Main RS Methods:

* Collaborative filtering methods.
* Content-based methods.
* Knowledge-based methods.

Chapters classification:

* Algorithms and evaluations: discuss the techniques behind each method and ways to assess the performance of a system.
* Recommendations in specific domains and contexts: defines the context of a system (ie. location for restaurants) using spacial, temporal or social data. Details issues of using social information to increase trustworthiness of RS.
* Advanced topics and applications:  introduce principles of development while discussing robustness of systems. Application of various methods.

Intro:

* Easy to provide feedback on the web (Netflix 5 star rating, Amazon browsing detection).
* Recommending user’s futur choices from previous interactions is a good method for **content-based systems** (except in **knowledge-based RS** which uses requirements specified by users combined by knowledge of the domain to provide recommendations).
* Basic principle of recommendations is the dependencies between user and item centric activity (ie. a user’s tastes).
* Can group users by cohorts of similar users to make recommendations (larger sets of data and users increase accuracy). This is the basis of **neighborhood models** or more generally **collaborative filtering**.

Goals of RS:

* Problem models:
  + **Prediction version** (or matrix completion problem): predict rating of a user/item combination using training data.
  + **Ranking version** (or **top-k recommendation problem**): rank best users (for an item) or best items (for a user, more common) in order to make recommendations of top ones.
* Primary goal is to increase sales (used by merchants to make customer recommendations). Divide in following sections:
  + Relevance: recommend relevant items to user increases chances of purchase (this is the primary goal).
  + Novelty: Recommendations that a user has not seen (avoid repeated recommendations of popular items).
  + Serendipity: item somewhat unexpected (element of ,lucky discovery) to surprise user. Has positive long term (user finds new trend) effects which outweigh negative short term effects (irrelevant user recommendations).
  + Diversity: include diverse top-k items (or user might not like all of them) for greater chance that user likes at least one.
* Other soft goals to increase overall user satisfaction (so they are more likely to come back). Can also add explanation of why item (or connections for Facebook) was recommended to the user.
* Example of different RS:
  + GroupLens: research prototype for news recommendations by collecting ratings from readers to make estimations.
  + Amazon: uses rating based recommendation. Contains explanations of the recommendation. Browsing data collected from logged-in users.
  + Netflix: uses 5 point rating and user’s history of actions to provide (explained: telling user ‘why’ can improve retention and loyalty) recommendations. Holds prize contests evaluating efficiency of recommender algorithms.
  + Google News: uses user’s click history (implicit rating of news =/= not explicitly defined by user).
  + Facebook: recommending friends has different goals (not the same as recommending products), that is to improve user experience (not product sales) which increases network’s growth (and ad revenue). Problem known as **link prediction**, uses structural relationships (different algorithm).
  + LinkedIn: referred to as **reciprocal recommender** where job seekers and employers are recommended to one another.

Basic models of RS:

* Two kinds of data: user-item interaction (ratings or buying behaviors) and attribute information (textual profiles and relevant keywords).
* **Collaborative filtering** methods use the former. Challenge is that rating matrices are sparse (a user will not rate all movies: specified or unspecified). Use similarity in users to determine unspecified ratings (inter-item/user correlation). Better when we have a lot of starting data.
  + **Memory-based methods**: (aka **neighborhood-based collaborative filtering algorithm**) where ratings of user/item combinations predicted based on neighborhood, which is defined in two ways:
    - **User-based collaborative filtering**: ratings provided by like-minded users for a target user in order to find value of ratings of target user (based on average rating given by like-minded users). Find similar users by comparing rows of rating matrice.
    - **Item-based collaborative filtering**: to find target user’s rating of an item, find similar items (that have been rated by the target user). Find similar items by comparing their ratings (columns).

These methods are simple to implement and easy to explain but perform poorly with sparse rating matrices: can lack full coverage of rating predictions, but this may not be an issue if only top-k items are required.

* + **Model-based methods**: Use machine learning and data mining to build **predictive models**. Use optimization framework if model is parameterized. Examples include **decision trees**, **rule-based models**, **Bayesian methods** and **latent factor models**. They have high level of coverage (even with sparse rating data).
  + Types of ratings: usually interval-based designs. It is unbalanced if there are more favorable settings than unfavorable ones. It is a forced choice system if there is no neutral rating. Ordinal ratings use categorical values rather than numbers. Binary ratings have two settings (like or dislike). Unary ratings more common in implicit feedback data sets.
* **Content-based recommender** methods use the latter (can also use rating matrices). Description/attributes of items used for recommendations (ie. Terminator -> recommend action movies). Create a user-specific classification to predict how user will rate an unknown item.
  + *Advantages*: Good when item does not have sufficient amount of ratings (since we can leverage other items with similar attributes).
  + *Disadvantages*: often provide obvious recommendations (and little diversity if user has never ‘liked’ an item with a given set of attributes). Not effective at giving recommendations to new users (no rating history to use).
* **Knowledge-based systems**  use explicitly specified user requirements. Better for a new user (“cold start”) since they can specify attributes they like. Used where items not purchased often (automobile, houses, tourism, luxury…) which gives little rating data. This requires good domain knowledge (ie. knowing attributes of items) to make adequate recommendations (based on specified preferences and similar users).
  + **Constraint-based systems:** user specified constraints (price range, boundaries, location…). Often link user attributes to item attributes (ie. young people buy cheaper houses).
  + **Case-based systems:** Specific cases given by user as target or anchor point. System uses metrics to retrieve similar items. Returned results can be used recursively as new target.

