Design and Evaluation of Recommender Systems

Paolo Cremonesi *Politecnico di Milano*
Pearl Pu *EPFL*

Outline

1. Design of Recommender Systems
2. Off-line evaluation
3. On-line evaluation

Outline

1. Design of Recommender Systems

- Goals and motivations
  - Benefits for the service provider
  - Benefits for the users
- How RSs work
- Design space
  - Application domains
  - Non-functional requirements
  - Functional requirements: quality of a RS

Outline

1. **Design of Recommender Systems**
   - Goals and motivations
     - Benefits for the service provider
     - Benefits for the users
   - How RSs work
   - Design space
     - Application domains
     - Non-functional requirements
     - Functional requirements: quality of a RS

The goal of a RS: **service provider**

- Persuasion
- Prediction

The goal of a RS: **service provider**

- **Persuasion**
  - Increase sales (pay per usage)
  - Increase customer retention (subscription based)
  - Increase user viewing time (ads based)
- How do we obtain this goal?
  - By recommending items that are likely to suit the user's needs and wants

$\downarrow$

  - Goals for the user

The goal of a RS: **service provider**

- **Prediction**
  - Predict user behavior, without recommending
- Sizing
  - How many items should I have in stock in order to satisfy the expected demand?
  - How many users will watch a TV program?
- Pricing
  - How many users will watch a TV commercial?
- Marketing
  - Better understand what the user wants

The goal of a RS: **service provider**

- Success of recommendations can be measured in two ways
- **Direct**
  - e.g., **conversion rate**: the ratio between number of sales made thanks to recommendations and number of recommendations
- **Indirect**
  - e.g., **lift factor**: increase in sales after the introduction of the recommender system

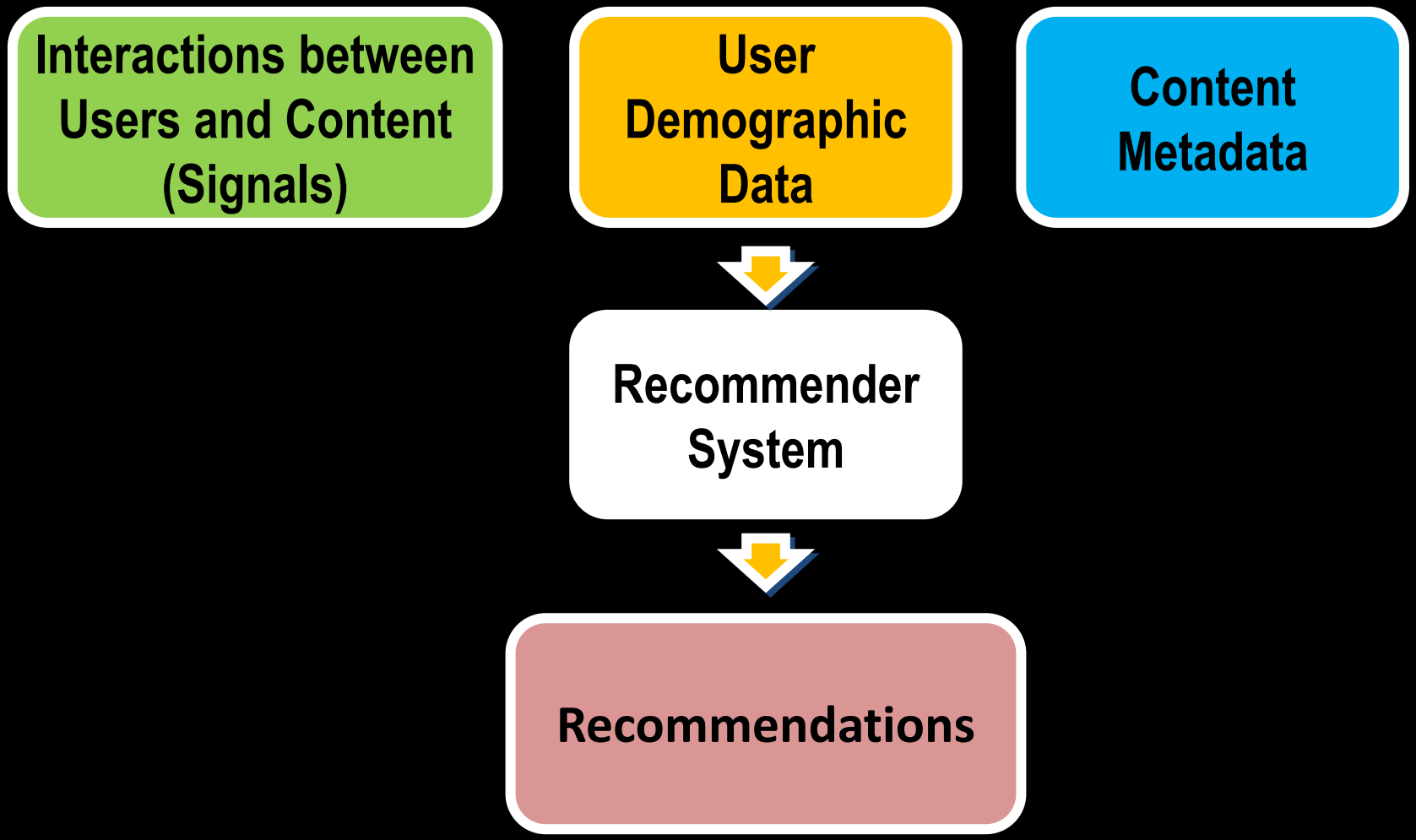
The goal of a RS: **users**

- Find **some** good items
- Find **all** good items
- Find a sequence or a bundle of items
- Understand if an item is of interest
- …

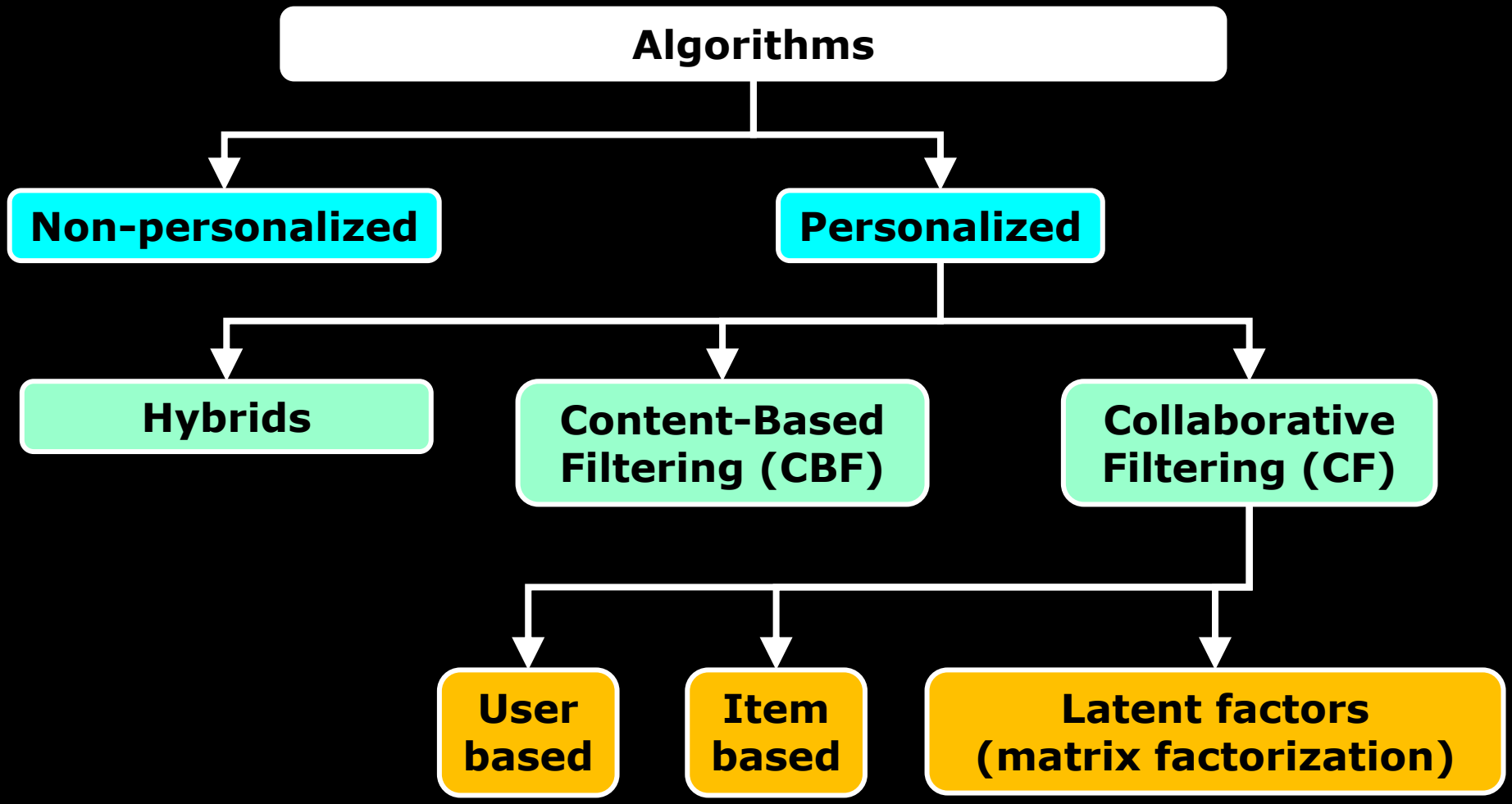
Outline

1. **Design of Recommender Systems**
   - Goals and motivations
     - Benefits for the service provider
     - Benefits for the users
   - How RSs work
   - Design space
     - Application domains
     - Non-functional requirements
     - Functional requirements: quality of a RS

How RSs work


Interactions between Users and Content (Signals)
User Demographic Data
Content Metadata
Recommender System
Recommendations

How RSs work


Algorithms
Non-personalized
Personalized
Hybrids
Content-Based Filtering (CBF)
Collaborative Filtering (CF)
User based
Item based
Latent factors (matrix factorization)

Outline

1. **Design of Recommender Systems**
   - Goals and motivations
     - Benefits for the service provider
     - Benefits for the users
   - How RSs work
   - Design space
     - Application domains
     - Non-functional requirements
     - Functional requirements: quality of a RS

The design space

Depends on:

- **Functional requirements**
  - Goal and motivations (service and user)
- **Application domain**
- **Non functional requirements**
  - Performance
  - Security

Application domains: what varies

- The **type of products** recommended
  (goods vs. services)
- The **amount** and **type** of information about the **items**
- The **amount** and **type** of information about the **users**
- **How** items are generated and **change** as **time passes**
- The **level of knowledge** users have about items
- The **task** the user is facing
  (e.g., buying a product, searching for a TV program)
- The **context**: where, when, with whom, …
  (tutorial in the afternoon)

Non-functional: performance

- **Adaptability**: ability to timely react to changes in
  - user preferences
  - item availability
  - contextual parameters
- Measured in term of recommendation quality for new users, items, signals, and contexts
- Depends on the performance of both
  - **training phase**
  - on-line recommendation phase

Non-functional: performance

- **Response time** of the on-line phase
  - correlated with shopper conversion and bounce rate
  - a 3 seconds slowdown on a web page (from 4 seconds up to 7 seconds) transforms a buyer into a bouncer
    - research performed on the Walmart web site

http://www.webperformancetoday.com/2012/02/28/4-awesome-slides-showing-how-page-speed-correlates-to-business-metrics-at-walmart-com/

Non-functional: performance

- **Scalability**: ability to provide good quality and timely recommendations regardless of
  - size of the dataset (users and items)
  - number of concurrent recommendations

Non-functional: performance

- **Scalability** is measured in terms of **maximum on-line recommendations per second**
  - can be measure **only** with load tests
  - a low response time is not necessary and not sufficient for scalability
    - a RS with all the logic embedded in the client-side may take 100 ms to provide recommendations, regardless of the number of users (scalable)
    - a RS with all the logic is the server, might require 10 ms to recommend one user and 10 secs to recommend 1000 users (not scalable)

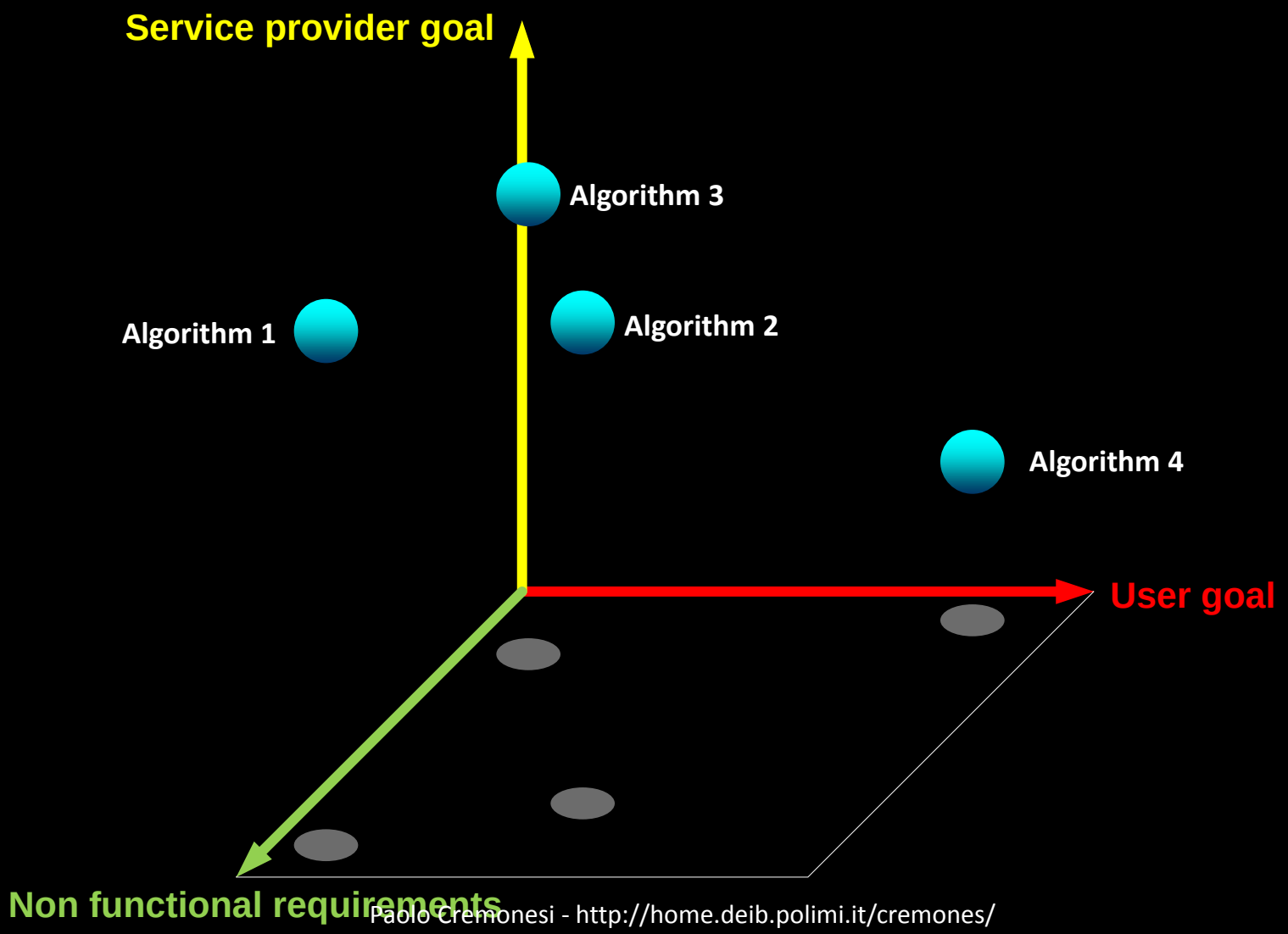
Non-functional: performance

- **CPU/memory/and storage** requirements
- Mostly important for
  - online phase
  - embedded systems
- Architectural choices
  - Recommendations computed server side *(higher load on server → lower scalability)*
  - Recommendations computed client side *(client CPU/memory constraints)*

Non-functional: security

- Privacy issues
  - E.g., User profiles cannot be stored in a centralized location or outside a country
  - Only some types on users' info can be collected and stored
- Security, robustness to attacks
  - To alter the outcome of a recommender systems (e.g., to promote/demote some items)
  - To identify users personal data

3D benchmark



The design space: **variables**

- Elicitation techniques
- Information on items
- Algorithms
- Interface: presentation and interaction
- …

The design space: **variables**

- **Elicitation techniques**
- Information on items
- Algorithms
- Interface: presentation and interaction
- …

Preference elicitation

- Whenever a new user joins a RS, the system tries first to learn his/her preferences
- This process is called **preference elicitation**

Preference elicitation

- Whenever a new user joins a RS, the system tries first to learn his/her preferences
- This process is called **preference elicitation**
- Why do we need to collect ratings?

