Recommender Systems - Introduction

Machine Learning, A.Y. 2022/23, Padova

Fabio Aiolli

December 19th, 2022

The paradox of choice

Is it always good to have many alternatives?

24 flavors of jam

- **60%** of the customers stopped at the booth;
- On average, 2 tastes;
- Only the 3% of the customers purchased.

6 flavors of jam

- **40%** of the customers stopped at the booth;
- On average, 2 tastes;
- 30% of the customers purchased.

Recommender Systems everywhere

RecSys Boom: The Netflix Prize

Between 2006 and 2009, Netflix sponsored a famous competition offering 1M$ to the team that, on the basis of a dataset with

- $\sim$ 480K users
- $\sim$ 18K movies (a.k.a. items)
- $>$ 100M ratings,

was able to **improve by at least 10%** the performance of the Netflix algorithm in predicting the missing ratings.

- R.M. Bell, Y. Koren, C. Volinsky (2007).
  "The BellKor solution to the Netflix Prize"
- -, and J. Bennet (2009). "The Million Dollar Programming Prize"

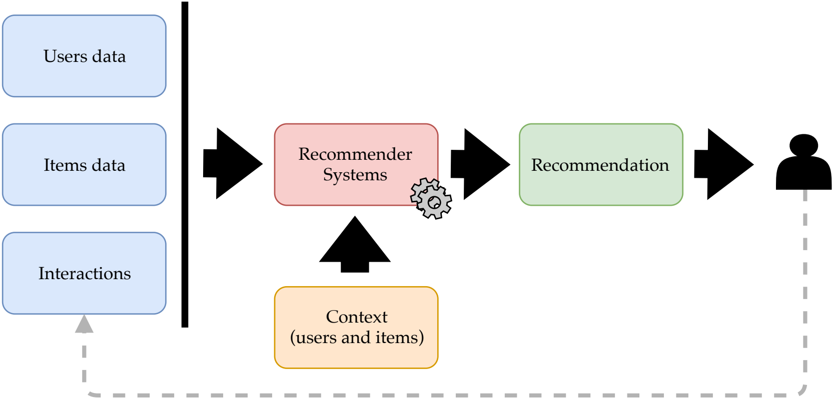
Trivia: the winning team used an **ensemble** composed by more than 100 predictors.

What is a recommender systems?

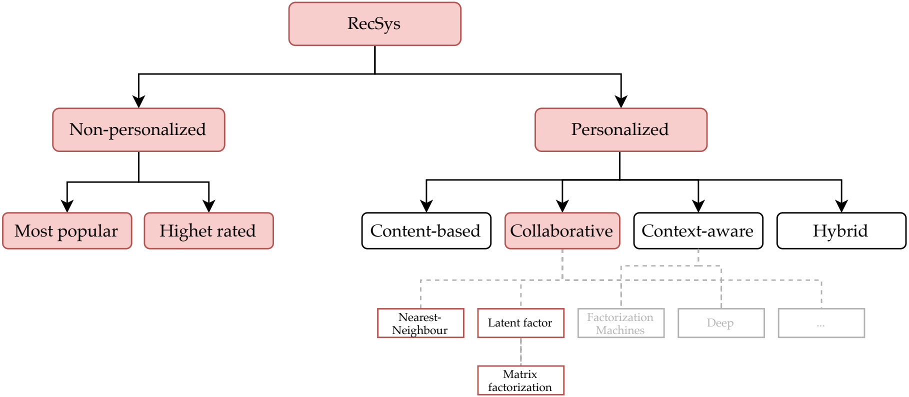
Wikipedia *A recommender system is a subclass of information filtering system that seeks to predict the preference a user would give to an item.*

Handbook *Recommender Systems (RSs) are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user.*

General RecSys scenario



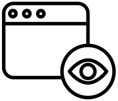
Recommender systems' taxonomy

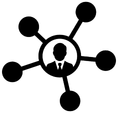


Types of feedback/interaction/rating

- **Explicit feedback**
  "Reliable" but hard to collect since they require an effort by the user.

