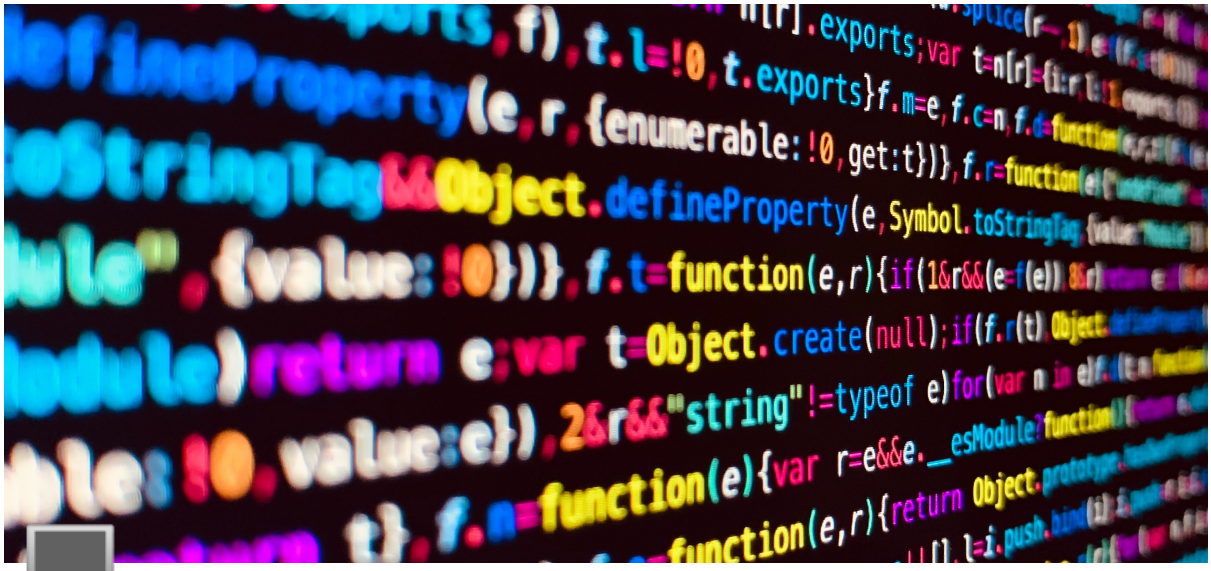


Recommender Systems

- Lecture 1 - Introduction, non-personalized, stereotype based
  - Non-personalized
  - Case-based reasoning
  - Frequency set
    - Association rules
  - Stereotype-based recommendations
- Lecture 2 - Classical recommender systems algorithms I
  - Content-based Recommender Systems
    - Vector Space Model (VSM)
    - Similarity-based retrieval using the VSM
    - Probabilistic methods
    - Text classification strategies
    - Linear Classifier
    - Decision Models
    - Features Selection
    - Limitations of Content-based RS
  - Ontologies
  - Folksonomies
- Lecture 3 - Classical recommender systems algorithms 2
  - Collaborative filtering
    - User-based neirest-neighbor collaborative filtering
    - Item-based collaborative filtering
    - Memory-based and model-based approaches
    - Alternative methods

        - Metrics
        - Collaborative Filtering Issues
    - Knowledge-based recommendations
        - Constraint-based
        - Case-based
- Lecture 4 - Evaluation of recommender systems
    - Evaluating Recommender Systems
    - Offline Evaluation
        - Methodology
        - Metrics
    - Online evaluation
    - Quasi-experimental Evaluation
- Lecture 5 - Interfaces, explanations and conversational recommender systems
    - Interfaces
    - Natural Language Generation (NLG)
    - Conversational recommender systems
- Lecture 6 - Group recommender systems
    - Group recommender systems
        - Recommending sequences to groups
    - Evaluating group recommenders
    - Explaining group recommendations
        - Privacy concerns with Group Explanations
- Lecture 7 - Hybrid recommender systems
    - Evaluating Explanations
    - Hybrid Recommender Systems
    - Hybridization Designs
        - Monolithic hybridization
        - Parallelized hybridization
        - Pipelined hybridization
- Lecture 8 - Context aware recommendations
    - Dimensions of context
        - Explicit vs latent
        - Static vs dynamic
        - Complete vs incomplete
    - Algorithmic paradigms
        - **Contextual pre-filtering**
        - **Contextual post-filtering**
        - **Contextual modelling**
        - Conclusion
- Lecture 9 - Ethics, bias and fairness in recommender systems
    - Fairness basic concepts
        - Individual vs group fairness

Distributional vs representational harms
Anti-classification vs anti-subordination
Stakeholder fairness
Studying fairness
Defining fairness
Data collection
Experiments design and results reporting
Consumer and provider fairness
Other stakeholders
Fairness mitigation
Lecture 10 - Person to person and multi-stakeholder recommendations
Person to person recommendation
Multi-stakeholder recommendations

Lecture 1 - Introduction, non-personalized, stereotype based

Benefits of recommendation: Avoid information overload, searching can be difficult, tells what's good/what's not.

> The core focus of most recommender systems is devoted to profiling users and matching items based on these profiles and current context.

Example of Recommender Systems

Recommender systems reduce information overload by estimating relevance.

- **Personalized recommendations:** user profile & contextual parameters
- **Collaborative:** tell me what's popular among my peers (community data)
- **Content-based:** show me more of the same what I've liked (product features)
- **Knowledge-based:** tell me what fits based on my needs (knowledge models)
- **Hybrid:** combinations of various inputs and/or composition of different mechanism

Non-personalized

Order by: *price* (often a bad idea), *recency* (domain specific), *popularity* (viewed often, liked on social media, bought often).

- **Memory-based** means that the recommender accesses the log data in real time
- **Model-based** signifies that the algorithm aggregates the data beforehand to make it more responsive

Experience shows that *memory-based* algorithms only work up a certain point because they do not require many views per minute before it becomes difficult for the servers to keep up

Seeded recommendation: Similar to a search query. Can be an item, a product, or an article. Frequently Bought Together category (affinity analysis or, in more familiar terms, shopping basket analysis).

