodes of interest.
- Product recommendations with the help of social cues and network connections is called *viral marketing.*
- Determining influential and topically relevant entities within a network is called *influence analysis*

- Trust information is incorporated into social recommender systems in direct or indirect ways through feedback mechanisms (reviews etc.)
- Social tags are useful as they provide info. about the interests of both the user and the content of the item because the tag is associated with both.
- Context-aware recommendation systems can use social tag info to make more robust recommendations.

4 Advanced Problems in Recommender System

4.1 Cold Start

- Cold start occurs in CF based systems where number of initially available ratings is small.
- Content and knowledge based systems are more robust in this case, but can also suffer from cold start when sufficient knowledge is not available.

4.2 Attack-Resistant Systems

- Attacks may be of various forms such as:
  - Vendor submits inflated ratings for their own products to the recommendation service
  - Vendor submits negative ratings/reviews for competing products
- RS algorithms are needed that are resistant to such forms of attack

4.3 Group Recommender Systems

- RS that are targeted towards groups of users instead of a single user (e.g. deciding a movie for a group, deciding music to play at a gym etc.)
- Early models in this area simply aggregated preferences of individual users to create group recommendations.
- The latest research in this domain moves away from simple averaging strategies, and towards modelling interactions between members of a group, because individuals have impact on each other's taste.

4.4 Multi-Critera Recommender System

- A user's rating is not modelled as a single number, but as a real vector that corresponds to item's utility towards the user based on multiple criteria
- In addition to this multi-criteria rating, users may or may not submit an overall rating as well.
- IF the overall rating is present then similarity has to be with respect to both overall as well as multi-criteria rating, but not both
- If overall rating is absent then the system must present ranked list of items bsaed on multiple criteria, which becomes more difficult.

4.5 Active Learning

- In an active learning setup, a learning algorithm can interactively query a user to label new data points with the desired
- Active learning recommenders encourage users to enter ratings in order to populate the data and avoid cold start (provide incentives to rate)
- Users can be asked to rate items similar to the ones already rated (not helpful), or rate completely unrelated items (helpful for the data, but user may not be aware enough to rate it)

4.6 Privacy Concerns

- The tradeoff is between the sensitive and personal nature of data that is collected from real users, and the release of real data that is crucial for algorithmic advancements

4.7 Application Domains

- Web search, music/retail/content recommendation, news recommendations, computational advertising, etc.
- Domains are web-centric in nature, hence long-term user data (needed to track user interests) may not be available.
- Changes to off-the-shelf techniques as well as new models such as multi-armed bandit recommender systems can mitigate these challenges.

5 Collaborative Filtering

5.1 Introduction

- The basic idea of this class of algorithms is to exploit the "wisdom of the crowds", to use behavioural patterns of a community to determine suitable recommendations for an individual user or in the context of a given reference item.
- CF techniques do not rely on content features and metadata of the items when determining the relevance of an item
- In its pure form, CF relies solely on a collection of explicit or implicit preference signals of users towards items. (i.e., ratings)
- Given the past ratings of an individual user, the goal is then to determine the (current) relevance of each item by combining these individual signals with the patterns of the community

5.2 CF as a matrix completion task

- The input to the CF system is a matrix, where each row corresponds to one user and the columns correspond to recommendable items. Each entry of the matrix is the rating by the user of one particular item.
- The problem statement of CF is to find the missing entries

5.2.1 User-based CF

- General idea: that users "who agreed in the past will probably agree again"
- To predict the score given by user a for item p, the following 2 steps are done:
  - Identify the K users who exhibit the most similar rating patterns to the user a
  - Combine the ratings given by these K users for the item p into the predicted rating for item p by user a
- We define two users a and b. We define P as the set of all items rated by both a and b. Then the similarity score between the users a and b is:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}} \tag{1}$$

- Essentially, take the ratings for items in P from a and b in two vectors. Subtract the mean of the 2 vectors from the vectors themselves to now get 2 centered vectors.
- Then once the vectors hae been centered, find the cosine of the angle between the two vectors.