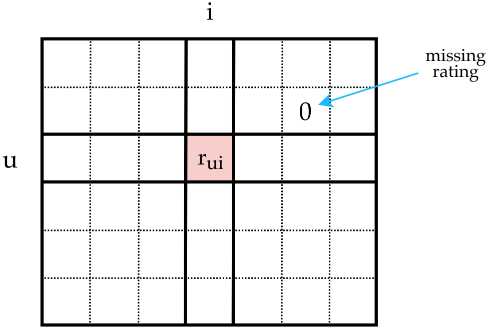
- **Implicit feedback**
  Easier to collect, but noisy.



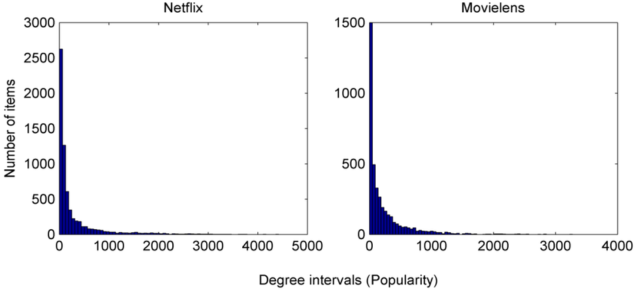


Rating matrix

- Typically, users are arranged on the rows, and items on the columns;
- r_{ui} is the rating given by user u to item i, e.g.,
  - Explicit: $r_{ui} \in \{0, 1, \ldots, 5\}$;
  - Implicit: $r_{ui} \in \{0, 1\}$.
- 0 means no information.



- Typical rating matrix density $< 0.01\%$, e.g.,
  - Netflix rating matrix density $\approx 0.002\%$;
  - MovieLens rating matrix density $\approx 0.005\%$.
- Both users activity and items popularity usually follow a long tail distribution.



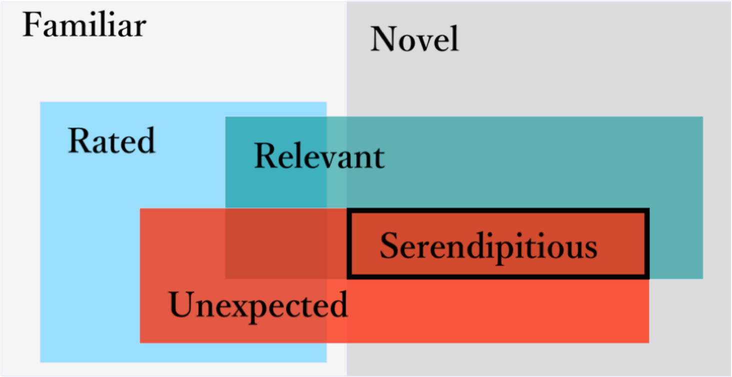
Recommendation tasks

- Rating prediction (explicit feedback) in which the RS aims at predicting the missing ratings in the rating matrix;
- TOP-N item recommendation (implicit feedback) in which the RS aims at predicting the N (previously unseen) items the user will like the most. Given a user, a recommender needs to associate a relevance score to the items. This scoring will be used to rank items and accordingly make the recommendation. Specifically, the N highest scored items are recommended to the users.

Quality indicators

- Relevance: recommend items that users like;
- Coverage: ability to recommend most of the items in a catalogue;
- Novelty: recommend items unknown to the user;
- Diversity: diversify the recommended items;
- Serendipity: ability of surprising the user, i.e., the ability to recommend items that users would have never been able to discover by themselves.

Items from the user's perspective: a visual intuition

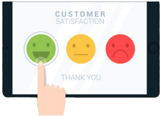


Evaluating a recommender systems

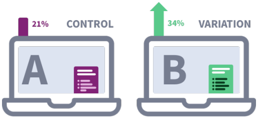
- **Offline**
  - For years the only used evaluation technique;
  - Do not require the user “live”;
  - Based on benchmark datasets (training set + test set);
  - It cannot be used to evaluate the experience as a whole.
- **Online**
  - Users are directly involved in the evaluation;
  - The evaluation can be both qualitative and quantitative;
  - The overall user-experience plays a role;
  - However, users are not always consistent.