Case-based reasoning

- Compare to previous similar cases
- Key in the notion of similarity

Frequency set

Association rules

Itemsets:

1. {bread, yoghurt}
2. {milk, bread, carrots}
3. {bread, carrots}
4. {bread, milk}
5. {milk, chocolate, carrots}
6. {milk, chocolate, yoghurt, bread}

Support: represents the popularity of that product of all the product transactions

$$S(X->Y) = (T(XandY)/T())$$

- T(), all transactions = 6
- S(bread → milk) = 3/6
- S(chocolate → carrots) = 1/6

- S(chocolate → milk) = 2/6

Confidence: can be interpreted as the likelihood of purchasing both the products A and B

$$c(X->Y) = (T(XandY)/T(X))$$

- c(chocolate → milk) = 2/2 = 1

Stereotype-based recommendations

- people use to build models of other people using stereotypes, or clusters of characteristics
- judge, he or she is probably - over forty, well -educated, reasonably pro - establishment, fairly affluent, honest, and well -respected in the community
- Computationally:
  - A set of facets, e.g., age, education
  - each of which contains a value (or values), e.g., over 40
  - Rating or confidence for the value e.g., 800/1000

Lecture 2 - Classical recommender systems algorithms I

Content-based Recommender Systems

Summary: A user is likely to have the same opinion for similar items. System learns importance of item features, so builds a model of what user likes.

Idea: A user is like to have similar opinion on similar items

Approach: use the content information of the items liked in the past to predict items that can be of interest for the user

The task then consists of determining the items that match the user's preferences best.

Content of the item - features: assume that characteristics of the items can be automatically extracted from the content or from unstructured textual descriptions

Typically, a content-based system evaluate how similar a not seen item is to items the user has liked in the past. For instance, the system could just check if the genre

of a new book is in the list genres of the user's preference profile. Another option is to compute a similarity evaluating the overlap between the keywords of the new book and the keyword in the user's preference profile:

Dice Coefficient

$$\frac{2 \cdot |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

Assuming that keywords(b) is the set of keywords associated to a book B. Several different similarity measures can be used.

Problem: Need to know about item content (requires manual or automatic indexing). Item features do not capture everything. *"User cold-start"* problem (needs to learn what content features are important for the user, so takes time. What if user's interest change? Lack of *serendipity* ("the effect by which one accidentally discovers something fortunate, especially while looking for something entirely unrelated")

Vector Space Model (VSM)

TF-IDF Model: encodes a textual document in a multidimensional Euclidian space. The space dimensions correspond to the keywords appearing in all the document. It combines two measures : Term frequency and Inverse document frequency

▼ Term Frequency

Describes how often a certain keyword i appears in a document j

$$TF(i,j) = \frac{freq(i,j)}{maxOthers(i,j)}$$

- freq(i, j) is the number of occurrences of the keyword i in the documend j
- maxOthers(i, j) = max(freq(z, j)), for each keyword z in the document j different from i

▼ Inverse Document Frequency

Is combined with term frequency to reduce the weight of keywords that appear very often in all documents

$$IDF(j) = log\frac{N}{n(j)}$$

- N is the number of all documents
- n(j) is the number of documents in which the keyword j appears

The combined TF-IDF weight for a keyword *i* and a document j is computed by multiplying the two terms

$$TF - IDF(i,j) = TF(i.j) * IDF(j)$$

TF-IDF vectors are usually very large and sparse.

▼ **Techniques to improve the model**

- Removing stop-words
  - words that will appear in every document
  - For instance "a", "the" or "on"
  - Such words are not informative and only increase the dimension of the vector
- Stemming
  - Replace variants of the same term by their common stem (root word)
  - usually performed combining morphological analysis and dictionaries (as WordNet)
- Size cut-off
  - Remove noise form the model only using the N most informative words
  - If N is too small important documents might not be covered while if N is too large it adds noise to the model
  - Features selection strategies can be applied for determining the most informative keywords
- Phrases detection
  - Sequences of words can be more informative than single words("United States of America") → in such case we use "phrases" (sequences of words) as single keywords

The vector space model presented may not capture the meaning correctly (because not having the context of the keyword)

Similarity-based retrieval using the VSM

The items are represented using the vector space model, computed from the textual content information associated to them. Each document is represented with a vector with a value for each feature.

▼ ***K Nearest Neighbors (KNN)***

Assumption: we have some history of feedbacks provided by the user about previously seen items

Idea: recommendation for a not-seen item D is made using the K most similar seen items

Can use the cosine similarity measuring how close two vectors are in their space (computes the cosine of the angle between them)

$$cos(\theta) = \frac{a \cdot b}{\|a\| * \|b\|}$$

where

$$a \cdot b = \sum_{i=1}^{N} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_N b_n \qquad \text{(dot product)}$$

$$\|a\| = \sqrt{a_1 + a_2 + \ldots + a_N} \qquad \text{(norm)}$$

Advantage: Simple to implement, able to quickly adapt to recent changes, able to provide a reasonable quality of the recommendations with a relatively small number of ratings