Why do we need to collect ratings?

- **Community utility (cold-start)**
  - learn recommendation "rules" (e.g., users who like A like B)
  - algorithm **learning** phase (*learning rate*)
  - eliciting ratings for items that don't have many

Why do we need to collect ratings?

- **Community utility (cold-start)**
  - learn recommendation "rules" (e.g., users who like A like B)
  - algorithm **learning** phase (*learning rate*)
  - eliciting ratings for items that don't have many

vs.

- **New user utility**
  - apply recommendation rules (e.g., user Paolo likes sci-fi movies)
  - algorithm **recommendation** phase
  - eliciting ratings from new users

Design space: preference elicitation

- The effectiveness of the **elicitation** process depends on its **goal**
  - Community utility vs. User utility

Design space: preference elicitation

- The effectiveness of the **elicitation** process depends on its goal and **strategy**
  - Community utility vs. User utility
  - Human vs. System Controlled
  - Explicit vs. Implicit

Design space: preference elicitation

- The effectiveness of the **elicitation** process depends on its goal and **strategy**
  - Community utility vs. User utility
  - Human vs. System Controlled
  - **Explicit** vs. Implicit
  - Type of information collected (Ratings, Free Text, User Requirements, User Goals, Demographics)

Design space: preference elicitation

- The effectiveness of the **elicitation** process depends on its goal and **strategy**
  - Community utility vs. User utility
  - Human vs. System Controlled
  - **Explicit** vs. Implicit
  - Type of information collected (**Ratings**, Free Text, User Requirements, User Goals, Demographics)
  - Rating scale (Thumbs up/down, 5/10 stars, …)
  - Profile length (# of ratings in the user profile)

Outline

1. **Design of Recommender Systems**
   - Goals and motivations
     - Benefits for the service provider
     - Benefits for the users
   - How RSs work
   - Design space
     - Application domains
     - Non-functional requirements
     - Functional requirements: **quality of a RS**

Quality of RSs

- Different quality levels:
  - data quality (content metadata and user profiles)

    ↓
  - algorithm quality

    ↓
  - presentation and interaction quality

    ↓
  - perceived qualities

Quality indicators of a RS

- Relevance
- Coverage
- Diversity
- Trust
- Confidence
- Novelty
- User opinion
- Attractiveness
- Context compatibility
- Serendipity
- Utility
- Risk
- Robustness
- Learning rate
- Usability
- Stability
- Consistency
- …

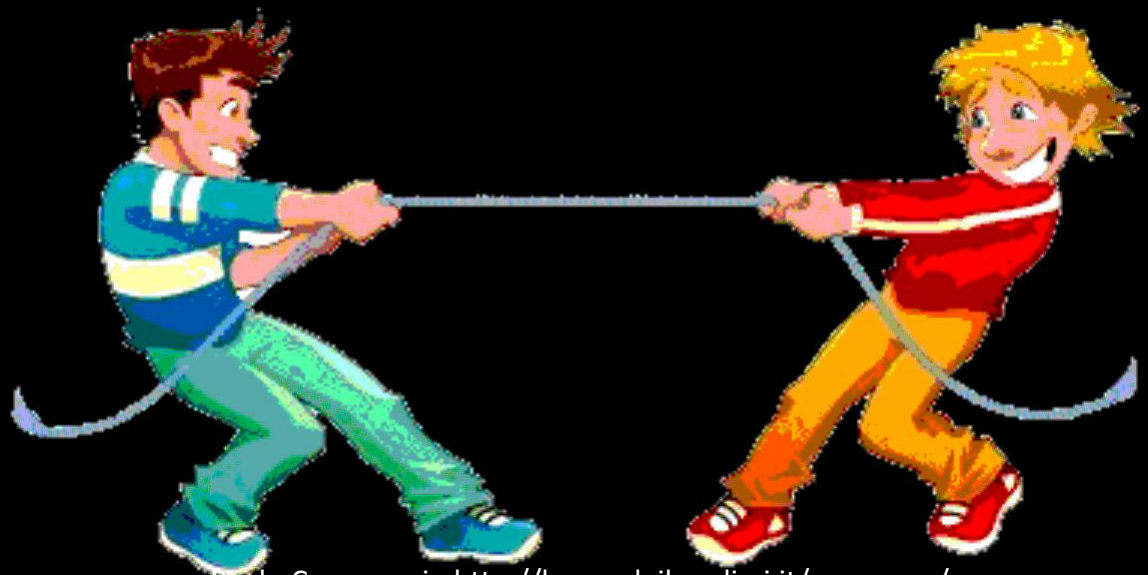
Design **tradeoffs**

- Quality of recommendations vs.
  - elicitation effort
  - cost of data enrichment
  - computational complexity
  - …

Design tradeoffs: **example**

The system must collect **enough** information in order to learn user's preferences and improve utility of recommendations

Gathering information adds a **burden** on the user, and may negatively affect the user experience



SYSTEM: Getting more information

USER: Reducing the effort

Design tradeoffs: **example 1**

- Rating scale:
  thumbs up/down, 5 stars, 10 stars, … ?

- Users' rating effort and noise increase as they have more rating choices

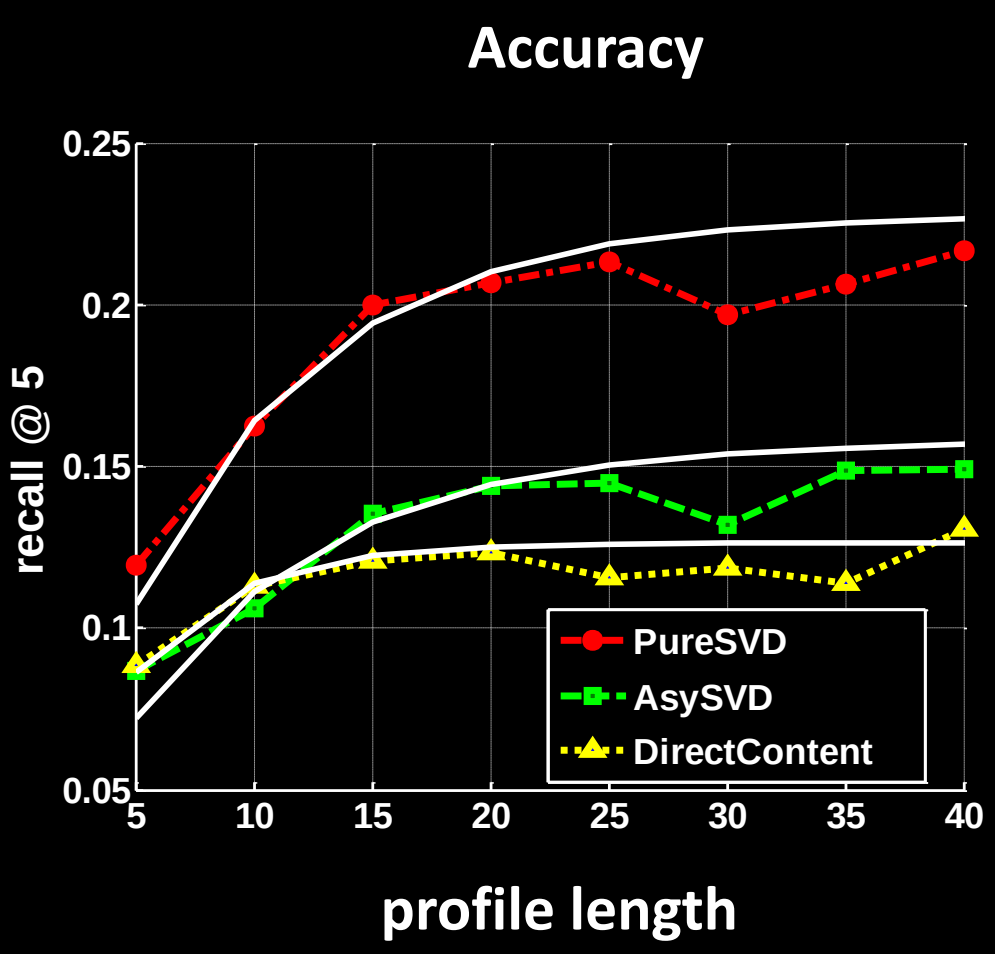
Vs.

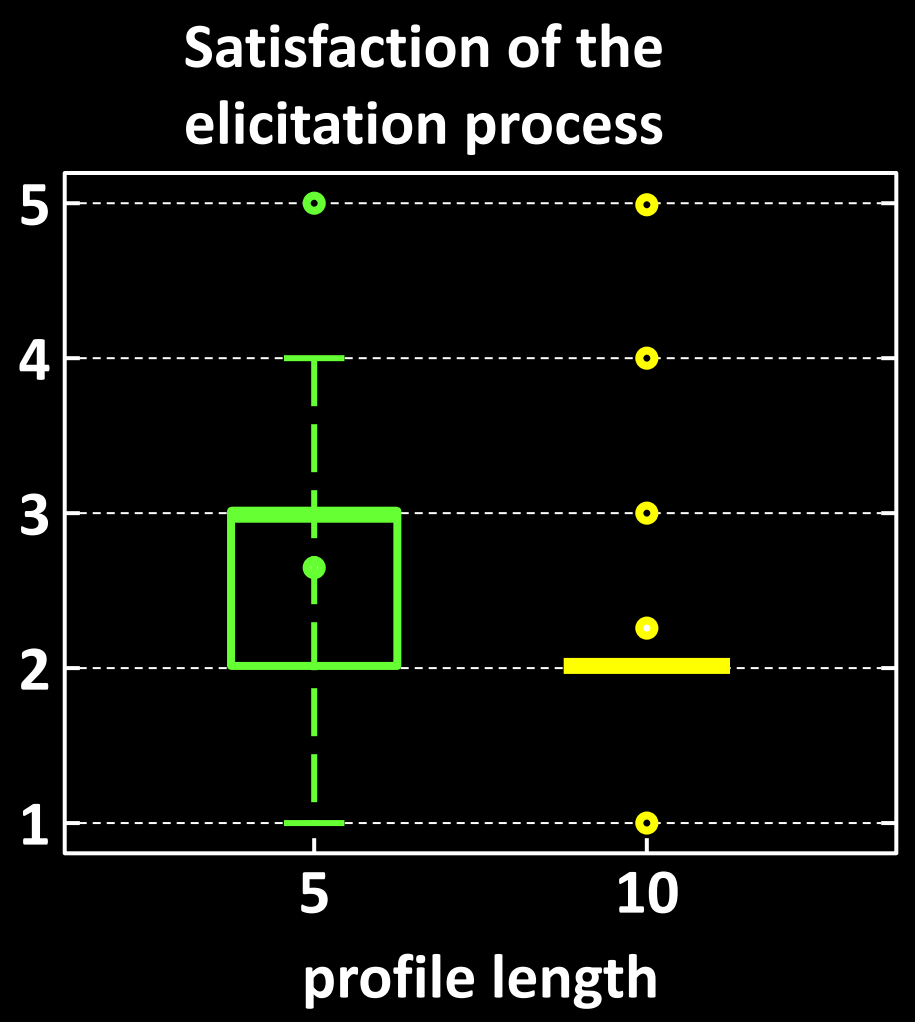
- Higher precision rating scale improve accuracy

E. I.Sparling, S.Sen. **Rating: how difficult is it?**. In *RecSys '11*

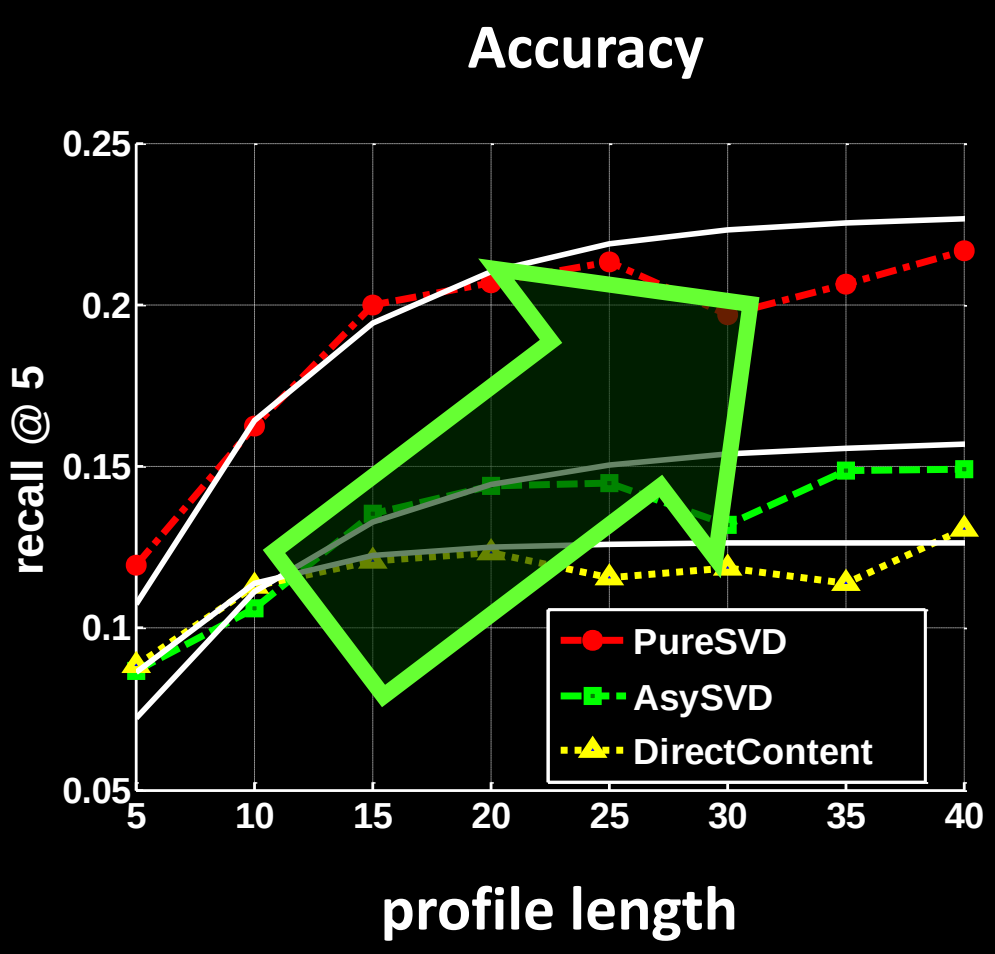
D.Kluver, T.T. Nguyen, M.Ekstrand, S.Sen, J.Riedl.
How many bits per rating?. In *RecSys '12*