Online evaluation

- Direct user feedback



- A/B testing

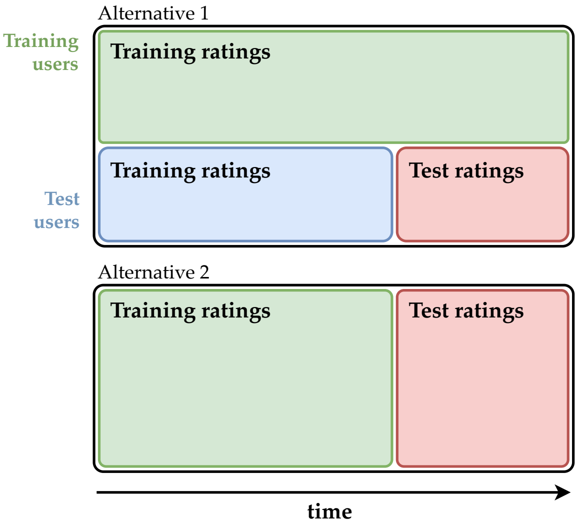


Users directly provide a feedback about the recommendation through questionnaires or self-reports.

The recommender is tested against a baseline (usually a previous version of the system) on two set of users: a control set that uses the baseline and the variation set that uses the new system. The improvements are evaluated in terms of standard metrics or through users feedback.

Offline evaluation: how to partition a dataset

- *Alt. 1* avoids the cold-start problem, i.e., no users are unknown at testing time;
- *Alt. 2* randomly selects test ratings (can have cold-start users);
- When available, it is good practice to split training-test ratings on the basis of the timestamp.



Offline evaluation for rating prediction

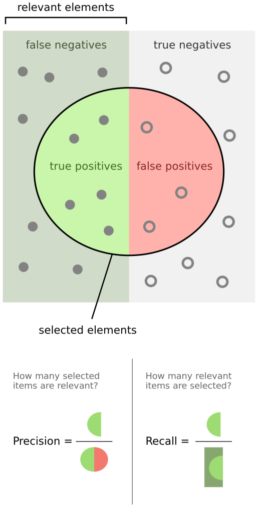
Given the predicted ratings $\hat{r}_{ui}$, for (u, i) in the test set (Te), then the usual evaluation metrics are:

- **MAE (Mean Absolute Error)** $= \frac{1}{|Te|} \sum_{(u,i) \in Te} |r_{ui} - \hat{r}_{ui}|$
- **MSE (Mean Squared Error)** $= \frac{1}{|Te|} \sum_{(u,i) \in Te} (r_{ui} - \hat{r}_{ui})^2$
- **RMSE (Root Mean Squared Error)** $= \sqrt{\frac{1}{|Te|} \sum_{(u,i) \in Te} (r_{ui} - \hat{r}_{ui})^2}$

Offline evaluation: top-N (1/3)

- **Recall**
  $$\frac{\#\ \text{relevant recommended}}{\#\ \text{relevant items}} = \frac{TP}{FN+TP}$$
- **Precision**
  $$\frac{\#\ \text{relevant recommended}}{\#\ \text{recommended items}} = \frac{TP}{FP+TP}$$

NOTE: All these metrics can be limited to the first k retrieved items to give more emphasis to the top of the list.



Offline evaluation: top-N (2/3)

- **AUC** $= \frac{1}{N^+N^-}\sum_i\sum_j [\![s(i) > s(j)]\!]$
  - $N^+ = \#$ relevant items and $N^- = N - N^+$;
  - $s(i)$ is the score given by the recommender to the item i;
  - Computes the number of miss ordered pairs of items in the ranking;
  - Considers all the positions in the list as equally relevant;
  - Often in RecSys AUC is not the best choice.