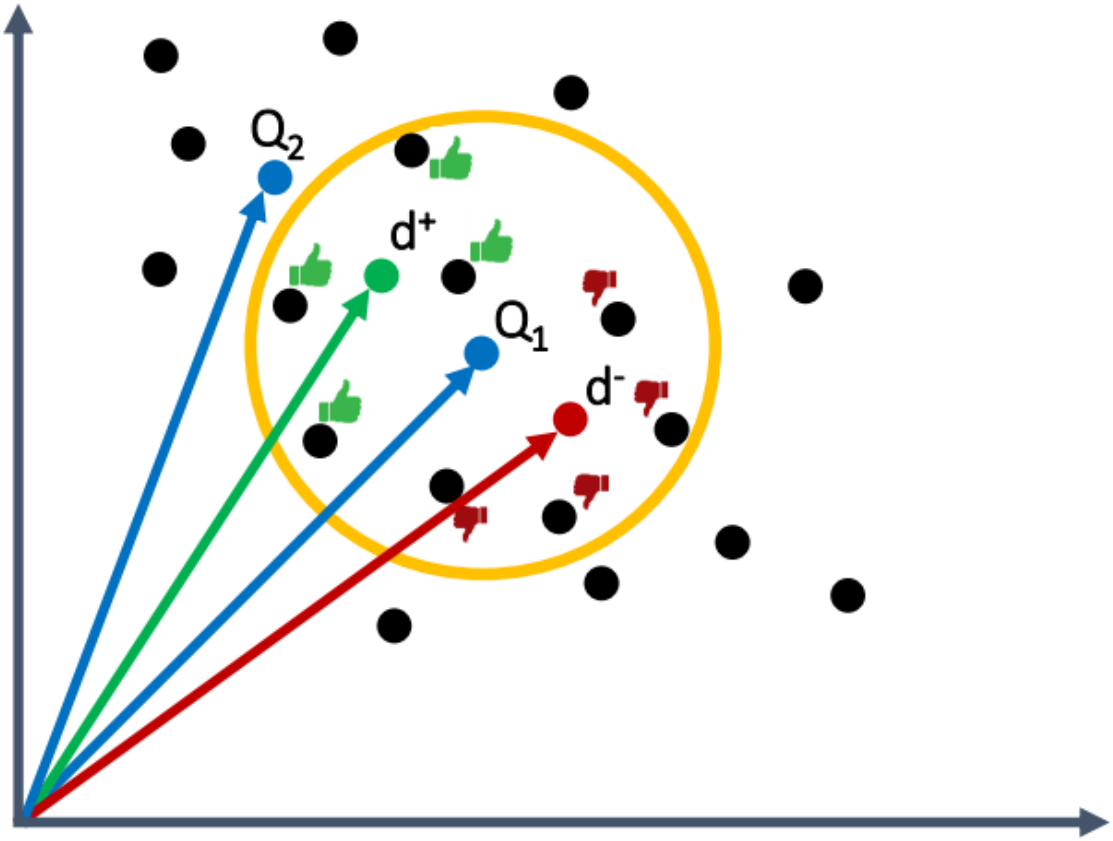
Inconvenient: prediction accuracy can be lower than those of other more sophisticated methods

▼ ***Rocchio's method***

Idea:

- Split the already rated items D in two sets D+ and D- of liked and unliked documents and calculate a prototype for these categories

- The current query Qi is represented as a vector in the vector space model of the documents
- The next query Qi+1 is refined by adding the prototype vector of the relevant documents and subtracting the prototype vector of the not relevant documents



Advantage: showed to improve the performances even after the first iteration. Simple and has been showed to lead to improvement in real-world settings

Inconvenient: certain number of ratings is necessary to build a reasonable user model. User interaction is required during the retrieval phase

Probabilistic methods

Based on the naïve Bayes assumption of conditional independence with respect to term occurrences. The probability of an unseen document to belong to a certain class c (like/dislike) is computed using the probability P(v | C=c) of each keyword *v* in it to belong to the given class. The probabilities associated to each unseen item can be then used to rank them and determine the recommendations.

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	1
2	0	0	1	1	0
3	1	1	0	0	1
4	1	0	1	1	1
5	0	0	0	1	0
6	1	1	0	0	?

$$
\begin{aligned}
P(X|Label = 1) = {} & P(recommender = 1|Label = 1) \times \\
& P(intelligent = 1|Label = 1) \times \\
& P(learning = 0|Label = 1) \times \\
& P(school = 0|Label = 1) \\
= {} & {}^{3}/_{3} \times {}^{2}/_{3} \times {}^{1}/_{3} \times {}^{2}/_{3} \approx 0.149
\end{aligned}
$$

Advantage: good results even if the naïve Bayes assumption does not hold. Model can be easily updated with new observations. Learning time is linear on the number of observations, while the predicting time is independent from it.

Inconvenient: A certain number of data is necessary to build the model.

Text classification strategies

The problem of deciding if an item will be liked by the user can be seen as a text classification problem (like/dislike).

Idea: Both the training documents and the unclassified documents are generated by the same probability distribution.

Different models have been proposed using different probability distributions:

- Multivariate Bernoulli Model: document is modelled using a binary vector describing if a keyword is present in it or not
- Multinomial Naive Bayes: frequency of the keyword in the document is also considered.

Empirical evaluations show that the Multinomial model leads to better performances, in particular when it comes to longer documents and a larger set of features.

Linear Classifier

In linear classifier, the problem is to find the coefficients of a linear model to discriminate between relevant and not relevant documents.

Both Rocchio method and Naïve Bayes classifier are specific linear classifiers, while the KNN is not. A very common linear classifier is based on Support Vector Machines (SVM), and try to identify decision boundaries that maximize the distance (margin) to the existing datapoints.

One challenge for such approaches is represented by “noise” in the training data.

Decision Models

Generating explicit decisions models in the training phase have also been applied for building content-based recommender systems.

Decision Trees construct a tree in which the inner nodes are labeled with the keywords.

Advantage: determining whether a new document is relevant can be done be done very efficiently with a prebuild classification tree (based on training data).

Inconvenient: Perform bad when the number of features is high, and reducing the number of features may lead to worst performances.

Features Selection

Vectors tend to be very long and sparse. This may lead to worst performances, computational issues and overfitting. Choose a subset of keywords based on domain knowledge, lexical information or frequency-based features selection.

Limitations of Content-based RS

- Keywords alone may not be sufficient to judge quality/relevance of a document or web page (limited content, writing style...)
- Overspecialization (algo tend to propose "more of the same")
- Ramp-up phase required (some training data is still required, use other sources to learn the user preferences)

Ontologies

Introduced as description of web resources (language that can be interpreted by software systems, use such description to better locate information on the web). Classification of web content that best matches the information need of the users (using a domain ontology that is exploited to describe web resources). Ontologies have been applied to improve filtering techniques in recommender systems (are actually hybrid systems).

Maidel et al. 2008 defined a recommender system for news using a taxonomy of concepts to compute the user-item similarity. The similarity is defined on five cases, considering the concepts *c* associated to an item (Considering if user profile's contains the concept *c* or a concept that is a parent/child/grandparent/grandchild of the concept c in the taxonomy).