- This similarity function takes values between -1 and +1. 0 similarity indicates no correlation between the two user's choices.
- Once the similarity has been found for all users, and we choose the top K most similar users to our target user, we predict the rating for item p using:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in K} sim(a,b) \times (r_{b,p} - \overline{r_b})}{\sum_{b \in K} |sim(a,b)|} \tag{2}$$

- b indicates a user who is part of the top K users chosen by the similarity function.
- The idea of this prediction function is to start with the average rating of user a and then consider for each neighbor if item p has been rated above or below the individual average rating.
- These rating signals of the nearest neighbors are consequently weighted with the similarity between users.
- Thus the ratings of more similar neighbors have a stronger influence on the derived rating predictions.
- **Variations**:
  1. **Similarity Function**: Cosine similarity, Jaccard coefficient, Dice coefficient (Latter 2 are applicable to binary data only). The similarity fn. could also take the *number* of co-rated items into account and apply some weighting.
  2. **Nearest Neighbour Selection**: Instead of selecting fixed number of neighbours or using some threshold to select, use an *adaptive* strategy, that dynamically adjusts the similarity threshold to keep the number of nearest neighbors within a predefined range
  3. **Prediction Function**: Adjusting the importance weight of controversial items with a high rating variance or amplifying the importance of very similar users

5.2.2 Item-based CF

- To predict the score given by user a for item p, the following 2 steps are done:
  - Identify the K items who exhibit the most similar rating patterns to the target item p
  - Combine the ratings given to these K items by the user a into the predicted rating for item p by user a
- Item-based methods have some practical benefits over the user-based filtering:
  - In general there are always more ratings per item than there are per user. This leads to more stable similarity models as they are now based on more common data points.
  - Pre-computing the item similarities speeds up the prediction process at runtime. It is expensive but scalable (there are way more customers than there are items, and items can be filtered out by popularity)

5.2.3 CF as a session-based approach

- The CF algorithms explained above are independent of the current context of the user's intent, recent behaviour etc.
- Hence session-based recommendation systems are formulated, which outputs recommendations that suit the context of one specific user session
- **Input**:
  - Part 1: Ordered list of user actions (each associated with an item). Data is organized at tuples of (*user, item, action*).
  - This data can be further enhanced with additional attributes like user demographics or meta-data features.

  - Part 2: Active session context, which consists of an ordered list of user actions within the current session.

- **Output**: Ordered lists of predicted next user actions, where these actions are related to items
- Goal: determine a ranked list of items that are relevant to the *actions* of the user in the *current session*. Balance long-term user preferences with session's data.
- The intended purpose of the recommended items is also important, apart from session data. (e.g.: items could be recommended as *alternatives* to the item chosen by the user, or as *accessories* to the chosen item).
- Session-based recommender systems do not have any fixed evaluation criteria which makes evaluation in an academic context difficult.

5.2.4 Nearest-neighbour approach for session-based RS

- Historical approaches to sequential pattern mining are HMMs, RNNs and explicit user feedback.
- Alternatively, as done by Amazon, focus on co=occurence patterns between user sessions.
- In the session-based nearest-neighbour algorithm:
  - Take the elements of the current user session and look for past sessions — including those by other users — who are similar to this session
  - Find which other elements appeared frequently within this set of "neighbor sessions". These represent our recommendation candidates.
- We define a sessions s as an ordered list of user actions. Given a set S of past sessions of all users, and a similarity function $sim(s_1 m s_2)$ that returns a similarity score for two sessions s_1 and s_2, we find the set of K sessions that are most similar to the target session s.
- Now we compute a relevance score for each item i for a given session s and the neighbours N of that sessions s. The relevance is given as:

  $$score_{KNN}(i, s) = \sum_{n \in N} sim(s, n) \times 1_n(i) \tag{3}$$

  where $1_n(i) = 1$ if session n contains item i and 0 otherwise
- Final recommendations are gound by sorting the items in decreasing order of relevance score.
- This approach is found to give competitive results in certain domains (music, e-commerce, workflow modelling)
- Improved results are shown to come with the use of set-based similarity functions like the binary cosine similarity or Jaccard similarity. This is due to the binary nature of data seen most often in recommender systems.
- Number of nearest neighbours N is chosen based on the application domain.