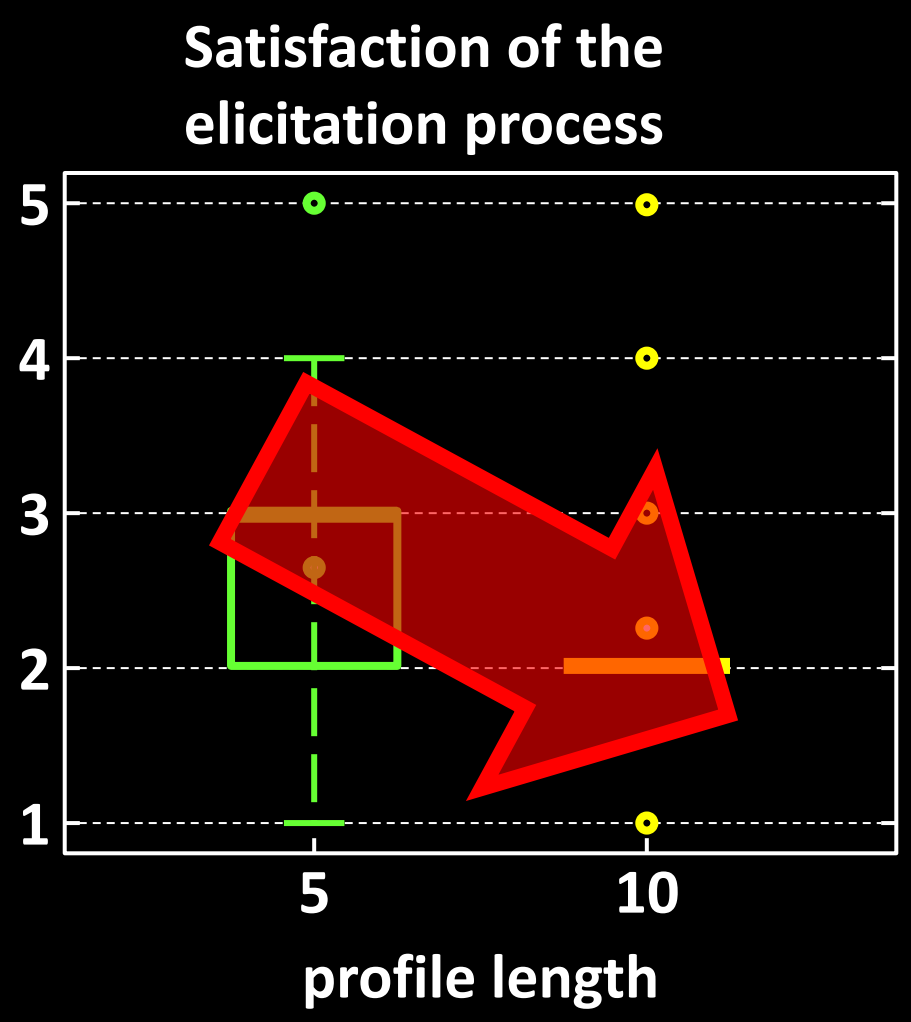
Design tradeoffs: **example 2**



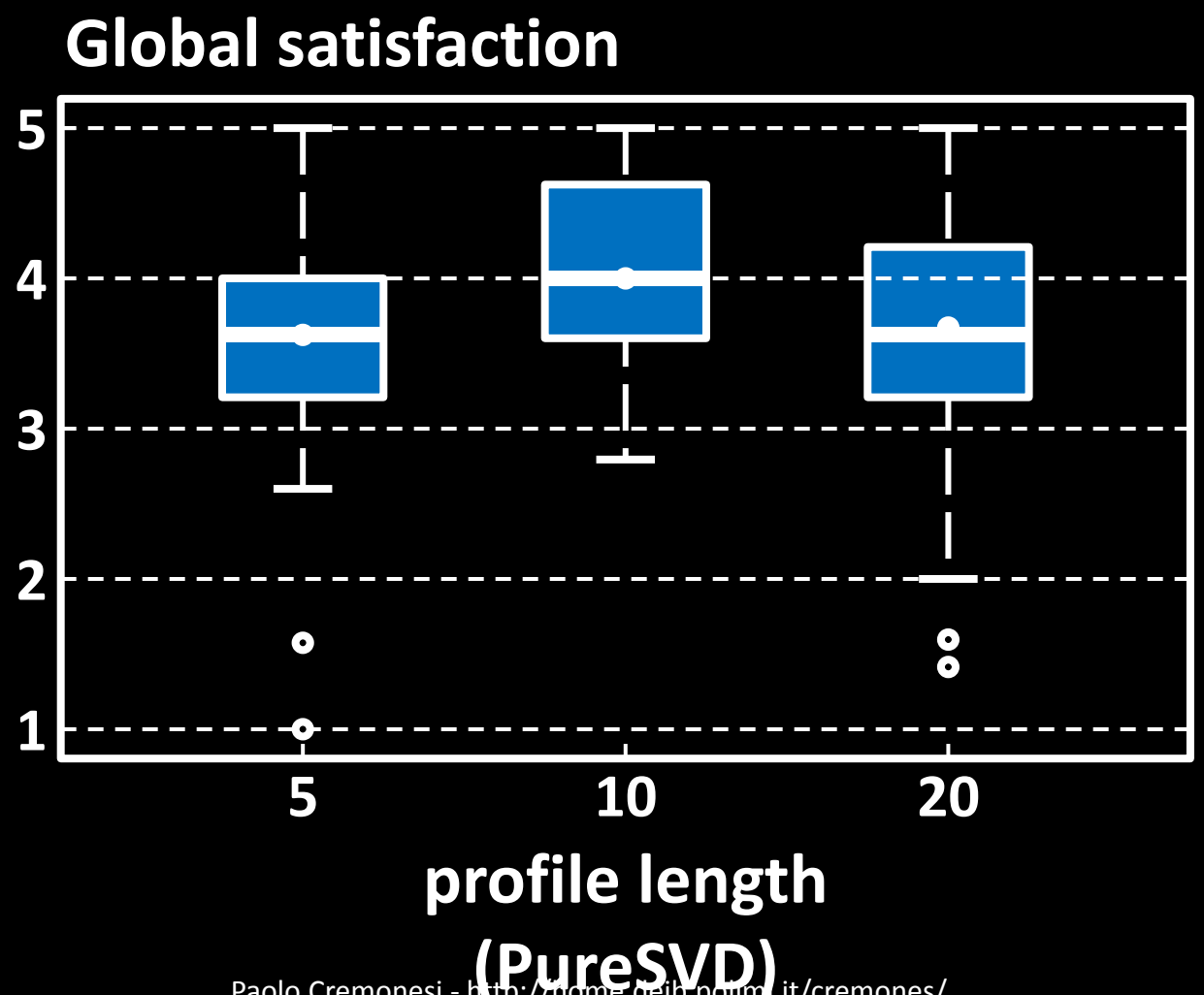


Design tradeoffs: **example 2**

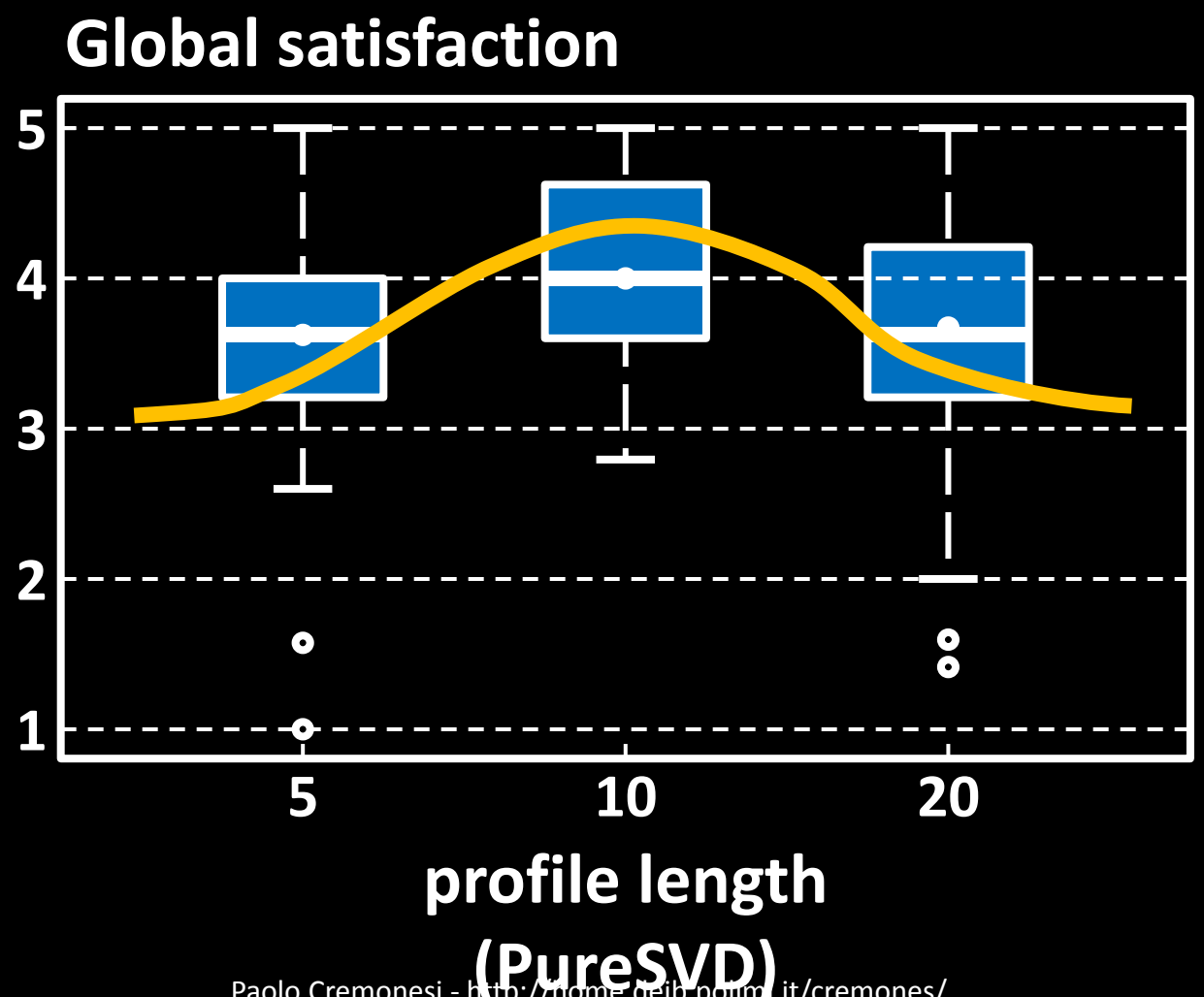




Design tradeoffs: **example 2**



Design tradeoffs: **example 2**



Taxonomy of evaluation techniques

- System-centric evaluation (**off-line**)
- User-centric evaluation (**on-line**)

Taxonomy of evaluation techniques

- System-centric evaluation (**off-line**)
  - The system is evaluated against a prebuilt ground truth dataset of opinion
  - Users do not interact with the system under test
  - Comparison between the opinion as estimated by the RS and the judgments previously collected from real users

Taxonomy of evaluation techniques

- User-centric evaluation (**on-line**)
  - Users interact with a running recommender system and receive recommendations
  - Feedback from the users is then collected
    - **Subjective analysis (explicit)**: interviews, surveys
    - **Objective behavioral analysis (implicit)**: system logs analyses
  - Laboratory studies vs. Field studies
  - A/B comparison of alternative systems (both survey and objective measures)