Both systems need UI (user may adapt requirements) for good interaction. Most common models are **conversational systems** (iterative feedback loop determines user preference), **search-based systems** (preset of questions or search interfaces for user to specify constraints) or **navigation-based recommendation** (user specifies change requests to be applied on item that is currently recommended, also known as **critiquing RS**).

* + **Utility-based systems:** define utility function to compute probability of user liking an item (can be challenging) by giving it a utility value.
* **Demographic systems** take user’s demographic information to make recommendations (can be combined with additional context). Not best results if used by itself but good as component of hybrid or ensemble systems.
* **Hybrid & ensemble-based systems** combine different aspects of the previous systems (performing more robustly in wider settings). Ensemble-based systems can combine the power of multiple data sources and improve the effectiveness of a class of systems by combining models.

**Evaluation of RS**: The problem can be viewed as generalization of the classification problem (so we can use the same evaluation models). We consider rating, prediction and ranking.

Domain Specific challenges in RS:

The context plays a critical role in recommendations, so **contextual RS** were introduced (using temporal, location or social data).

* **Context-based systems** take contextual information like time, location or social data into account (ie. clothes) which improves the effectiveness of recommendations.
* **Time-sensitive systems** are very useful (movies, clothes…). The rating of an item may evolve with time (user interests, fashion…) or it may depend on the specific time (day, month, season). Using temporal context may accentuate the sparsity problem of the data.
* **Location-based systems** is used (ie. for restaurants) with two common types of spatial locality: user specific locality (geographical location of user) and item specific locality (location of the item impacts its relevance).
* **Social systems** are based on network structures, social cues and tags. Either purely based on structural aspects and used to suggest other nodes, or based on cues to recommend other products.
  + **Structural recommendations of nodes and links** used in many node-based systems (ie. graphs - web search use page rank algorithm) to recommend other nodes of interest (using known nodes of interest if available: collective classification). Closely related to link recommendation or link prediction problems (not just for social networks).
  + **Product & content recommendations with social influence** use network connections and social cues (aka viral marketing). Need to determine influential and topically relevant entities in the network (influence analysis). For example finding influential twitter accounts.
  + **Trustworthy RS** users can express their trust and distrust to one another (robust recommendations).
  + **Leveraging social tagging feedback for recommendations** as it is a form of feedback between users (mostly on sharing platforms). User can tag content, which provides information about content itself and tastes of the user.

Advanced topics and applications:

* **The cold-start problem in RS**  small number of initial ratings make it difficult to start for most methods. Best are content-based and knowledge-based methods (not always available). Specific methods have been designed for this problem.
* **Attack resistant RS** sellers have high interest to manipulate recommenders systems (ie. inflated ratings or malicious reviews to rivals). There are methods to make recommendations more robust against this kind of attacks.
* **Group RS** are tailored to recommend product/activity to a group rather than a user by aggregating the preferences of individuals (must be better than simple sum of parts). Averaging strategies do not work well when group is heterogeneous (diverse tastes). Users often impact each other tastes (emotional contagion, social conformity...).
* **Multi-criteria RS** can take multiple criterias from a single user (ie. price and location). Similar overall rating (if specified) between two users for a product should not be interpreted as same tastes (components may be very different).
* **Active Learning in RS** sparsity of data is one of the major challenges. Option is to encourage user to provide ratings on selected items (that the user has experience with). Selection is important (useless to ask a user to rate same type of item repetitively).
* **Privacy in RS** is key as feedback may provide a lot of sensitive information about a user (depending on product). However, it is very important for algorithms to have access to real data for progress (while respecting privacy and following regulations).
* **Application Domains** are varied (retail, music, content, web search, querying, advertisements…). The domain may require different recommendation methods. RS assume strong user-identification to identify long term interests.

Summary:

Most methods have advantages and disadvantages, and can be more or less effective depending on the situation. The recommendation problem is a rich one, with many variations. Hybrid cases may be developed to exploit certain trade-offs.