5.2.5 Pros and Cons of CF

- **Pros**:
  1. No knowledge about items is needed. This is useful for large e-commerce companies where catalogs comprise of 10^6 items or more.
  2. CF systems learn over time when additional ratings are available. They are more flexible to new data than knowledge or content based systems
- **Cons**:
  1. Require the use of a large and returning user base to identify behavioural patterns
  2. CF systems struggle with new users and new items that are added to the catalogue. This is commmonly referred to as the cold-start problem

5.2.6 Procs and Cons of Nearest neighbour approaches

- **Pros**:
  1. Easy to implement, debug and maintain.
  2. Can make use of new ratings as soon as they are available (unlike model based aproaches that need to be re-trained on new data)
  3. Explainable outputs
- **Cons**:
  1. Computational complexity is high, not suitable to provide real-time recommendations
  2. Offline processing, parallelization mitigate this

5.3 Implementation Considerations

- **Scalability**: engineering effort is needed to build scalable recommender systems
- **Freshness**: Continuous updates as and when new ratings are available. The concept of a three-tier pipeline architecture consisting of offline, nearline and online tiers has been proposed for this purpose, where the offline tier preioidically rebuilds the model, the nearline tier does near-real-time updates and the online tier performs the filtering based on user feedback and trends.
- **UI**: User-centric evaluation approaches, some consider this to be more important than the backend algorithm
- **Data Integration**: Use multiple sources using feature engineering, use implicit signals instead of the explicit ratings, and manage large amounts of data.
- **Business metrics**: RMSE is a common proxy measure, but click-through rates, choice time, user engagement and customer churn rates are used as business indicators.
- **Frameworks**: Apache PredictionIO ML server, TensorFlow, LensKit for Java, MyMediaLite for .NET, `sklearn` for Python and `rrecsys` for R, as well as plenty of public datasets.

5.4 Evaluation Considerations

- Most informative method is A/B Testing (2+ groups, each served different recommendations, compare the effects across groups)
- Large samples are needed to make the results of A/B testing statistically significant. Unexpected periodic effects lead to wrong conclusions and distortions
- Offline procedures are cheaper and more scalable to evaluate large number of recommender strategies.

5.4.1 Offline Evaluation

- For matrix-completion tasks, offline evaluation is done in the form of K-fold cross validation on a hidden test dataset.
- Typically the metrics chosen are the MAE or the RMSE. Many CF algorithms are designed to minimize the RMSE. Values cannot be compared across papers as different researchers use different data pre-processing.
- CF is also sometimes evaluated as a classification task. Hence measures such as precision (how many items recommended are relevant), recall (how many relevant items were recommended) and F-score are used.
- Precision and recall values are dependent on the items for which no ratings are available. Removing these items from consideration inflates the precision and recall scores.
- Other measures of performance are diversity (based on pairwise similarity of recommended items), novelty and serendipity (recommending items outside of user's normal preference), but these may affect accuracy hence domain-wise analysis helps.

5.4.2 Evaluation Protocol for session-based systems

- The dataset of user actions logs is split into train and test sets, and the task is to predict the hidden actions.
- In Jannach 2015, the most recent 20% of sessions are considered as test data. The sessions in the test set are then evaluated individually according to the given prediction task, which could be, e.g., to predict the purchase event in the
- Two parameters are set for this:
  - The v parameter indicates how many items of the current session to be revealed to the algorithm when making a prediction (measures how quickly algorithm adapts to current trends)
  - The p parameter indicates how many sessions that precede the current user session should be revealed to the algorithm (test how well the algorithm leverages info from preceding sessions)
- Accuracy measures include precision, recall, MRR and other measures that measure different dimensions. e.g. in ACM's REcSys challenge, predict whether or not a user will make a purchase in the current session and which item the user will purchase, hence the evaluation metric considers both aspects.

5.4.3 User Studies

- Accuracy of recommendations is not the only component that influences the performance of a system.
- User studies play an important role in recommender system evaluation.
- Real studies have shown contradictory results between precision measures and user studies. For these reasons, industrial leaders often employ a development pipeline that involves three steps:
  1. Traditional offline experiments to identify failing approaches from the very beginning
  2. User studies to identify and better understand promising candidates
  3. A/B testing in the field as the third step in the assessment cycle.
- Lack of a representative sample of users, and differing intents from survey and real-world situations are the challenges faced when conducting user surveys.

5.5 Datasets

- MovieLens and Netflix challenge datasets
- The increasing availability of data for different domains, different product categories and different sources will further intensify research on cross-domain recommendations and combining multiple user feedback sources.