CS5228 – Tutorial 8

Recommender Systems

1. **Recommendation Systems – Basic Challenges.** In the lecture, we came across 2 basic problems when building recommendation systems or engines: *popularity bias* and *cold-start problem*. Briefly describe both problems in your own words.

   Solution:

   - Popularity bias: only a small subset of items gets regularly recommended ("few get richer", "rich get richer")
   - Particularly when a new user joins the platform, no or not much information about that user is available to provide proper personalized information. In principle, the same is true for a new item (as long as it hasn't received a sufficient number of ratings) but at least some kind of item-item similarity can be used to make (halfway decent) recommendations.

2. **Explicit vs implicit feedback.** Amazon's 5-star rating scheme or Reddit's upvote/downvote scheme are considered explicit feedback. In contrast, implicit feedback may refer to users' playlist/purchase/clickthrough/etc. history. What are the main limitations of implicit feedback compared to explicit feedback?

   Solution:

   - Implicit feedback is not necessarily a clear indicator for a user's preference. For example, just clicking on some content does not imply that the user did indeed like the content.
   - Implicit feedback lacks a clear notion of negative feedback; the absence of an interaction does not imply lack of interest, liking, etc.
   - Implicit feedback is often likely to contain more noise. For example, a user might accidentally click on a well disguised ad.

3. **Normalization.** Why do we typically normalize the ratings by mean-centering them, i.e., by subtracting the mean, either for a user or an item?

Solution:

- Most numeric rating schemes use only positive values where low positive values reflect dislike; for most computations, representing dislike by negative values yield more expressive results.
- Normalizing the ratings for each user allows us to bring more "generous" users (often give high ratings) and more "grumpy" users (often give low ratings) to the same scale.
- Algorithms that use all entries of rating matrix R incl. missing values represented by 0 – e.g., when calculating vector similarities – assume a meaningful interpretation of 0. Without normalization 0 would represent a stronger dislike than the lowest value of 1 star. Normalization allows that 0 represents neutral feedback – not good or bad – which is more reasonable.

4. **Content-Based Recommender Systems**. Content-based recommender systems require to represent items as some form of features vector (item profiles) to calculate distances/similarities between them.

   (a) For the following 5 types of items, what are arguably useful information to create a item profile to allow for meaningful recommendations

   - Electronic devices (e.g., phone, cameras, laptops)
   - News articles
   - Hotel (rooms)
   - Books
   - Property/Housing

   Solution:

   - Electronic devices (e.g., phone, cameras, laptops)
     - basically all technically features
   - News articles
     - Good: source (newspaper and/or author), text features (but not trivial to extract)
     - Questionable: article length, number of images
   - Hotel (rooms)
     - basic information such as size, amenities, category (star rating), location
     - The problem is that this information typically provides a very incomplete picture.

- Books
  - author, genre, publication year, (length?)
  - Again, this information will often be not sufficient to appropriately describe the quality of a book.
- Property/Housing
  - area size, floor height, age, location

(b) Based on your answers in (a), how would you classify items into 2 basic categories when it comes to building a content-based recommendation system? This is a very open question, and there are probably many good answers.

Solution: There is arguably a big difference between items that can be (more or less objectively) evaluated (e.g., electronic devices, cars). For such types of items it's generally easier to identify useful features to build item profiles from. In contrast, opinions about books, movies, hotels, restaurants, etc. can be very subjective, making it difficult to create good item profiles.

5. **KNN-Based Recommender System**. We saw that many data mining algorithms can be used to make recommendations, such as Clustering, Association Rule Mining, or Classification/Regression models. Let's consider the K-Nearest Neighbor Algorithm here.

(a) Sketch a KNN algorithm to recommend items based on user similarity derived using only the rating matrix R!

Solution:

- Optional: Extract user vectors from data matrix and normalize values by subtracting the mean of the ratings of each vector.
- Calculate distances between user vectors using a suitable distance metric (e.g., Cosine Similarity, Pearson Correlation Coefficient)
- For each user u, find the K-nearest neighbors N (i.e., the most similar users) based on the pairwise similarities.
- Aggregate the interactions of the N users (weighted by their similarity scores) to calculate the predicted ratings for u – this should only be done for items that have been rated by a sufficient number of users in N.
- Rank the items that the u has not yet rated by the predicted ratings to derive meaningful recommendations (e.g., top-ranked items but with some diversity).

(b) How does the choice of K in KNN is likely to affect the quality of recommendations?

Solution:

- K too small
  - recommendations rely on a very limited number of neighbors, which might lead to biased or highly personalized recommendations; this may cause overfitting to individual tastes
  - recommendations that are likely to be very narrow or idiosyncratic
  - recommendations may be highly accurate for some users but less diverse or useful for others, as the model may miss out on exploring broader patterns across multiple users
- K too large:
  - recommendations start to rely on too many neighbors that may not be very similar to u
  - too many neighbors are likely to "dilute" the influence of the most similar users and increases the risk of introducing noise from less relevant users, leading to less personalized recommendations
  - recommendations become more generalized and less tailored to individual preferences; while the diversity of recommendations may improve, the overall accuracy and relevance may decrease.