Folksonomies

Folk taxonomies represent a more informal way of allowing users to annotate items with keywords. In contrast to ontologies, where the taxonomy must be precise and defined. Although less reliable, the folksonomy is easier to obtain.

Lecture 3 - Classical recommender systems algorithms 2

Collaborative filtering

Basic assumption and idea: users give ratings to catalog items (implicitly or explicitly). Customers who had similar tastes in the past, will have similar tastes in the future.

Input: only a matrix of given user-item ratings.

Output types: a (numerical) prediction indicating to what degree the current user will like or dislike a certain item. A top N-list of recommended items.

Advantage: No need to analyse contents. Can capture more subtle things. Serendipity.

User-based neirest-neighbor collaborative filtering

Idea:

- To predict a user's opinion for an item, use the opinion of similar users
- If users had similar tastes in the past they will have similar tastes in the future
- Similarity between users is decided by looking at their overlap in opinions for other items

Assumption: User preferences remain stable and consistent over time.

The basic technique: Given an "active user" (Alice) and an item *i* not yet seen by Alice

- find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past and who have rated item
- use, e.g. the average of their ratings to predict, if Alice will like item
- do this for all items Alice has not seen and recommend the best-rated

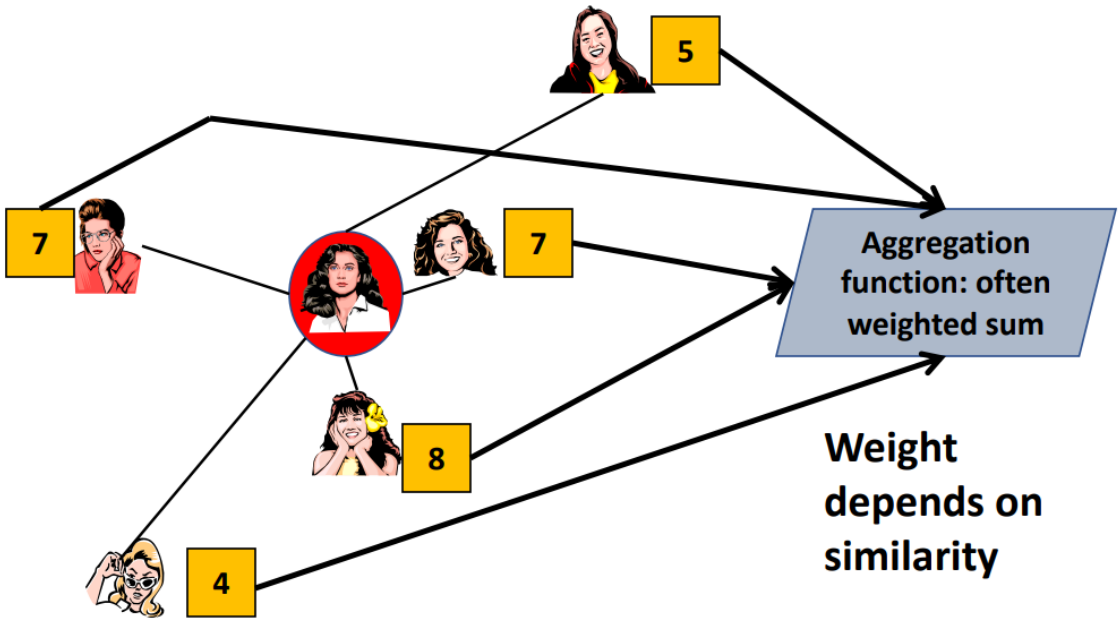
▼ Measuring user similarity (algorithm 1)

A popular similarity measure in user-based CF: ***Pearson correlation.*** Possible similarity values between -1 and 1.

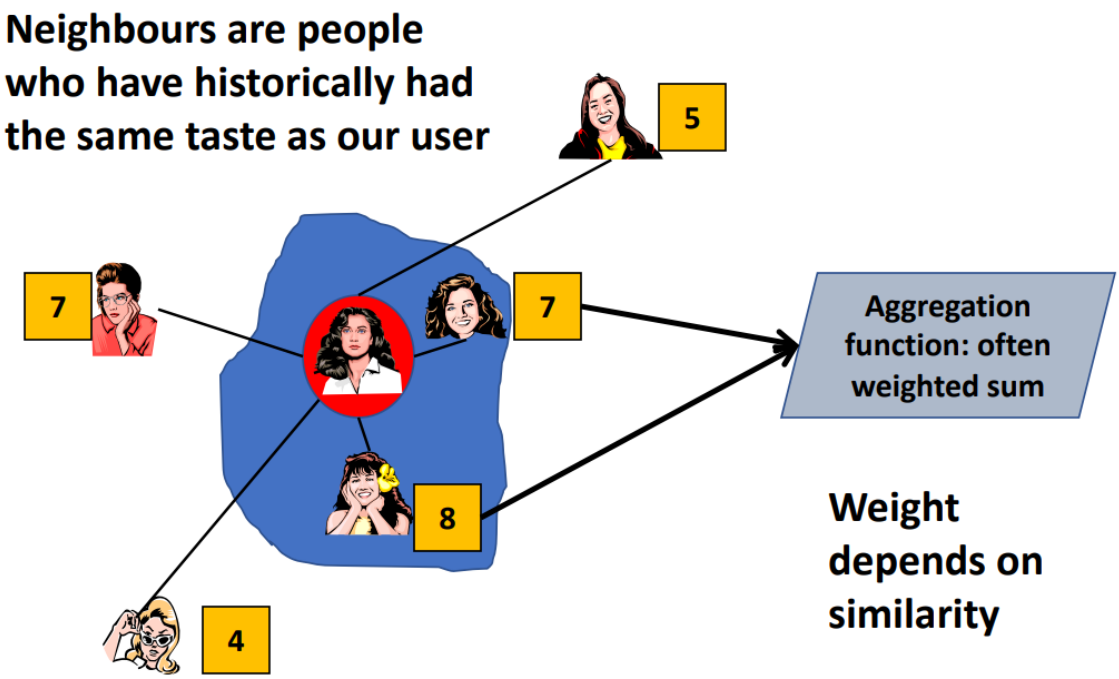
	Item1	Item2	Item3	Item4	Item5	
Alice	5	3	4	4	?	
User1	3	1	2	3	3	sim = 0,85
User2	4	3	4	3	5	sim = 0,00
User3	3	3	1	5	4	sim = 0,70
User4	1	5	5	2	1	sim = -0,79

▼ Making predictions

Calculate, whether the neighbors' ratings for the unseen items are higher or lower than their average. Combine the rating differences. Add/subtract the neighbors' bias from the active user's average and use this as a prediction.