- **AP** (Average Precision) $= \frac{\sum_k \text{Precison@k} \cdot \text{rel}(k)}{\#\text{ relevant items}}$
  - $\text{rel}(k) \in \{0, 1\}$ indicates whether the k-th item is relevant or not;
  - As for precision and recall it can be truncated (AP@k);
  - mAP (mean AP) is the mean AP over all users.

Offline evaluation: top-N (3/3)

- **DCG@p** (Discounted Cumulative Gain) $= \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$
  - rel_i is the graded relevance of i, usually $\in \{0, 1\}$;
  - Most useful at top ranks;
  - Utility decreases quite fast (proportionally to the rank);
  - The normalized version (nDCG) is divided by the DCG of the ideal rank.
- **MRR** (Mean Reciprocal Rank) $= \frac{1}{|Q|} \sum_{i \in Q} \frac{1}{rank_i}$
  - $rank_i$ is the rank of i in the recommended ordered list;
  - Q is the set of positive test items;
  - Similarly to nDCG it is useful at top ranks;
  - It has usually higher values than both mAP and nDCG.

Offline evaluation: beyond accuracy

- **Diversity**: given a retrieved set R of m items and given a similarity measure between items *sim* then

$$diversity = \frac{\sum_{i \in R} \sum_{j \in R, j \neq i} 1 - sim(i,j)}{m(m-1)}$$

- **Novelty** $= \frac{\# \text{ relevant and unknown items}}{\# \text{ recommended relevant}}$, approximately the inverse of the popularity of the retrieved items

$$novelty = \frac{\sum_{i \in TP} \log_2 \left(\frac{1}{popularity(i)} \right)}{|TP|}$$

Non-personalized RS: most popular

- The most popular item, i.e., **the one with the highest number of ratings**, is *k*;
- User *u* already interacted with *k*;
- The most popular item after *k* is *i* that will be recommended to *u*;
- *Note*: *k* can be recommended if the “re-consumption” is likely in the application domain.

	i				k
	3			1	2
	4	2		2	3
u				1	2
			5		
	3		1		5
Popularity	3	1	2	3	4

Non-personalized RS: highest rated

- The highest rated item, i.e., **the one with the highest average rating**, is *i* that will be recommended to *u*;
- It is good practice to take into account also the number of ratings;
- Usually, a normalization factor is added to the average in order to give a bias towards popular items.

	i				
	3			1	2
	4	2		2	3
u				1	2
			5		
	3		1		5
Rating avg.	3.3	2	3	1.3	3

Overview of the standard RS approaches (1/2)

- Content Based (CB)
  - Recommend the most similar items to the ones the user liked in the past. E.g., same genre (movies), same artist (song), etc...
- Collaborative Filtering (CF)
  - Recommend to a user items liked by similar users, or viceversa, items that are similar to the ones liked in the past. In particular:
  - Item-item similarity: two items are similar if they share many users;
  - User-user similarity: two users are similar if they share may ratings;
  - **Remark**: in these approaches only the interactions are used to computed similarities. No specific users' and items' characteristic are used (see CB).

Overview of the standard RS approaches (2/2)

- Hybrid approaches
  - CB $>$ CF when there is no much history (cold-start problem);
  - CF $>$ CB when information about user-item interactions prevail on the explicit content;
  - Hybrid approaches tend to take advantage of the strength of the different methods while mitigating their weaknesses.
- Context-aware (CARS)
  - **Assumption**: the quality of a recommendation depends on the user (and item) state;
  - Recommender uses contextual information to tune the recommendation. E.g., the mood, the weather, the time, the presence of kids, etc...

Recap

In this lesson we have seen:

- What is a recommender system;
- RSs taxonomy;
- Types of feedback;
- Rating matrix and its properties;
- Recommendation tasks;
- Evaluation;
- Non-personalized RSs.

Try this at home:

- Analysis of the sparsity and distribution of standard RS datasets, e.g., **MovieLens** and **Netflix**;
- Application of non-personalized recommenders.