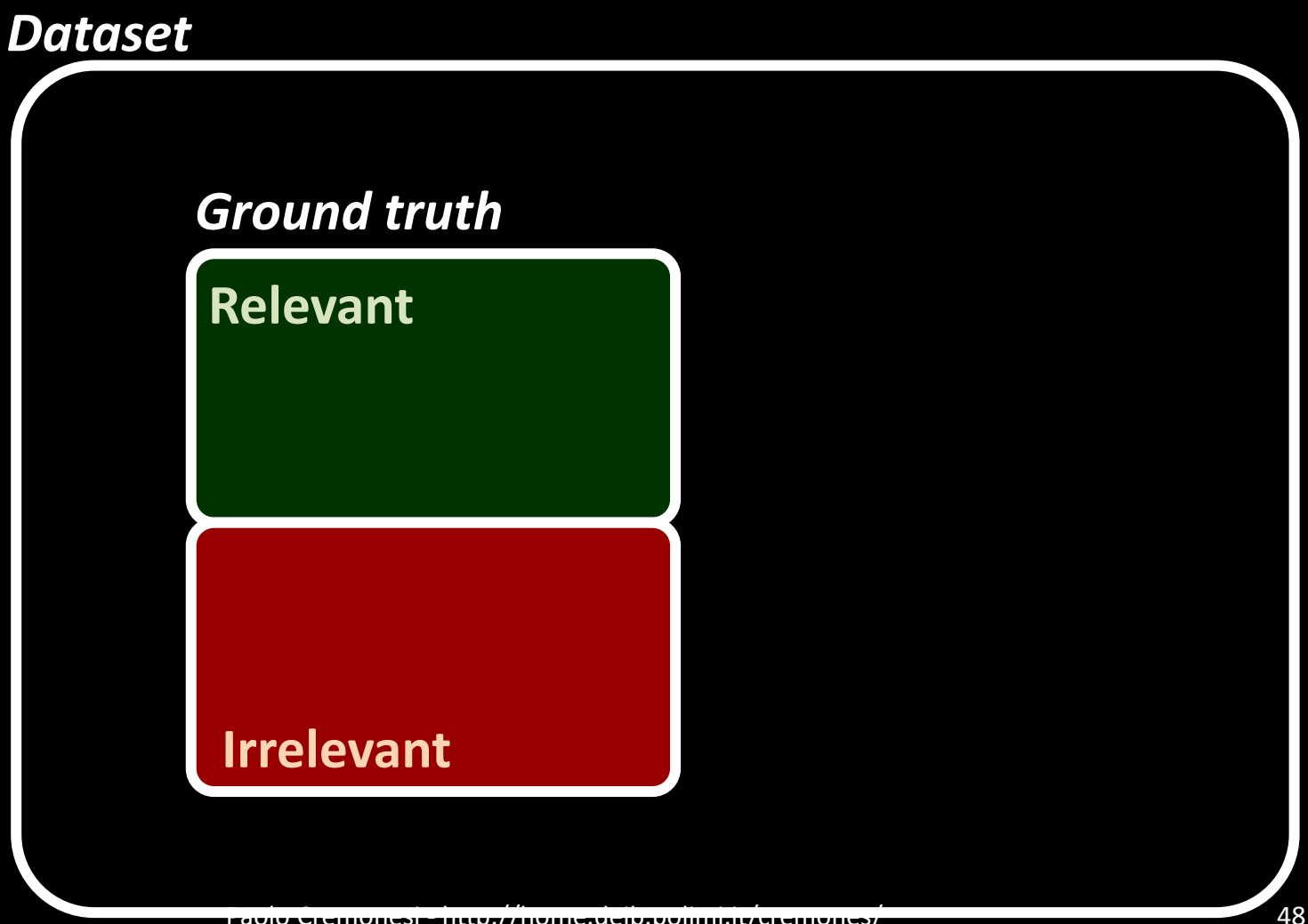
Outline

1. Design of Recommender Systems
2. **Off-line evaluation**
3. On-line evaluation

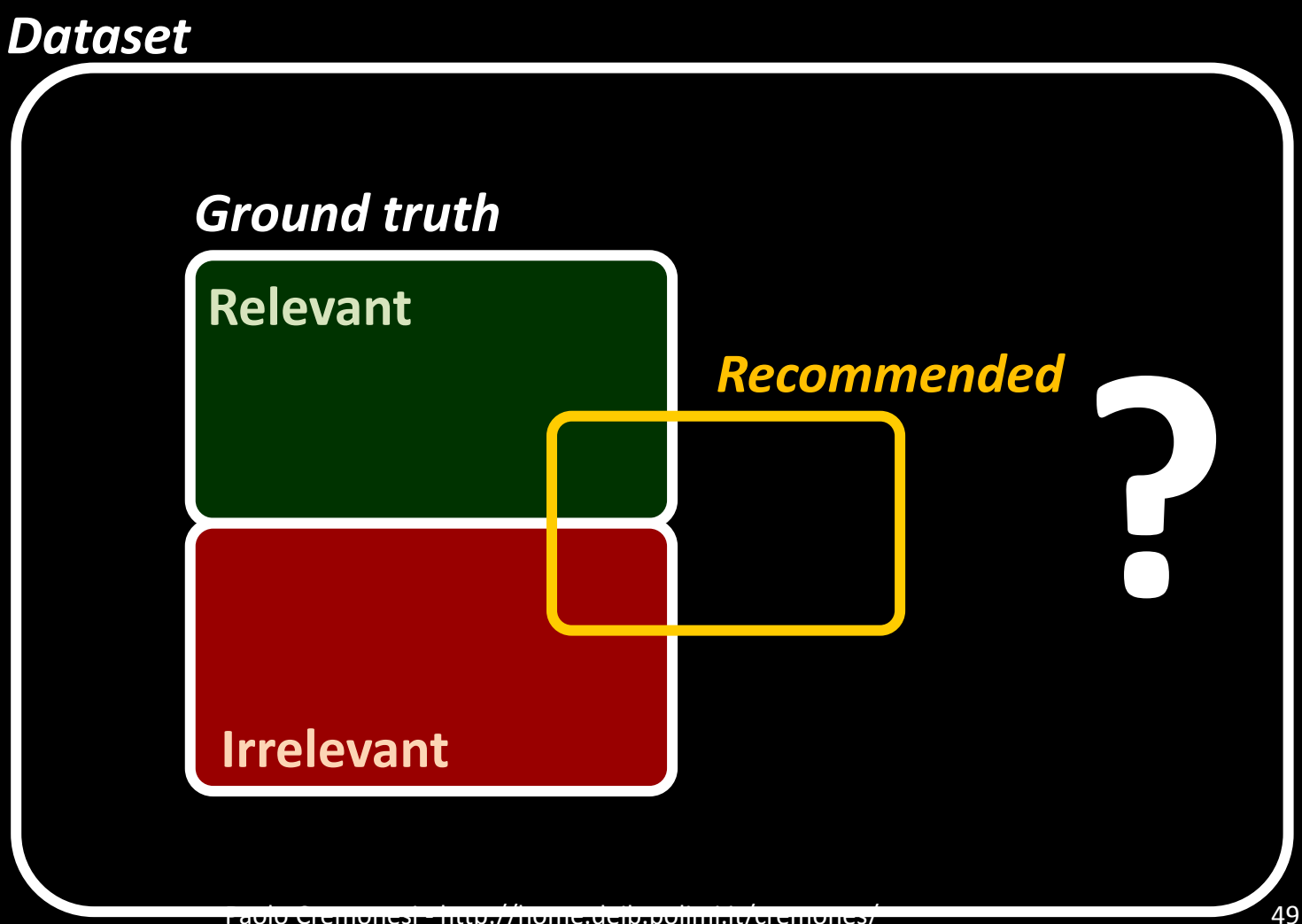
Dataset partitioning

Dataset

Dataset



Dataset



Dataset partitioning

- Off-line evaluation metrics require to **partition** the user-rating-matrix (URM) into **three** parts
  - Training
  - Tuning
  - Testing

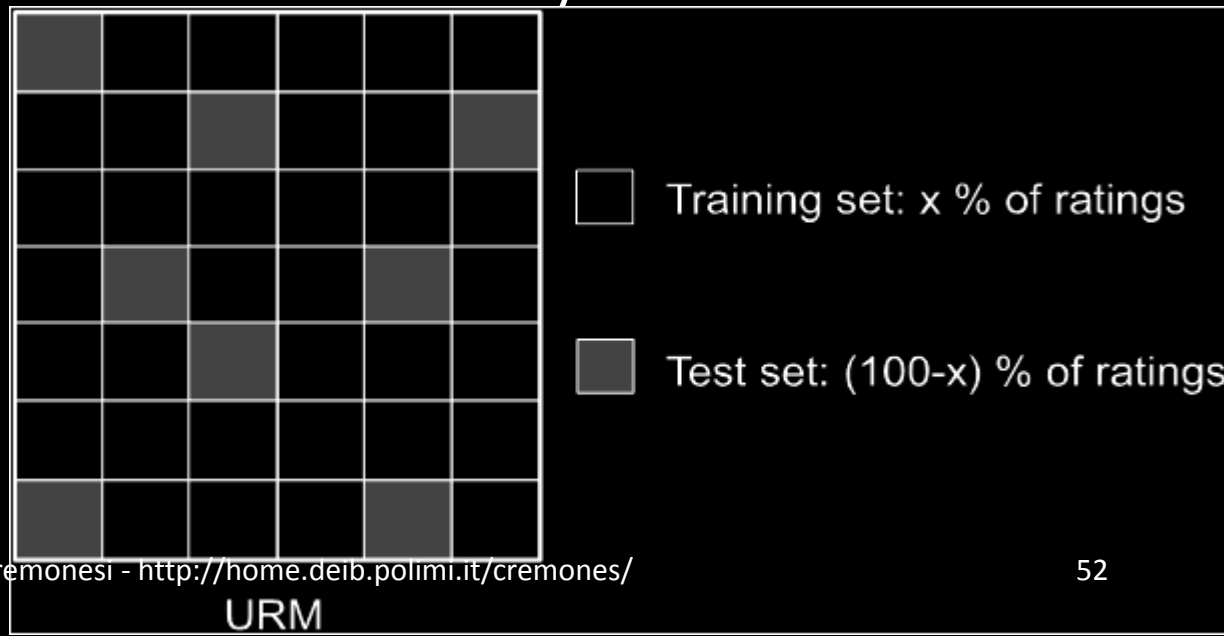
- The partitions should be disjoint …
  - … otherwise they are not partitions

Partitioning techniques

- Different techniques can be used to partition the URM:
  - Hold-out
  - K-fold
  - Leave-one-out

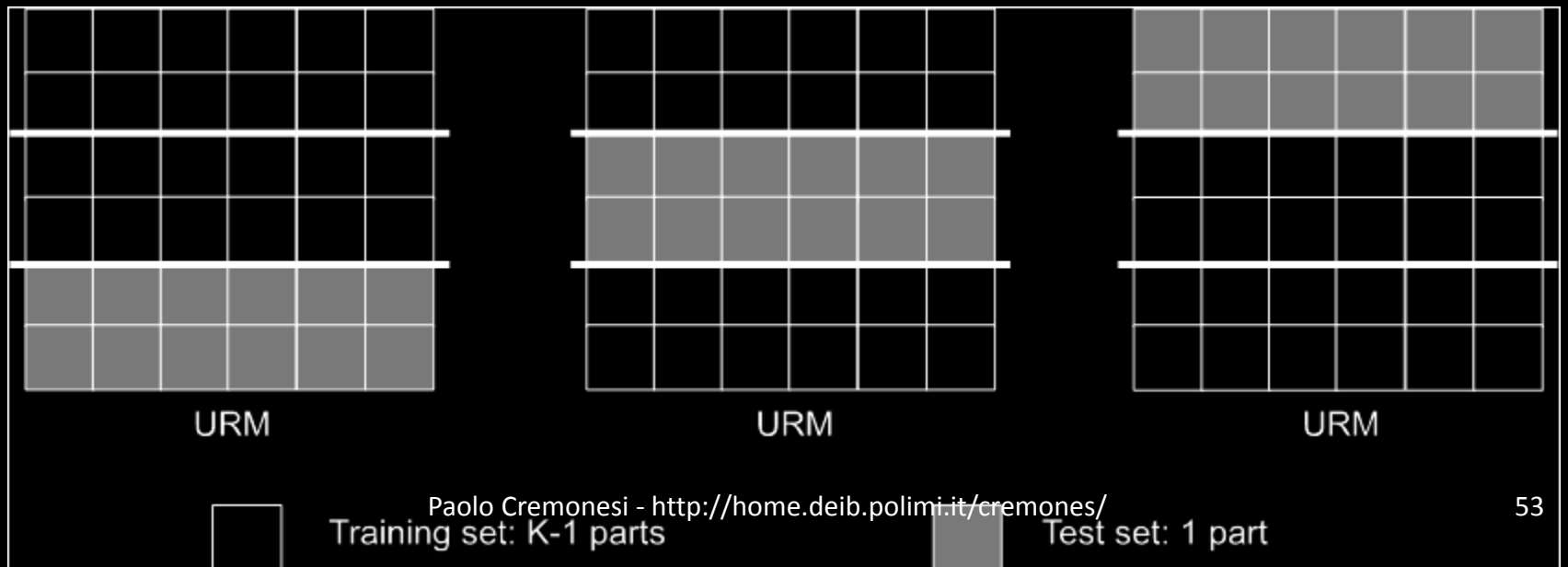
Hold-out

- A random set of ratings is withheld from the URM and it is used as test set
- The remaining ratings are used as training set
  - Modifies the user profiles
  - Overfitting: tested users are not totally unknown to the model



K-fold

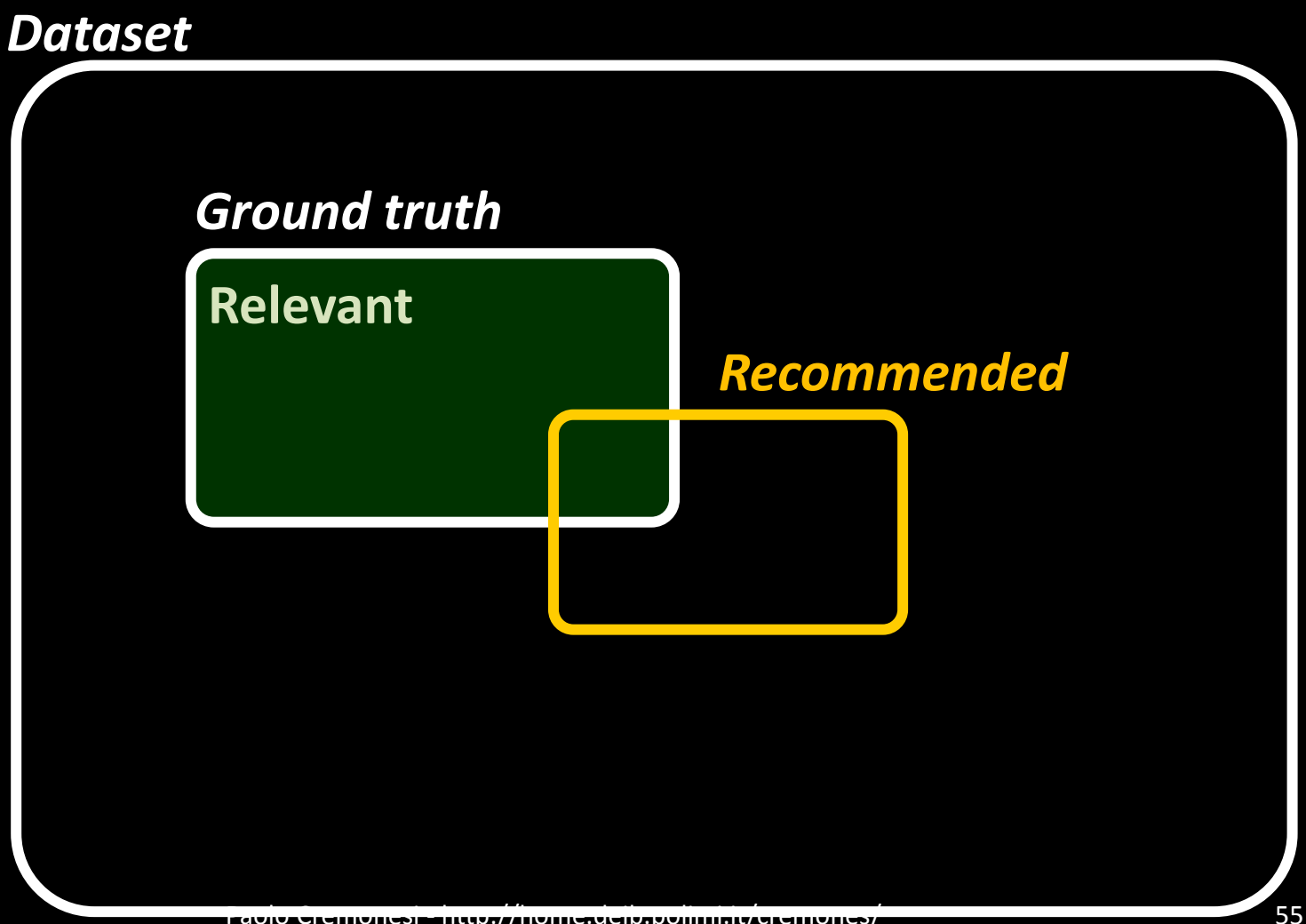
- Users of the URM are divided into *K* partitions (*K*=10)
  - *K*-1 partitions are used for the raining and the remaining partition is used for the test
  - tested users are unknown to the system because they are not used to build the model



