# Recommender Systems

Nivio Ziviani

Departamento de Ciência da Computação da UFMG

Abril de 2014

# Introduction

**Chapter 1 of Recommender Systems Handbook**
**Ricci, Rokach, Shapira and Kantor (editors), 2011.**

# Introduction

- *Recommender Systems* suggests items to a user.
- The sugestions relate to decision-making processes, such as what products to buy, what music to listen, or what news online to read.
- Item: general term used to denote what the system recommend to users.
- Personalized recommendations are offered as ranked lists of items.
- RSs predict what the most suitable products or services are, based on the user's preferences and constraints.
- RSs collect from users their preferences, e.g., as ratings for products, or are inferred by interpreting user actions.
- Amazon.com: personalize the online store for each customer.

# RSs Motivations

- *individuals often rely on recommendations provided by others in making routine, daily decisions*.
- It is common to rely on oppinion of others when selecting a book to read.
- Employers count on recommendation letters in their recruiting decisions.
- When selecting a movie, individuals tend rely on movie reviews.
- In seeking to mimic this behavior, the first RSs recommend items that users with similar tastes had liked.
- This approach is termed *collaborative-filtering*.

# Introduction

- RSs emerged as a research area in the mid-1990s, but in the recent years the interest in RSs has increased:
  - RSs play an important role in Amazon.com, YouTube, Netflix, Yahoo, TripAdvisor, Last.fm, IMDb, etc..
  - There are dedicated conferences and workshops: ACM International Conference on Recommender Systems (RecSys), stablished in 2007, is now the premier conference. Others: ACM SIGIR and ACM SIGMOD.

# Recommender Systems Function

- *Increase the number of items sold*: This is probably the most important function for a commercial RS.
  - To be able to sell an additional set of items compared to those usually sold without any kind of recommendation.
  - Primary goal is to increase the conversion rate.
- *Sell more diverse items*: Select items that might be hard to find without a precise recommendation. E.g., Netflix suggests or advertises unpopular movies to the right users.
- *Increase user satisfaction*: Accurate recommendations will increase subjective evaluation of the system by users.
- *Increase user fidelity*: Recognize old customers and leverage the information acquired in previous interactions.
- *Better understand what the user wants*: Description of users preferences, collected explicitly or predicted by the system.

# Eleven Popular Tasks

**Popular tasks by Herlocker, Konstan and Riedl, "Explaining Collaborative Filtering Recommendations", 2000**

- *Find some good items*: Recommend items as a ranked list with predictions (e.g., one to five star scale).
- *Find all good items*: Recommend all items that can satisfy some user needs.
- *Annotation in context*: Given a list of items, emphasize some of them depending on long-term preferences.
- *Recommend a sequence of items*: e.g., TV series; a book on RSs after having recommended a book on data mining.
- *Recommend a bundle*: e.g., a travel plan with various attractions, destinations, and accommodation services that are located in a delimited area.

# Recommender Systems Function

- *Just browsing*: help the user to browse the items that are more likely to fall within the scope of users interests.
- *Find credible recommender*: Offer specific functions to let the users test its behavior in addition to those just required for obtaining recommendations.
- *Improve the profile*: Provide what the user likes and dislikes (necessary to provide personalized recommendations).
- *Express itself*: Allow users to contribute with their ratings and express their opinions and beliefs.
- *Help others*: Evaluation of items (ratings) by users (they believe that the community benefits from their contribution).
- *Influence others*: There are users whose main goal is to influence other users into purchasing particular products (carefull with malicious users).

# Data and Knowledge Sources

- Data used by RSs refers to three kinds of objects:
  - Items
  - Users
  - Transactions

# Data and Knowledge Source: Items

- Are the objects that are recommended.
- Are characterized by their complexity, value or utility.
- The value of an item is *positive* if it is useful, or *negative* if the user made a wrong decision when selecting it.
- Acquiring an item incurs in the cognitive cost of searching for the item and the real monetary cost paid for the item.
- Items with low complexity: news, Web pages, books, CDs, movies.
- Items with larger complexity: digital cameras, mobile phones, PCs, etc.
- most complex items: insurance policies, financial investments, travels, jobs, etc.

# Data and Knowledge Source: Items

- RSs can use a range of properties and features of items.
  - E.g., in a movie recommender system, the genre (comedy, thriller, etc.), the director, and actors can be used to describe a movie and to learn how the utility of an item depends on its features.
- Items can be represented using various information and representation approaches, e.g., as a single id code, or in a richer form, as a set of attributes.

# Data and Knowledge Source: Users

- May have very diverse goals and characteristics.
- RSs exploit information about users to personalize recommendations.
- This information can be structured in various ways:
  - In collaborative filtering, users are modeled as a simple list containing the ratings provided by the user for some items.
  - In a demographic RS, attributes such as age, gender, profession, and education, are used.