▼ **K-nearest neighbour (algorithm 2)**



Problems:

- *user cold-start problem*: not enough known about new user to decide who is similar (and perhaps no other users yet..)
- *sparsity*: when recommending from a large item set, users will have rated only some of the items (makes it hard to find similar users).
- *scalability*: with millions of users and items, computations become slow
- *item cold-start problem*: cannot predict ratings for new item till more similar users have rated it

Item-based collaborative filtering

Idea: use the similarity between items (and not users) to make predictions

▼ **The cosine similarity measure**

Produces better results in item-to-item filtering. Ratings are seen as vector in n-dimensional space. Similarity is calculated between similarity based on the angle between the vectors.

- *Adjusted cosine similarity:* take average user ratings into account, transform the original ratings. Each user's ratings are vectors, with each item as a value. Dimensions = number of items. Similarity is how "close" vectors are in the n-dimensional space

Advantage: Prevents user cold-start problem, improves scalability (similarity between items is more stable than between users).

Problem: Item cold-start problem (cannot predict which items are similar till we have ratings for this item), which is not a problem for content based.

Memory-based and model-based approaches

User-based CF is said to be *"memory-based"*

- the rating matrix is directly used to find neighbors / make predictions
- does not scale for most real-world scenarios
- large e-commerce sites have tens of millions of customers and millions of items

Model-based approaches

- based on an offline pre-processing or "model-learning" phase
- at run-time, only the learned model is used to make predictions
- models are updated / re-trained periodically
- large variety of techniques used
- model-building and updating can be computationally expensive
- item-based CF is an example for model-based approaches (Calculate all pair-wise item similarities in advance. The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated. Item similarities are supposed to be more stable than user similarities)

Alternative methods

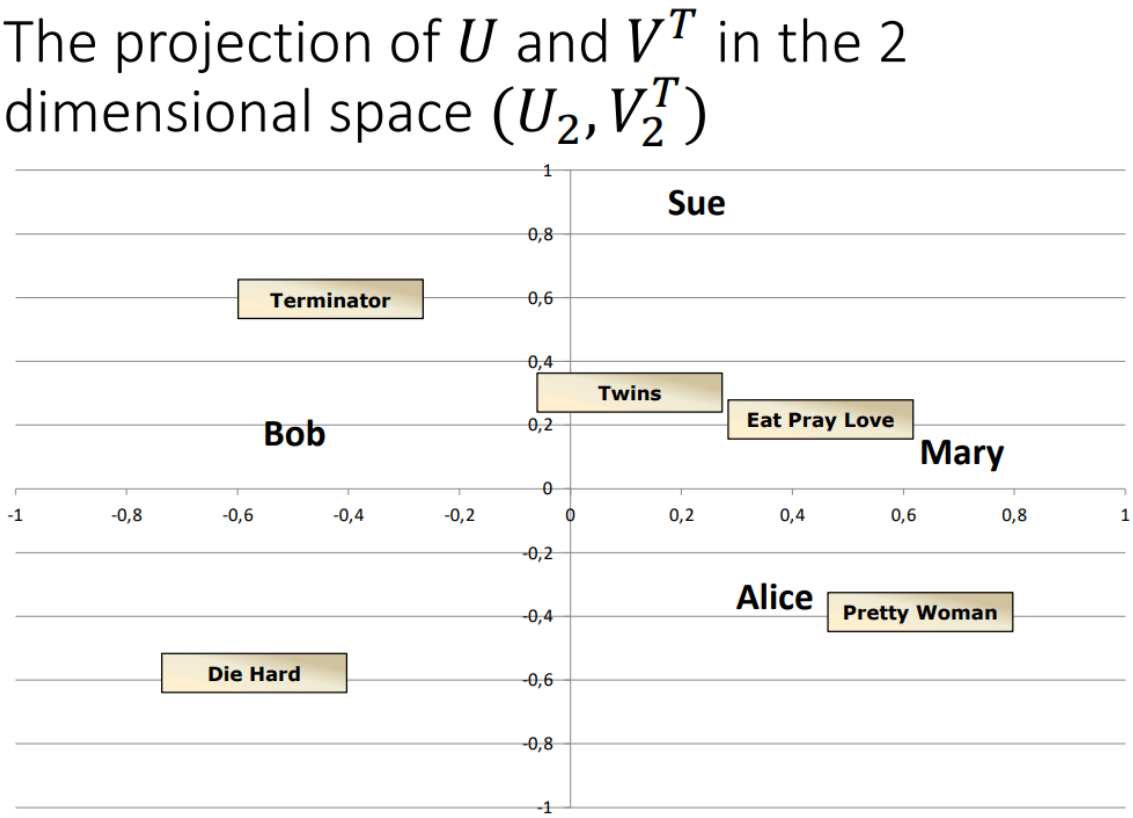
▼ **Data sparsity problems**

Solution to cold start problem: force users to rate a set of items, use another method in the initial phase, default voting (assign default values to items that only one of the two users to be compared has rated), use better algorithms (beyond nearest-neighbours approaches).

Recursive CF: apply CF-method recursively and predict a rating for item *i* for the neighbor that did not rate that item yet. Use this prediction rating instead of the rating of a more distant direct neighbor.

Singular Value Decomposition for dimensionality reduction of rating matrices (assume that k dimension capture the signals and filter out noise).