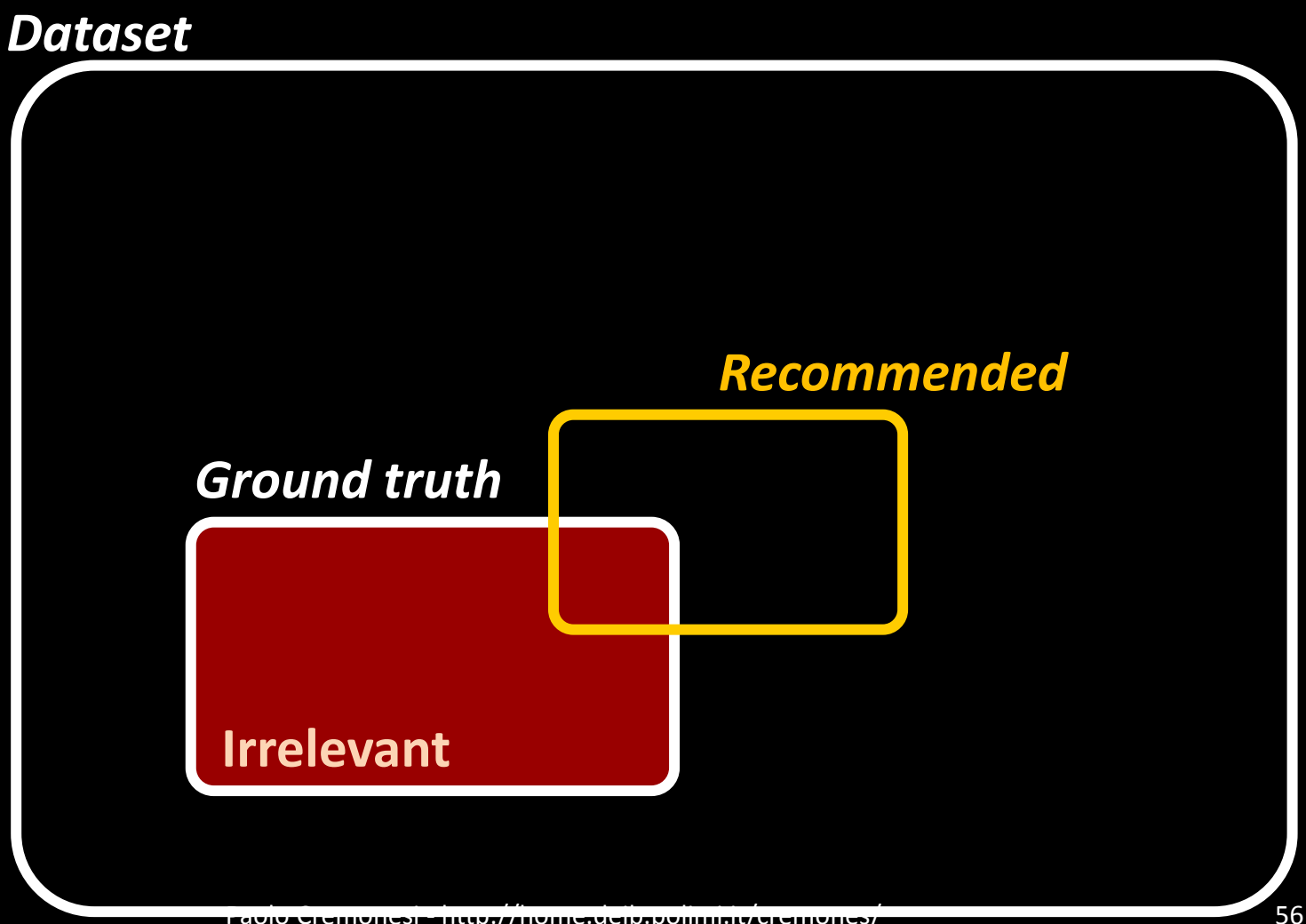
Leave-one-out

- During the testing, for each user we test **one** rated item at a time from the test set

Dataset partitioning



Dataset partitioning



Off-line analysis: quality

With off-line analysis we can try to measure:

- Accuracy
- Confidence
- Coverage
- Diversity
- Novelty
- Serendipity
- Stability and Consistency

Accuracy

- Comparison between **recommendations** and a predefined set of "correct" opinions of users on items (the "**ground truth**")
- Depending on the goal, different methods are used for measuring accuracy
  - **error**: which rating a user gives to an item
  - **classification**: which items are of interest to a user
  - **ranking**: what is the ranking of items (from most interesting to least interesting) for a user

Predicting user ratings: error metrics

$$\mathbf{RMSE} = \sqrt{\frac{\sum_{u,i \in T} \left(r_{ui} - \hat{r}_{ui}\right)^2}{|T|}} \qquad \mathbf{MAE} = \frac{\sum_{u,i \in T} \left|r_{ui} - \hat{r}_{ui}\right|}{|T|}$$

r_{ui} = rating of user u item i

T = test set

- Hypothesis:
  all the ratings are **missing-at-random**
  – implied by testing on observed ratings only

Recommending interesting items: classification metrics

$$\text{recall} = \frac{\text{\# relevant recommended items}}{\text{\# tested relevant items}}$$

$$\text{fallout} = \frac{\text{\# irrelevant recommended items}}{\text{\# tested irrelevant items}}$$

$$\text{precision} = \frac{\text{\# relevant recommended items}}{\text{\# recommended items}}$$

Precision

- **All-Missing-As-Negative** (AMAN) hypothesis
  - all missing ratings are irrelevant
- Precision with AMAN hypothesis underestimate the true precision computed on the (unknown) complete data

Harald Steck,

Training and testing of RSs on data missing not at random. In KDD '10

Item popularity and recommendation accuracy. In RecSys '11

Recall

- **Missing-Not-At-Random** (MNAR) hypothesis
  - relevant ratings are missing at random
  - non-relevant ratings are missing with a higher probability than relevant ratings
- Recall with MNAR hypothesis is a (nearly) unbiased estimate of recall on the (unknown) complete data
  - much milder than assuming that
    - all the ratings are missing at random (MAE and RMSE)
    - all missing ratings are irrelevant (Precision)

Precision vs. Recall

- Precision is not appropriate, as it requires knowing which items are undesired to a user
  - algorithms my suggest relevant but unrated items

- Positively/Negatively rating an item is an indication of liking it, making recall/fallout measures applicable

Combing recall and precision

- Mean Average Precision (MAP) = compute the average precision across several different levels of recall
- F-measure = harmonic mean between precision and recall

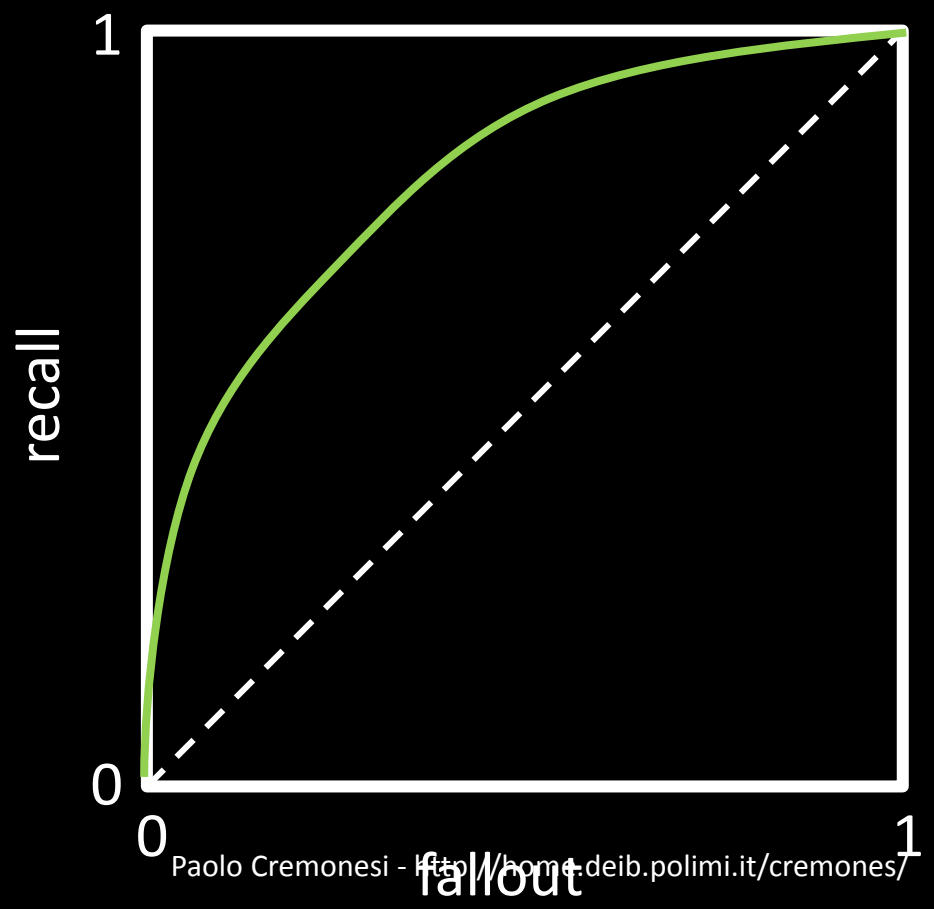
$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Combing recall and fallout: ROC

- Special ROCs
  - Global ROC (GROC)
    - different users are allowed to receive different number of recommendations
  - Customer ROC (CROC) is a variant in which
    - all users receive the same number of recommendations
    - all-missing-as-negative

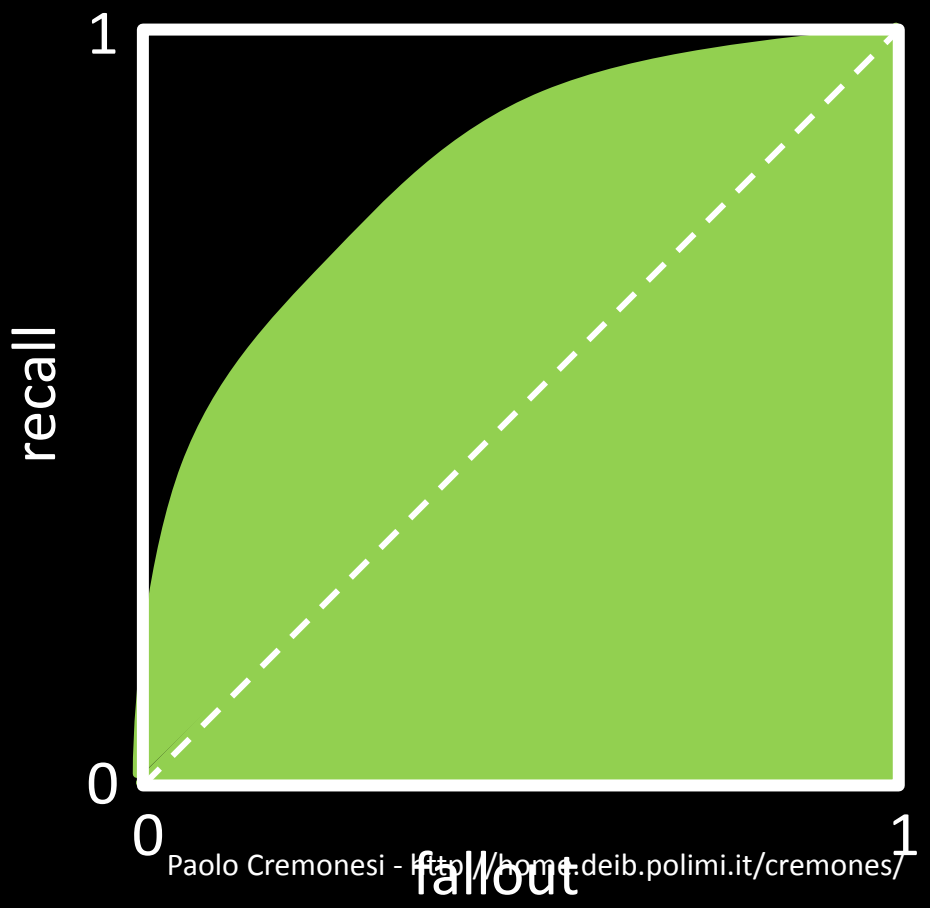
Schein et al.,
Methods and metrics for cold-start collaborative filtering, SIGIR 2002

Combing recall and fallout: ROC



Combing recall and fallout: ROC

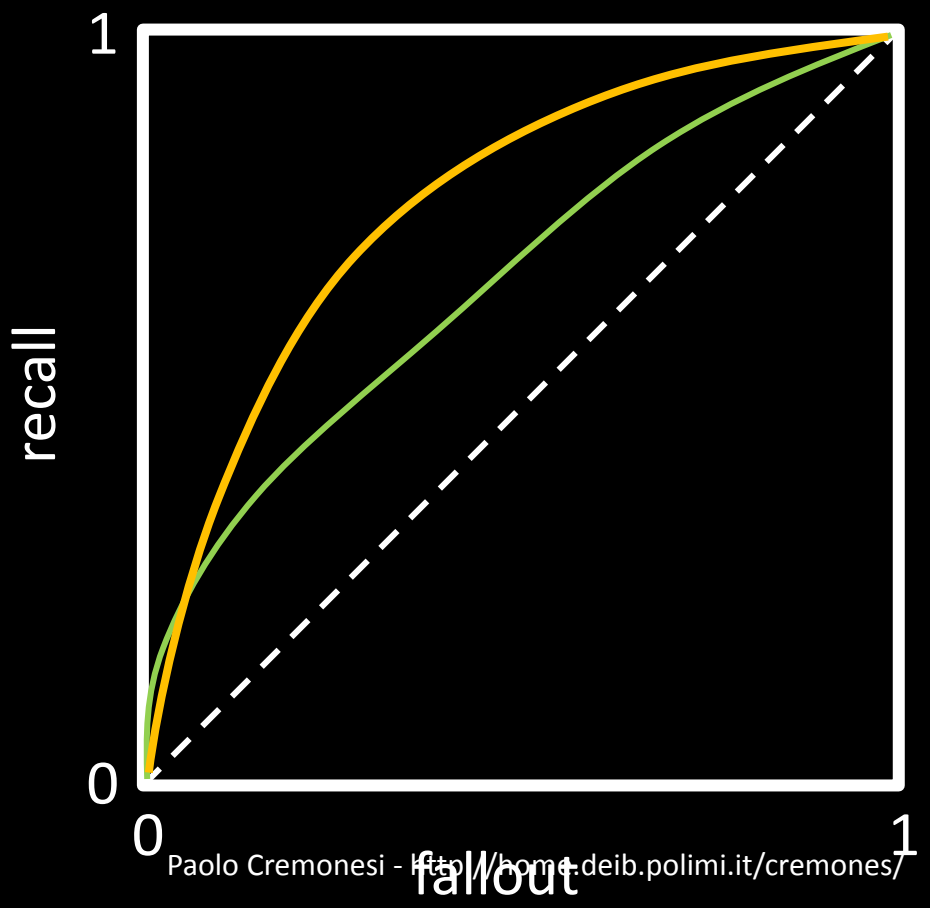
- Area Under Curve (AUC)