# Data and Knowledge Source: Users

- User data constitute the user model and encodes her preferences and needs.
- A RS generates recommendations by building and exploiting user models.
- The user model will always play a central role.
  - E.g., in a collaborative filtering approach, the user is either profiled directly by its ratings to items or, using these ratings, the system derives a vector of factor values, where users differ in how each factor weights in their model.

# Data and Knowledge Source: Transactions

- Recorded interactions between users and the RS.
- Log-like data that store information generated during the human-computer interaction, which are useful for the recommendation generation algorithm.
  - E.g., a transaction log may contain a reference to the item selected by the user and a description of the context for that particular recommendation.
  - That transaction may also include an explicit feedback the user has provided, such as the rating for the selected item.
- In fact, ratings are the most popular form of transaction data that a RS collects.

# Data and Knowledge Source: Transactions

Forms of ratings:

- Numerical ratings such as the 1-5 stars provided in the Amazon book recommender.
- Ordinal ratings; strongly agree, agree, neutral, disagree, strongly disagree, the user is asked to select the term that best indicates her opinion regarding an item.
- Binary ratings: an item is good or bad.
- Unary ratings: a user has observed or purchased an item, or rated the item positively.
- Tags associated by the user with items: e.g., in Movielens tags represent how MovieLens users feel about a movie – too long, or acting.

# Data and Knowledge Source: Transactions

Forms of ratings:

- Implicit ratings: infer users opinion based on their actions.
- If a user enters the keyword *Yoga* at Amazon.com she will be provided with a long list of books.
- The user may click on a certain book to receive additional information and the system may infer that the user is interested in that book.

# Recommendation Techniques

- To identify useful items for the user, a RS must predict that an item is worth recommending.
- The system must be able to predict the utility of an item or compare the utility of some items and decide what to recommend.
- The prediction step may not be explicit: e.g., consider a simple, nonpersonalized, algorithm that recommends just the most popular songs.
- The rationale is that in absence of more precise information, a popular song will be probably liked by a generic user, at least more than another randomly selected song.
- Hence the utility of these popular songs is predicted to be reasonably high for this generic user.

# Recommendation Techniques

**Taxonomy by Burke, "Hybrid Web Recommender Systems", In: The Adaptive Web, 2007**

*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. Examples:
  - 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.
  - Pandora uses the properties of a song or artist to seed a "station" that plays music with similar properties. User feedback is used to refine the station's results, deemphasizing attributes when a user "dislikes" a song and emphasizing other attributes when a user "loves" a song.

# Recommendation Techniques

*Collaborative filtering*:

- Recommends to the active user the items that other users with similar tastes liked in the past.
- The similarity in taste of two users is calculated based on the similarity in the rating history of the users.
- This is the reason why some authors refer to collaborative filtering as ”people-to-people correlation”.
- For example, Last.fm creates a ”station” of recommended songs by observing what bands that the user has listened to on a regular basis and comparing those against the listening behavior of other users.
- Last.fm will play tracks that do not appear in the user’s library, but are often played by other users with similar interests.

# Recommendation Techniques

*Demographic*:

- Recommends items based on the demographic profile of the user. The assumption is that different recommendations should be generated for different demographic niches.
- For example, users are dispatched to particular Web sites based on their language or country.
- Or suggestions may be customized according to the age of the user.

# Recommendation Techniques

*Knowledge-based*:

- 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.
- A similarity function estimates how much the user needs (problem description) match the recommendations (solutions of the problem).
- Here the similarity score can be directly interpreted as the utility of the recommendation for the user.
- Knowledge-based systems tend to work better than others at the beginning of their deployment but if they are not equipped with learning components they may be surpassed by other shallow methods that can exploit the logs of the human/computer interaction (as in CF).

# Recommendation Techniques

*Community-based*:

- Recommends items based on the preferences of the users friends.
- This technique follows the epigram “Tell me who your friends are, and I will tell you who you are”.
- Evidence suggests that people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals. This observation, combined with the growing popularity of open social networks, is generating a rising interest in social recommender systems.
- The recommendation is based on ratings that were provided by the user’s friends.

# Recommendation Techniques

*Hybrid*:

- 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.

# Algorithms (Wikipedia)

Hundreds of algorithms have been used in the design of recommender systems. Some examples:

*K-Nearest Neighbor*:

- One of the most commonly used algorithms in RSs.
- The k-NN algorithm classifies objects based on the properties of its closest neighbors in the feature space.
- In k-NN, an object is classified through a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small).
- If k = 1, then the object is simply assigned to the class of its nearest neighbor.