SVD-based recommendation: approximate by observing only the most important features



Dimensionality reduction:

Prediction quality can improve because...	Prediction quality can decrease because...
filtering out some "noise" in the data	the original ratings are not taken into account
detecting nontrivial correlations in the data	

▼ **Slope One predictors**

Idea: based on popularity differential between items for users

Example:

	Item1	Item5
Alice	2	?
User1	1	2

p(Alice, Item5) = 2 + (2 - 1) = 3

Basic scheme: take the average of these differences of the co-ratings to make the prediction

In general: Find a function of the form $f(x) = x + b$ (that is why the name is "Slope one")

More complex example:

	Item1	Item2	Item3
Alice	2	5	?
User1	3	2	5
User2	4	-	3

Item1/Item 3: 5-3=2; 3-4=-1 → avg 0.5 → 2+0.5
Item3/Item 2: 5-2= 3 → 5+3
pred(alice,Item3)=2*2.5 + 1*8/ 2+1

▼ **RF-Rec predictors**

Idea: Take rating frequencies into account for computing prediction

Metrics

Metrics measure error rate

- **Mean Absolute Error** (MAE) computes the deviation between predicted ratings and actual ratings

- **Root Mean Square Error** (RMSE) is similar to MAE, but places more emphasis on larger deviation

Collaborative Filtering Issues

Pros: well-understood, works well in some domains, no knowledge engineering required

Cons: requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results

How to evaluate the prediction quality? MAE / RMSE: What does an MAE of 0.7 actually mean? Serendipity (novelty and surprising effect of recommendations)

Knowledge-based recommendations

> "*Tell me what fits based on my needs*"

Why do we need knowledge-based recommendation?

- Products with low number of available ratings (=sparsity)
- Time span plays an important role (user lifestyle or family situation changes)
- customers want to define their requirements explicitly ("the color of the car should be black")

Both approaches (constraint-based and case-based) are similar in their conversational recommendation process.

1. users specify the requirements
2. systems try to identify solutions
3. if no solution can be found, users change requirements

Limitations: Cost of knowledge acquisition (from domain experts, users, web resources), accuracy of predefined models (very fine preference models require many interactions per cycles, collaborative filtering models preference implicitly), independence assumption can be challenged (preferences are not always independent from each other.

Constraint-based

Knowledge base: usually mediates between user model and item properties. Variables (user model features (requirements), item features (catalogue). Set of

constraints (logical implications, hard and soft/weighted constraints, solution preferences).

1. Find a set of user requirements such that a subset of items fulfills all constraints
    - ask user which requirements should be relaxed/modified such that some items exist that do not violate any constraint
2. Find a subset of items that satisfy the maximum set of weighted constraints
    - all proposed items have to fulfill the same set of constraints
    - compute relaxations based on predetermined weights
3. Rank items according to weights of satisfied soft constraints
    - rank items based on the ratio of fulfilled constraints
    - or based on criteria that interest the customer (price, quality, economy...)

Case-based

> Items are retrieved using similarity measures

expresses for each item attribute value its distance to the customer requirement according to a certain weight for each requirement.

In a real world, customers would like to

- maximize certain properties, i.e. peak watt, "more is better" (MIB)
- minimize certain properties, i.e. price, "less is better" (LIB)

Customers might not know what they are looking for. They can also specify their change requests that are not satisfied by the current item ("Like this but cheaper").

▼ **Dynamic critiques**

- Association rule mining
- Find compound critiques that are informative

For instance, user can click to improve results on buttons like "cheaper", "more quiet", "livelier"...

Lecture 4 - Evaluation of recommender systems

Evaluating Recommender Systems

Internal validity: observed effects due to the controlled test conditions instead of differences in the set of participants (aka predispositions)

External validity: results are generalizable to other user groups or situations, the scenario is representative of a real-world situation

Reliability: absence of inconsistence and errors in data and measurements

Sensibility: different evaluations of the observed aspects reflect in difference in the measurements

▼ **Measured variables**

Aspects of the users (or subjects) that are relevant in the context of the recommendation system and the research question examined (observed and measured factors)

- *Independent variables*: variables that are static throughout the course of the experiments
- *Dependent variables*: assumed to be influenced by the independent variables
- *Control conditions*: controlled by the evaluation designed (e.g. the recommendation algorithm)

▼ **Settings**

Lab studies (expressly created for the purpose of the study), field studies (real-world environment), offline studies (conducted on real-world data)

▼ **Design**

Experimental research: independent variables are manipulated, each participants experiences only one or all the system

Quasi-experimental: different from previous as no random assignments of subjects (subjects decide on their own)

Non-experimental:

- quantitative research (numerical measurements of different aspects of objects)
- qualitative research (interviews with open-ended questions)

- longitudinal research (entity under investigation is observed repeatedly over time)

cross-sectional: analyze relations among variables that are measured in different groups

case studies: focus on answering research question about how and why. Combine whichever types of quantitative and qualitative methods necessary to investigate contemporary phenomena in their real-life contexts

Offline Evaluation

Largely used in experimental evaluations for recommender systems.

Methodology

Usually, dataset is splitted in two ways:

1. Training set
2. Testing set (data used to evaluate the model)

▼ **Holdout**

A random set of ratings is withheld from user-item matrix, and it is used as test set (x%). The remaining ratings are used as training set (100-x%). Risk of overfitting (tested users are not totally unknown to the model). 20/80 holdout.