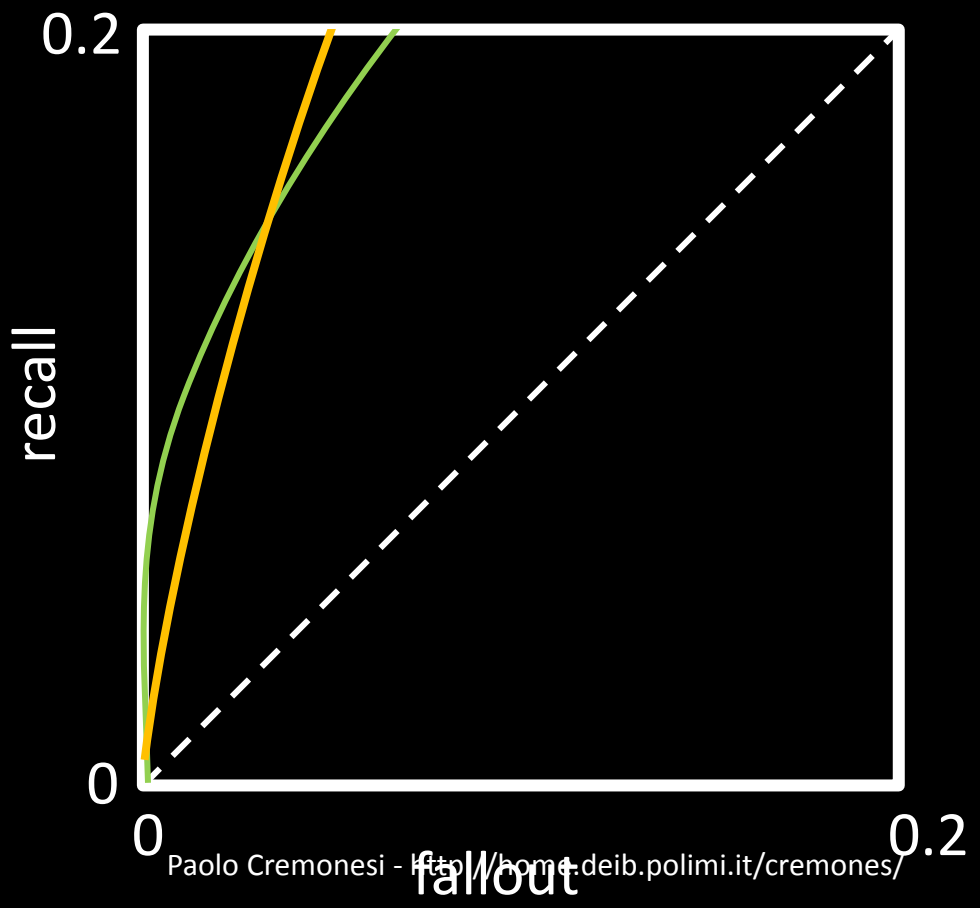
Combing recall and fallout: ROC

- Area Under Curve (AUC)
  - probability that the recommendation algorithm will rank a interesting item higher than a uninteresting one
  - percentage number of pairs of (positive, negative) items correctly ordered
- Can be computed only on rated item pairs
  - or on all item pairs with all-missing-as-negatives (CROC)

Combing recall and fallout: ROC



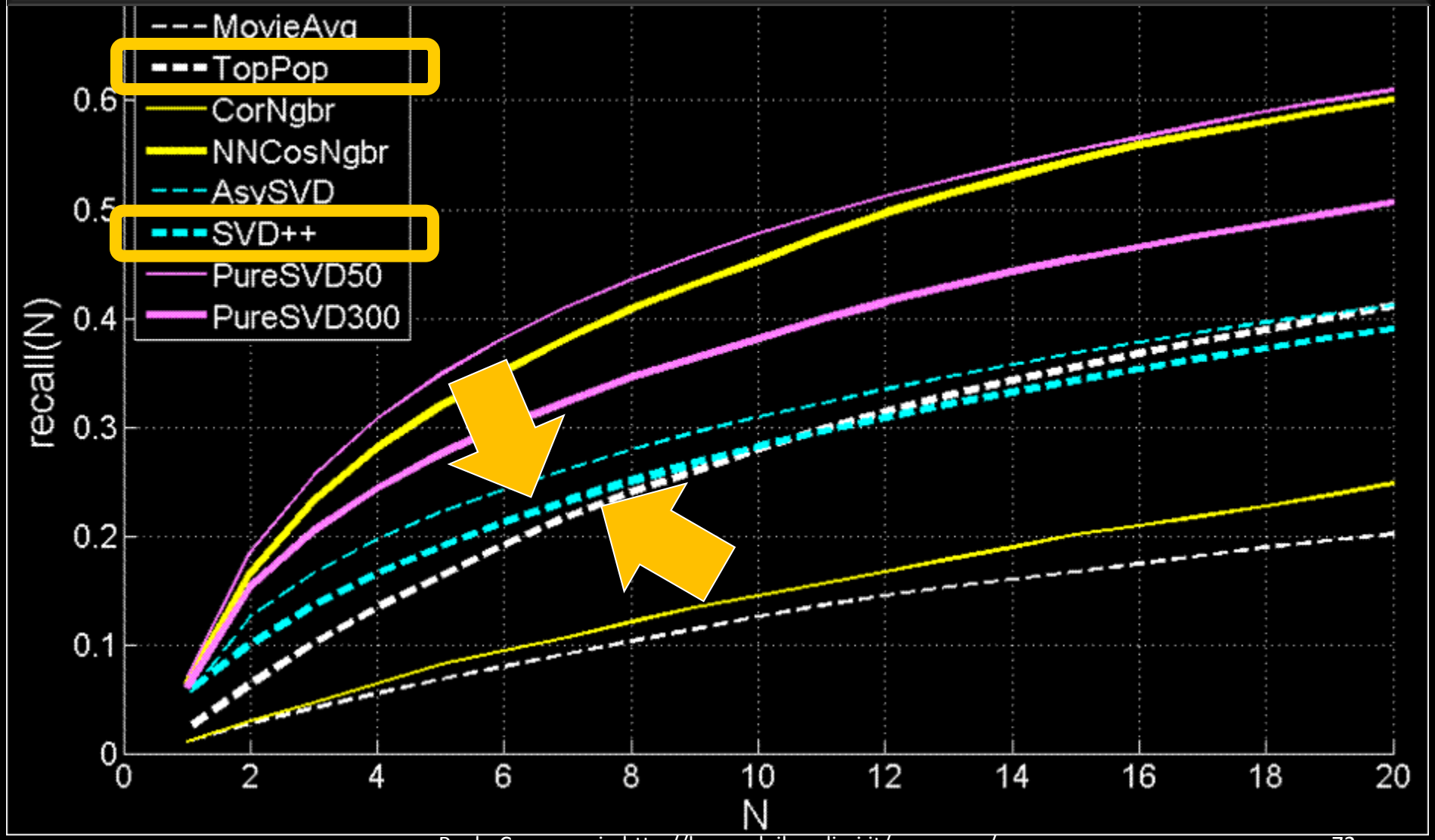
Combing recall and fallout: ROC



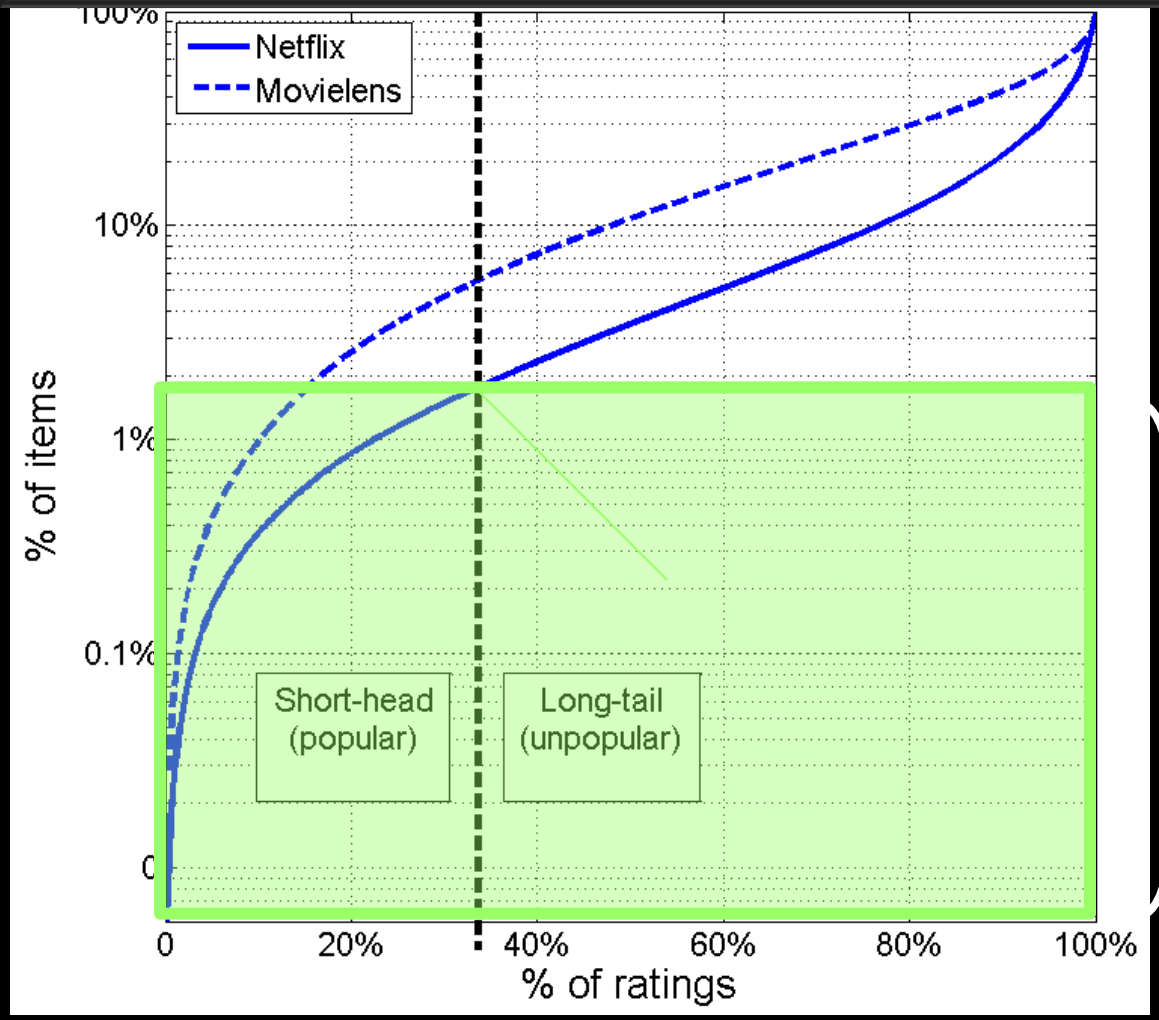
Item Popularity Bias

- Skewed datasets (datasets with a heavy short-head) contains lots of popular items
  - Popular items are best recommended by trivial recommender algorithms
    - Low novelty and serendipity
    - Low utility for both users and providers
  - Even sophisticated algorithms learn on the majority of ratings (top popular)