# Algorithms (Wikipedia)

*Pearson Correlation*:

- The Pearson Correlation is a measure of the correlation (linear dependence) between two variables X and Y, giving a value between +1 and -1 inclusive.
- In a social network, a particular user's neighborhood with similar taste or interest can be found by calculating the Pearson correlation coefficient.
- By collecting the preference data of top-N nearest neighbors of a particular user (weighted by similarity), the user's preference can be predicted.

# Algorithms (Wikipedia)

*Rocchio Relevance Filtering*:

- Method of relevance feedback dating back to the 1970s.
- Rocchio makes use of the VSM and is based on the assumption that most users have a general conception of which items should be denoted as relevant or non-relevant.
- User feedback is used to refine a search query by emphasizing or deemphasizing certain terms (similar to how Pandora refines its user recommendations).
- Through feedback, the user's search query is revised to include relevant and non-relevant terms as a means of increasing the search engine's recall, and possibly the precision as well.
- The number of relevant and non-relevant terms allowed to enter a query is dictated by a series of weights in the central equation.

# Cold Start Problem

- The system cannot draw any inferences for users or items it has not yet gathered sufficient information.
- In the CF approach, the RS would fail to consider items which no-one in the community has rated previously.
- The cold start problem is often reduced by adopting a hybrid approach between content-based matching and collaborative filtering.
  - New items would be assigned a rating automatically, based on the ratings assigned by the community to other similar items (according to the items' content-based characteristics).
- Thales Filizola Costa, Taxonomy-Driven Content-Driven Recommendation for New Items, Dissertação de Mestrado, DCC/UFMG, Fevereiro de 2014.

# Sparsity Problem

- The number of items sold on major e-commerce sites is extremely large.
- The most active users will only have rated a small subset of the overall database.
- Thus, even the most popular items have very few ratings.
- Bessa, A., Laender, A.H.F., Veloso, A. and Ziviani, N. “Alleviating the Sparsity Problem in Recommender Systems by Exploring Underlying User Communities”. *6th Alberto Mendelzon International Workshop on Foundations of Data Management*, Ouro Preto, Brazil, June 28-30, 2012.

# Metrics

Herlocker et al., Evaluating collaborative filtering recommender systems, TOIS, v22, jan 2004

- *Precision*: $P = \frac{N_{rs}}{Ns}$
  Ratio of relevant items to number of items selected. Represents the probability that a selected item is relevant.
- *Recall*: $R = \frac{N_{rs}}{N_r}$
  Ratio of relevant items selected to total number of relevant items available. Represents the probability that a relevant item will be selected.
- $F1 = 2 \times \frac{P \times R}{P+R}$
  Combines precision and recall into a single number.

# Metrics

- *Accuracy*: $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - r_i)}{n}}$
  Root-mean-square error measures the differences between values predicted by a model or an estimator and the values actually observed, where $p_i$ is the $ith$-prediction and $r_i$ is the actual value of the $ith$-prediction.
- *Diversity*
- *Novelty*
- Ribeiro, M.T., Lacerda, A., Veloso, A. and Ziviani, N. "Pareto-Efficient Hybridization for Multi-Objective Recommender Systems", *6th ACM Conference on Recommender Systems*, Dublin, Ireland, 9-13th Sept, 2012.

# Amazon.com Recommendations

**G. Linden, B. Smith and J. York, Item-to-Item Collaborative Filtering, IEEE Internet Computing, 2003**

- Use recommendation algorithms to personalize the online store for each customer.
- It is based on customer interests, showing programming titles to a software engineer and baby toys to a new mother.

# E-Commerce Environment

E-commerce recommendation algorithms often operate in a challenging environment. For example:

- A large retailer might have huge amounts of data, tens of millions of customers and millions of distinct catalog items.
- Results returned in realtime, in less than 300 ms, while still producing high-quality recommendations.
- New customers typically have extremely limited information, based on only a few purchases or product ratings.
- Customer data is volatile: each interaction provides valuable customer data, and the algorithm must respond immediately to new information.

# Common Approaches to Solving the Recommendation Problem

**Approach 1: Find similar customers**

- Find a set of customers whose purchased and rated items overlap the purchased and rated items of a user;
- Aggregates items from these similar customers;
- Eliminates items the user has already purchased or rated;
- Recommends the remaining items to the user.

Examples: *collaborative filtering* and *cluster models*

# Common Approaches to Solving the Recommendation Problem

**Approach 2: Find similar items**

- For each of the purchased and rated items of a user, find similar items.
- Aggregates the similar items and recommends them.

Examples: *search-based methods* and Amazon's *item-to-item collaborative filtering*.