▼ **N-fold cross-validation**

Users in the matrix are divided into N partitions

- Normally 10, common split 8-2 or 9-1
- K-1 partitions are used for the training
- the remaining partition is used for the test
- average of several ways of splitting the data

▼ **Leave-P-out validation**

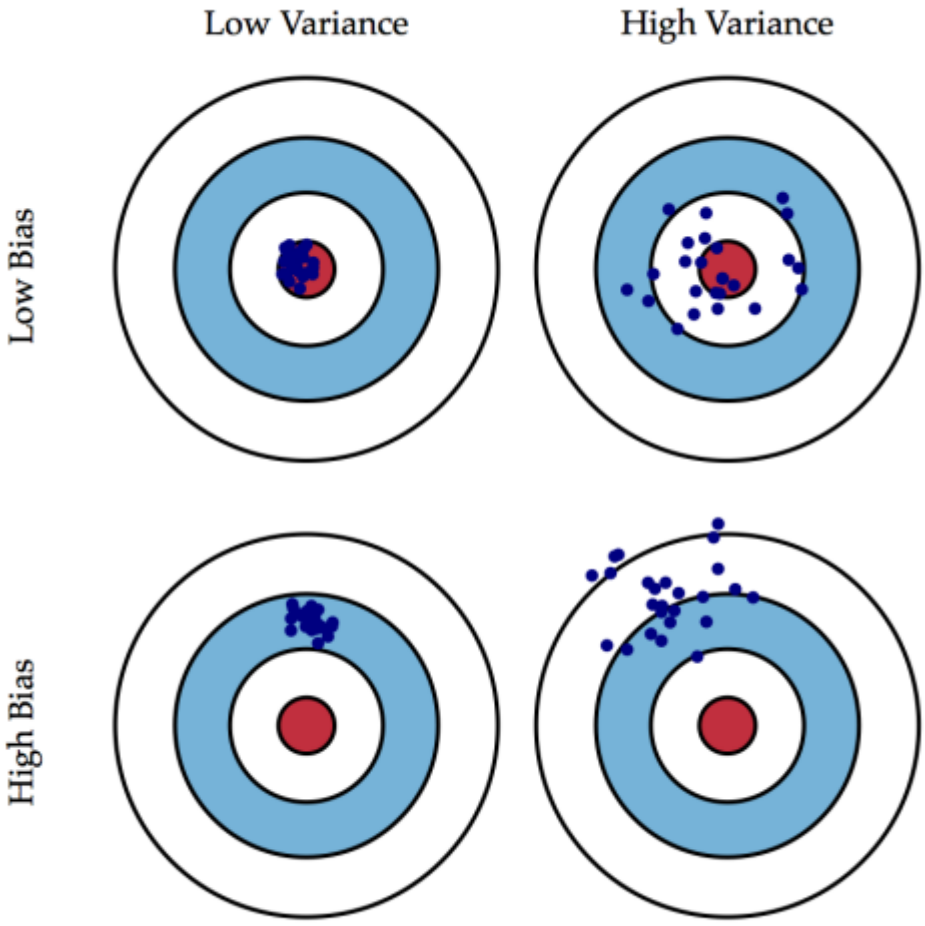
Usually leave-one-out. During testing, for each user, test one rated item at a time for the test set

- Leave-p-out: test predicted ratings for p items

▼ **Bootstrap**

Randomly select a sample of % of ratings using sampling with replacement.

- "Increases" sample size
- Less variance but more bias



Metrics

▼ **Accuracy of predictions**

Mean Absolute Error (MAE): computes the deviation between predicted ratings and actual ratings

Normalized Mean Absolute Error (NMAE): same as before but the returned value are between 0 and 1

Root Mean Square Error (RMSE): is similar to MAE, but places more emphasis on larger deviation

- **Mean Absolute Error (MAE)**
  - computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i|$$

- **Normalized Mean Absolute Error (NMAE)**
  - same as before but the returned values are between 0 and 1

$$NMAE = \frac{MAE}{r_{max} - r_{min}}$$

- **Root Mean Square Error (RMSE)**
  - is similar to MAE, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2}$$

▼ **Accuracy of classifications**

Precision: a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved (the proportion of recommended movies that are actually good)

$$Precision = (tp)/(tp + fp) = goodmoviesrecommended/allrecommendations$$

Recall: a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items (the proportion of all good movies recommended)

$$Recall = (tp)/(tp + fn) = goodmoviesrecommended/allgoodmovies$$

		Reality	
		Actually Good	Actually Bad
Prediction	Rated Good	True Positive (TP)	False Positive (FP)
	Rated Bad	False Negative (FN)	True Negative (TN)

Precision-recall curve: typically when a recommender system is tuned to increase precision, recall decreases as a result (or vice versa)

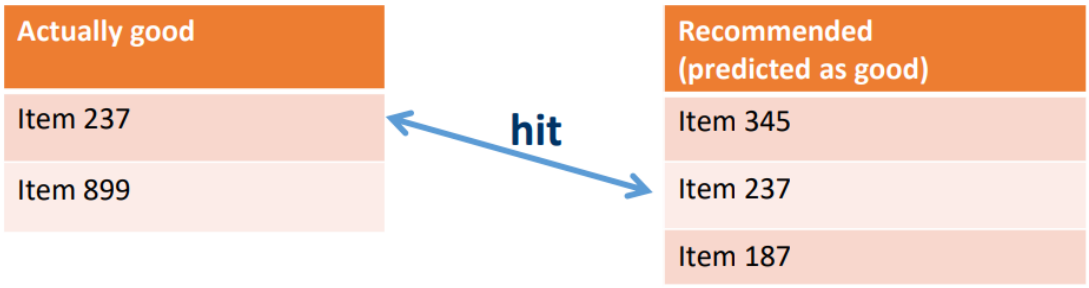
Area under the curve (AOC): performance measurement for the classification problems at various threshold settings.

F1 score: attempts to combine Precision and Recall into a single value for comparison purposes

▼ **Accuracy of ranks**

Rank metrics

- Take the positions of correct items in a ranked list into account
- Relevant items are move useful when they appear earlier in the recommended list
- particularly important in recommender systems as lower ranked items may be overlooked by users



Rank score

- extends the recall metric to take the positions of correct items in a ranked list into account.
- particularly important in recommender systems as lower ranked items may be overlooked by users.
- rank score is defined as the ratio of the Rank Score of the correct items to best theoretical Rank Score achievable for the user.