Netflix: recall



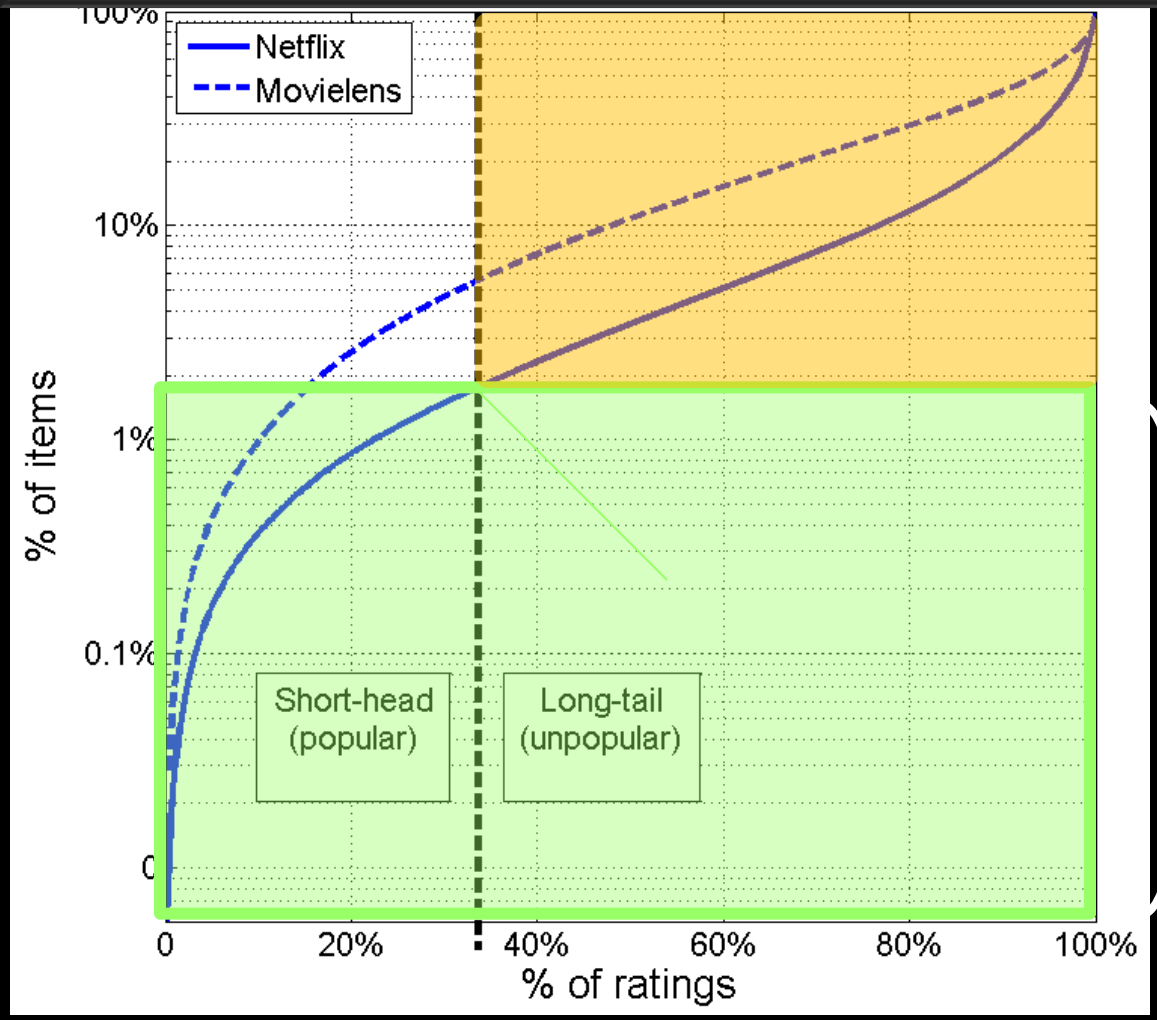
Rating distribution



Short-head
2% of items

33% of ratings

Rating distribution

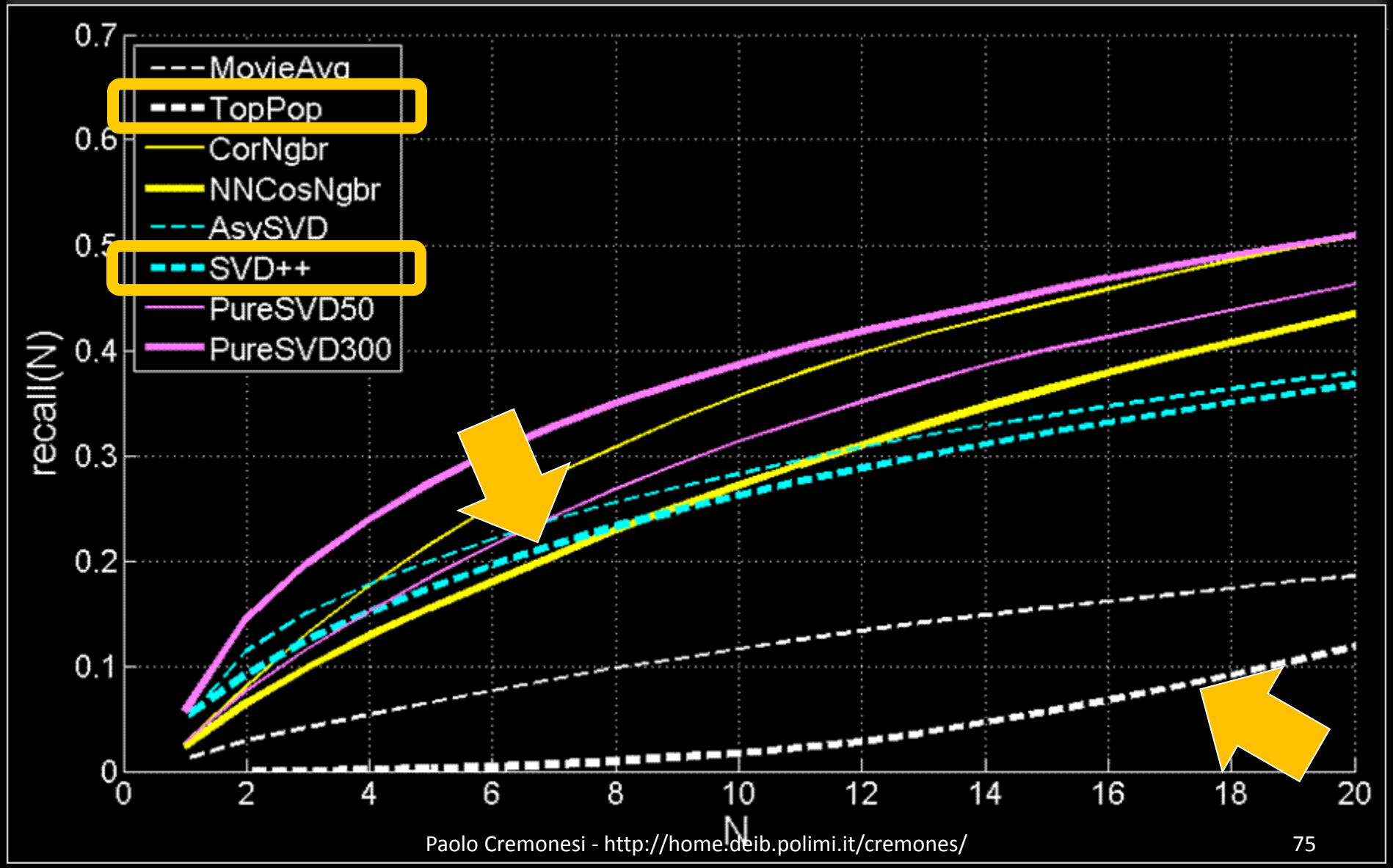


Long-tail test set

Short-head 2% of items

33% of ratings

Netflix: recall (long-tail)



Ranking items: ranking metrics

- Used when an ordered list of recommendations is presented to users
  - “most relevant” items are at the top
  - “least relevant” items are at the bottom

- Metrics depends on the availability of a reference ranking (true ranking)

Ranking items: ranking metrics

- Average Reciprocal Hit-Rank (ARHR)
  - weighted version of Recall
  - the weight is the reciprocal of the position in the list

$$\mathbf{ARHR} = \frac{\sum_{i \in \mathbf{HIT}} \frac{1}{\mathbf{rank}(i)}}{\mathbf{\#\ tested\ relevant\ items}}$$

- rank(i) = position in the list of relevant item i
  - rank(i) = 1 → first in the list

Karypis et al., Item-Based Top-N Recommendation Algorithms, ACM TIIS 2004

Ranking items: ranking metrics

- Average Relative Position (ARP)
  - percentile-ranking of an item, averaged over all tested items

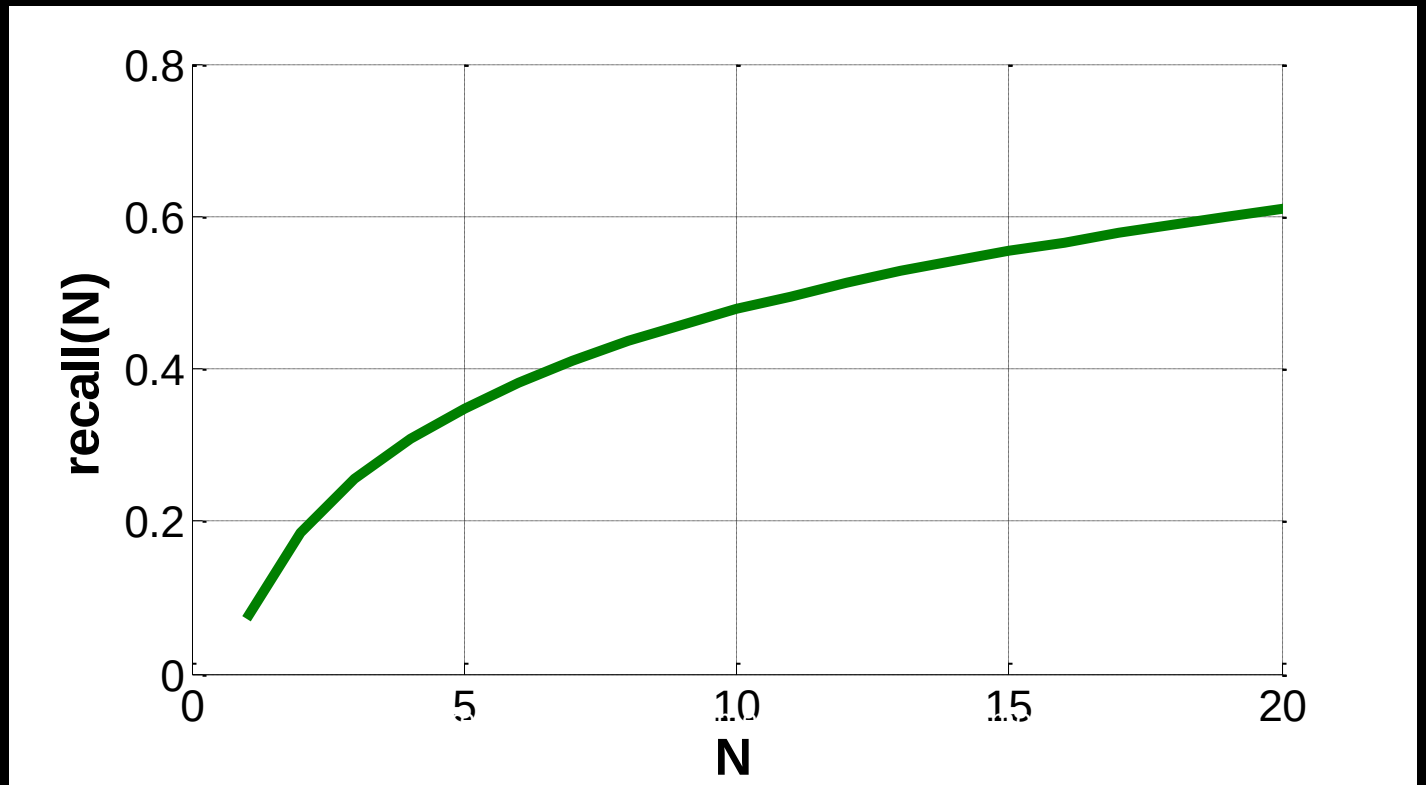
$$\mathbf{ARP} = \frac{\sum_{i \in \mathbf{HIT}} \frac{\mathbf{rank}(i)}{N}}{\#\ \mathbf{tested\ relevant\ items}}$$

  - N = length of the recommended list

Koren et al. Collaborative Filtering for Implicit Feedback Datasets. InICDM '08

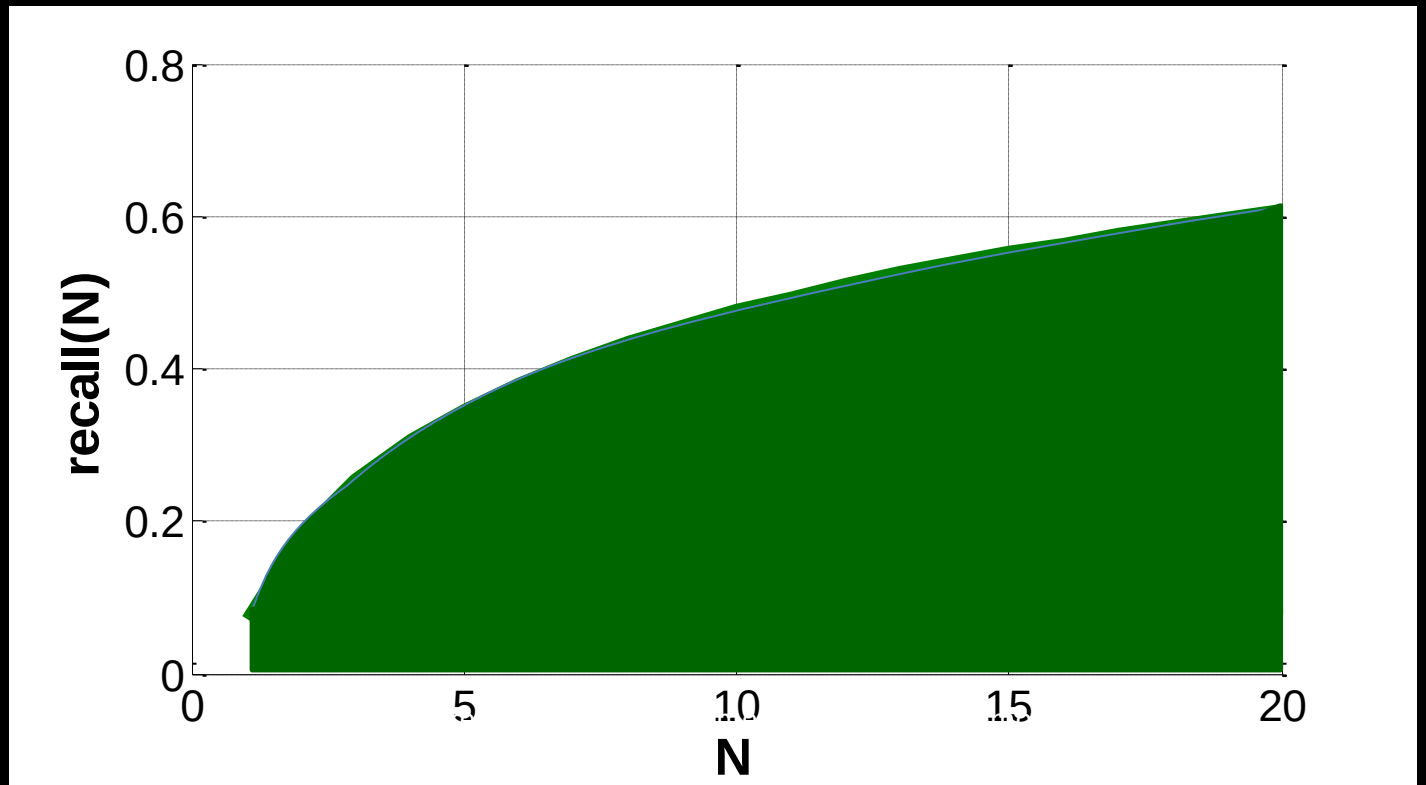
Ranking items: ranking metrics

- Average Relative Position (ARP)



Ranking items: ranking metrics

- Average Relative Position (ARP)
- ARP = area under recall / N = ATOP



Ranking items: ranking metrics

- if # relevant << # non-relevant
  - ARP = ATOP ≈ AUC with AMAN
- (we assume complete knowledge)
- In other words, recall is sufficient, we do not need fall-out
  - if we trust the AMAN hypothesis

Harald Steck, Training and testing of recommender systems on data missing not at random. In KDD '10

Ranking items: ranking metrics

- Normalized Cumulative Discounted Gain (NDCG)

$$\mathbf{CDG} = \frac{\sum_{i \in \mathrm{HIT}} \frac{1}{\max\left(1, \log_2\left(\mathrm{rank}(i)\right)\right)}}{\#\,\text{tested relevant items}}$$

$$\mathbf{NCDG} = \frac{\mathbf{CDG}}{\mathbf{CDG}_{\mathbf{IDEAL}}}$$

Ranking items: ranking metrics

- Spearman's Rho

$$\rho = \frac{1}{\#\,\mathbf{users}} \frac{\sum_i \left(r_{ui} - \bar{r}\right)\left(r'_{ui} - \bar{r}'\right)}{\mathrm{std}\left(r_{ui}\right)\mathrm{std}\left(r_{ui}'\right)}$$

- r_{ui} = true rank of item *i* for user *u*
- r'_{ui} = estimated rank of item *i* for user *u*

- Does not handle well partial (weak) orderings

Ranking items: ranking metrics

- Kendall's Tau

$$\tau = \frac{C - D}{\sqrt{(C + D + I)(C + D + I')}}$$

- *C* = number of concordant pairs
- *D* = number of discordant pairs
- *I* = # of pairs with the same rank
- *I'* = # of pairs estimated with the same rank
- An interchange between 1 and 2 is bad as an interchange between 1000 and 1001

Confidence

- Measure how much the recommender system trusts the accuracy of its recommendations
- Not to be confused with user-perceived trust

Confidence

- Strongly algorithm-dependent
  - support
    - number of ratings used to make a prediction
  - variance
  - probability of a recommendation
    - probability for an item to be relevant
- Techniques algorithm-independent
  - Bootstrapping
  - Noise injection

Coverage

- Percentage of items that the recommender system is able to provide predictions for
- Prediction coverage (potential)
  - percentage of items that **can be** recommended to users
  - A design property of the algorithm (e.g., CF cannot recommend items with no ratings, CBF cannot recommend items with no metadata)
- Catalogue coverage (de-facto)
  - percentage of items that **are ever** recommended to users
  - Measured by taking the union of all the recommended items for each tested user
- **Coverage can be increased at a cost of reducing accuracy**