# Traditional Collaborative Filtering

- Represents a customer as an $N$-dimensional vector of items, where $N$ is the number of distinct catalog items.
- Components of the vector are positive for purchased or positively rated items and negative otherwise.
- This vector is extremely sparse for most users.
- It generates recommendations based on a few customers who are most similar to the user.
- It can measure the similarity of customers A and B as the *cosine* of the angle between the two vectors.
- Rank each item according to how many similar customers purchased it.

# Traditional Collaborative Filtering

- It is $O(MN)$ in the worst case: it examines $M$ customers and up to $N$ items for each customer.
- As the average customer vector is sparse, it tends to $O(M + N)$.
- Scanning every customer is approximately $O(M)$ (not $O(MN)$ because customer vectors contain a small number of items.
- For large data sets – such as 10 million or more customers and 1 million or more catalog items – the algorithm encounters severe performance and scaling issues.

# Search-Based Methods

- Search- or content-based methods treat the recommendations problem as a search for related items.
- Given the user's purchased (or rated items), the algorithm constructs a search query to find other popular items by the same author, artist, or director, or with similar keywords or subjects.
- E.g., If a customer buys the Godfather DVD Collection, the system might recommend other crime drama titles, other titles starring Marlon Brando, or other movies directed by Francis Ford Coppola.

# Search-Based Methods

- If the user has few purchases or ratings, algorithms scale and perform well.
- For users with thousands of purchases, however, it is impractical to base a query on all the items.
- The algorithm must use a subset or summary of the data, reducing quality.
- In all cases, recommendation quality is relatively poor. The recommendations are often either too general (such as best-selling drama DVD titles) or too narrow (such as all books by the same author).
- Recommendations should help a customer find and discover new, relevant, and interesting items. Popular items by the same author or in the same subject category fail to achieve this goal.

# Amazon Algorithm: Item-to-Item Collaborative Filtering

- Online computation scales independently of the number of customers and number of items in the product catalog.
- Produces recommendations in realtime.
- Scales to massive data sets.
- Generates high quality recommendations.

# Item-to-Item Collaborative Filtering

## Amazon.com Algorithm

- It uses recommendations as a targeted marketing tool in email campaigns and on its Web sites' pages.
- Users can filter their recommendations by product line and subject area, rate the recommended products, rate their previous purchases, and see why items are recommended.
- Offers customers product suggestions based on the items in their shopping cart.
- The feature is similar to the impulse items in a supermarket checkout line, but our impulse items are targeted to each customer.

# Item-to-Item Collaborative Filtering

## How it Works

- Rather than matching the user to similar customers, it matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list.
- To determine the most-similar match for a given item, the algorithm builds a similar-items table by finding items that customers tend to purchase together.

# Item-to-Item Collaborative Filtering

## Similar-Items Table

Calculates the similarity between a product and all related products:

```
For each item I1 in product catalog
  For each customer C who purchased I1
    For each item I2 purchased by customer C
      Record that a customer purchased I1 and I2
  For each item I2
    Compute the similarity between I1 and I2
```

Similarity between two items given by the cosine measure:

- Each vector corresponds to an item;
- The dimension $M$ of the vector corresponds to customers who have purchased that item.

# Item-to-Item Collaborative Filtering

## Similar-Items Table

**Complexity of the off-line computation:**

- $O(N^2M)$ in the worst case.
- $O(NM)$ in practice, as most customers have very few purchases.
- Sampling customers who purchase best-selling titles reduces runtime even further, with little reduction in quality.

# Item-to-Item Collaborative Filtering

## The Algorithm

Given a similar-items table:

- The algorithm finds items similar to each of the purchases and ratings of a user;
- Aggregates those items;
- Recommends the most popular or correlated items.

Complexity: This computation depends only on the number of items the user purchased or rated.

# Item-to-Item Collaborative Filtering

## Scalability

- The key to scalability and performance is that it creates the expensive similar-items table off-line.
- Looking up similar items for the purchases and ratings of a user scales independently of the catalog size or the total number of customers.
- It depends only on how many titles the user has purchased or rated.
- Because the algorithm recommends highly correlated similar items, recommendation quality is excellent.
- Unlike traditional collaborative filtering, the algorithm also performs well with limited user data, producing high-quality recommendations based on as few as two or three items.

# Item-to-Item Collaborative Filtering

## Comparison to Tradition Collaborative Filtering

- Traditional collaborative filtering does little or no off-line computation.
- Its online computation scales with the number of customers and catalog items.
- It is impractical on large data sets, unless it uses dimensionality reduction, sampling, or partitioning – all of which reduce recommendation quality.

# Item-to-Item Collaborative Filtering

## Comparison to Search-Based Models

- Search-based models build keyword, category, and author indexes off-line, but fail to provide recommendations with interesting, targeted titles.
- They also scale poorly for customers with numerous purchases and ratings.