Diversity

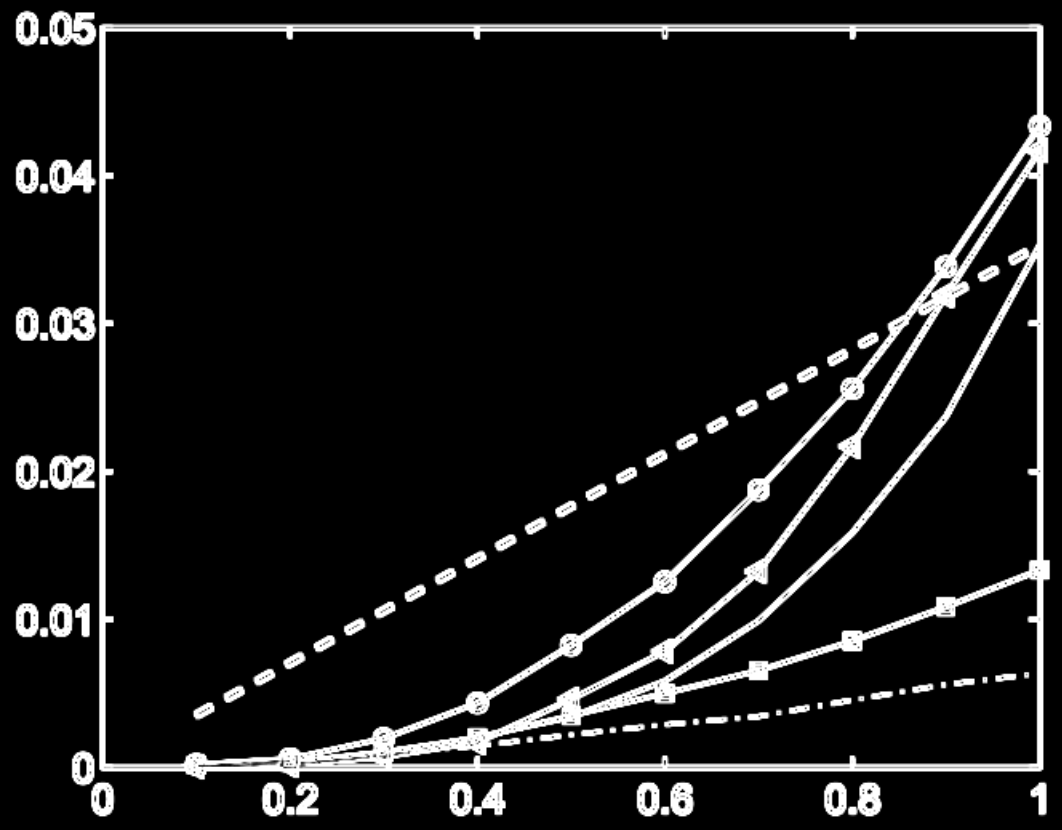
- Dissimilarity of items within the same list

$$\mathbf{diversity} = \frac{\sum_{i,j} \mathbf{distance}(i, j)}{N(N-1)}$$

  - This (dis)similarity is not necessarily the same used by the algorithms
- Most algorithms are based on similarity
  - **Diversity of the list can be increased at a cost of reducing accuracy**

Dissimilarity (diversity)

precision

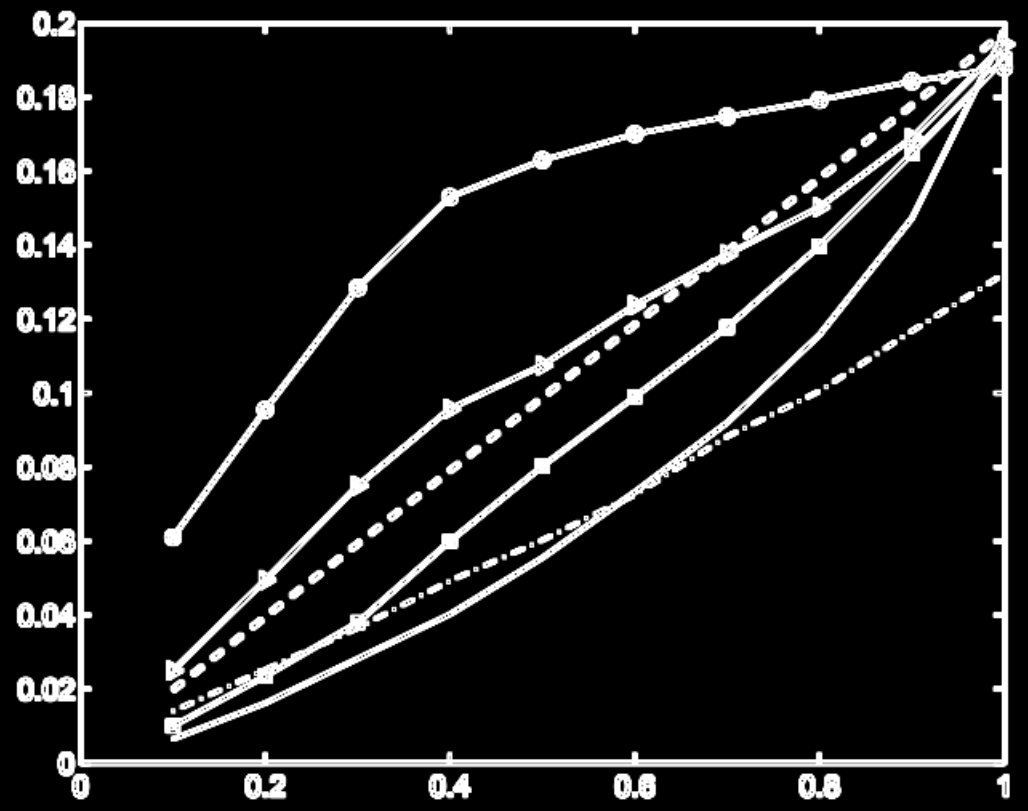


more diverse less diverse

Neil Hurley, Mi Zhang.
Novelty and Diversity in Top-N Recommendation - Analysis and Evaluation.
ACM Trans. Internet Technol. 2011

Dissimilarity (diversity)

precision



more diverse **less diverse**

Neil Hurley, Mi Zhang.
Novelty and Diversity in Top-N Recommendation - Analysis and Evaluation.
ACM Trans. Internet Technol. 2011

Diversity

- Temporal diversity
  - diversity of top-N recommendation lists over time
  - measures the extent that the same items are NOT recommended to the same user over and over again

$$\textbf{diversity} = \frac{L_{\text{NOW}} / L_{\text{PREVIOUS}}}{N}$$

- L_{NOW} = list of recommended items
- $L_{PREVIOUS}$ = list of items recommended in the past
- Temporal diversity is in contrast with consistency

Temporal Diversity in Recommender Systems, SIGIR'10

Novelty

- Measures the extent to which recommendations can be perceived as “new” by the users

$$\text{novelty} = \frac{\text{\# relevant and unknown recommended items}}{\text{\# recommended relevant items}}$$

- Novelty is difficult to measure

Novelty

- Two approximation
- **Distance based**
  - Novelty ≈ Diversity
  - Diverse recommendation increase the probability to have novel recommendations

Saúl Vargas, Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In RecSys '11

Novelty

- Two approximation
- **Discovery based**
  - Novelty ≈ 1/Popularity
  - Unpopularity is equivalent to Novelty applied to the whole set if users

$$\mathbf{novelty} = \frac{\sum_{i \in \mathbf{hits}} \log_2\left(1/\mathbf{popularity}(i)\right)}{\#\,\mathbf{hits}}$$

  - Popularity(i) = % users who rated item i

Serendipity

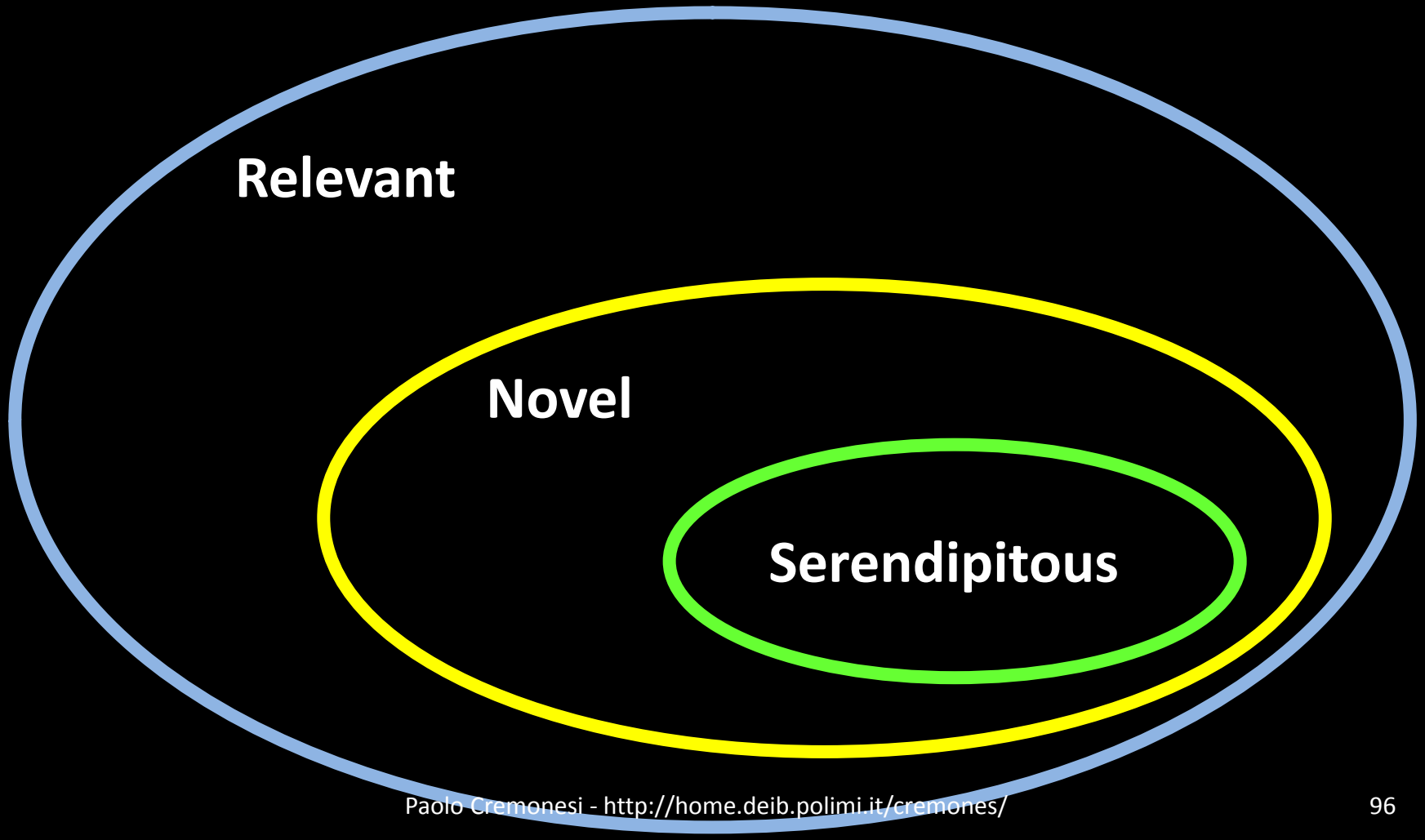
- **Serendipity** = unexpected recommendations, surprisingly and interesting items a user might not have otherwise discovered

- **Unexpected recommendations** = #recommendations from tested algorithm - #recommendations from "obvious" algorithm

$$\text{serendipity} = \frac{\text{\# unexpected relevant recommendations}}{\text{\# relevant recommendations}}$$

Serendipity

Serendipity → Novelty → Relevance

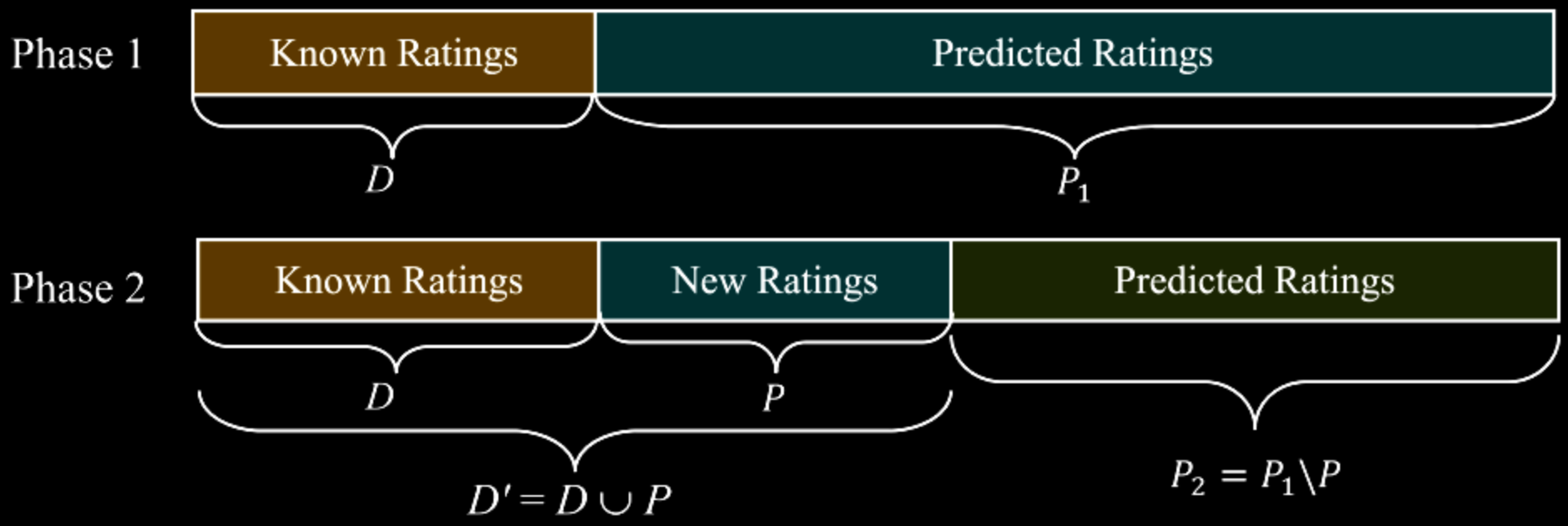


Stability and Consistency

- **Stability**: degree of change in predicted ratings when the user profile is extended with new estimated ratings

- **Consistency**: degree of change in the recommended list when the user profile is extended with new recommended items

Stability and Consistency



G.Adomavicius , J. Zhang. Stability of Recommendation Algorithms. *ACM Trans. Inf. Syst.* 2012

P Cremonesi, R Turrin
Controlling Consistency in Top-N Recommender Systems, ICDMW'10

Stability and Consistency

- Stability:
  - mean absolute shift (MAS) or root mean squared shift (RMSS)
  - MAE or RMSE computed between two different estimates of the ratings

- Consistency:
  - Consistency = Temporal Diversity

Thanks