- **Rank Score** (example)

Rank	Hit?
1	
2	X
3	X
4	X
5	

- Assumptions:
  - |T| = 3
  - Ranking half life (alpha) = 2

$$rankscore_p = \sum_{i \in h} 2^{-\frac{rank(i)-1}{\alpha}} \qquad rankscore_p = \frac{1}{2^{\frac{2-1}{2}}} + \frac{1}{2^{\frac{3-1}{2}}} + \frac{1}{2^{\frac{4-1}{2}}} = 1.56$$

$$rankscore_{max} = \sum_{i=1}^{|T|} 2^{-\frac{i-1}{\alpha}} \qquad rankscore_{max} = \frac{1}{2^{\frac{1-1}{2}}} + \frac{1}{2^{\frac{2-1}{2}}} + \frac{1}{2^{\frac{3-1}{2}}} = 2.21$$

$$rankscore = \frac{rankscore_p}{rankscore_{max}} \approx 0.71$$

Average Precision (AP): is a ranked precision metric that places emphasis on highly ranked correct predictions. Essentially it is the average of precision values determined after each successful prediction.

Rank	Hit?
1	
2	X
3	X
4	X
5	

$$AP = \frac{1}{3}\left(\frac{1}{2} + \frac{2}{3} + \frac{3}{4}\right) = \frac{23}{36} \approx 0.639$$

Rank	Hit?
1	X
2	
3	
4	X
5	X

$$AP = \frac{1}{3}\left(\frac{1}{1} + \frac{2}{4} + \frac{3}{5}\right) = \frac{21}{30} = 0.7$$

use *ANOVA* to check if differences are due to chance

Online evaluation

Offline experimentation	Online experimentation
Ratings, transactions	Ratings, feedback
Historic session (not all recommended items are rated)	Live interaction (all recommended items are rated)
Ratings of unrated items unknown, but interpreted as “bad” (default assumption, user tend to rate only good items)	“Good/bad” ratings of not recommended items are unknown
If default assumption does not hold: True positives may be too small False negatives may be too small	False/true negatives cannot be determined
Precision may increase Recall may vary	Precision ok Recall questionable

Quasi-experimental Evaluation

Conversational recommender system (question and answer dialog, matching of user preferences with knowledge base).

Question of causality cannot be answered.

Lecture 5 - Interfaces, explanations and conversational recommender systems

> Understandability is not an end goal, effective decision-making is.

Interfaces

Intelligent: Containing some measure of reasoning (human or artificial). Can be classification, recommendation, natural language processing, planning etc.

Purpose	Description
Transparency	Explain how the system works
Effectiveness	Help users make good decisions
Trust	Increase users' confidence in the system
Persuasiveness	Convince users to try or buy
Satisfaction	Increase the ease of use or enjoyment
Scrutability	Allow users to tell the system it is wrong

Purpose	Description
Efficiency	Help users make decisions faster

Justification: explains why a decision is good one, without explaining exactly how it was made.

Interpretability: degree to which a human can understand the cause of a decision and degree to which a human can consistently predict the model's result.

Natural Language Generation (NLG)

Construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information.

Input: data (raw, analysed)

Output: documents, reports, explanations, help messages, and other kinds of texts.

Requires: Knowledge of language, Knowledge of the domain.

▼ **Gricean maxims**

- Maxim of Quantity: Be exactly as informative as is required
- Maxim of Quality: Try to make your contribution one that is true
- Maxim of Relevance: Be relevant
- Maxim of Manner: Be perspicuous (~clear)

Three steps:

1. Document planning (content determination): decide on content and structure of text
2. Microplanning: decide how to linguistically express text (which words, sentences, etc to use)
3. Realisation: actually produce text, conforming to rules of grammar (e.g., referring expressions)

Realiser: tell realiser verb, tense, whether negated, nd it will figure out the rest (e.g. (watch, future) → will watch

Challenges of conversational experiences: high expectations for human conversations, languages, accents, dialects, grammar, work well only in case of limited simple queries, low discoverability due to lack of GUI.

Key takeaway on good conversations:

- Conversations with people are highly social & transactional
- Conversations with agents are highly functional
- Social agent relationships through conversation fundamentally questioned
- Paradigm of agent conversation may need redefinition

▼ **Information seeking tasks**

Search: I am looking for that Italian restaurant on the Besteenmarkt.
Exploration: I have a visitor coming and want to take them to a nice restaurant.
Recommendation: – somewhere in-between. “You know what I like, find it for me please”.

▼ **Group recommendation**

Explanations based on aggregation strategy

	Strategy	Explanation template
Consensus-based	*ADD*	*"Item X has been recommended to the group since it achieves the highest total rating.* ***This decision supports the preferences of users u_a, u_b, and u_c who were treated less favorably in the last n decisions."***
	FAI	***"The preference of user u_a was not considered in the last n decisions. Therefore, in this decision, item X has been recommended to the group since he/she likes it the most."***
Majority-based	*APP*	*"Item X has been recommended to the group since it achieves the highest number of ratings which are above a threshold θ.* ***This decision supports the preferences of users u_a, u_b, and u_c who were treated less favorably in the last n decisions."***
Borderline	*LMS*	*"Item X has been recommended to the group since no group member has a real problem with it.* ***This decision supports the preferences of users u_a and u_b who were treated less favorably in the last n decisions."***
	MAJ	*"Item X has been recommended to the group since most group members like it.* ***This decision supports the preferences of users u_a, u_b, and u_c who were treated less favorably in the last n decisions."***
	MPL	*"Item X has been recommended to the group since it achieves the highest of all individual group members' ratings.* ***This decision supports the preference of user u_a who was treated less favorably in the last n decisions."***

▼ **Context**

Location, time of day, activity, who else is there, mood, previous interactions

Conversational recommender systems

Traditionally, multi-turn interaction. Increasingly more, using natural language, or controlled language.

Lecture 6 - Group recommender systems

Group recommender systems

▼ Plurality Voting (majority based)

Each group member votes for his or her most preferred alternative. The alternative with the most votes wins. Normally, each voter has one choice (this might easily lead to ties). In this example, each user votes for the best items for him/her:

	A	B	C	D	E	F	G	H	I	J
Alice	10	4	3	6	10	9	6	8	10	8
Bob	1	9	8	9	7	9	6	9	3	8
Carl	10	5	2	7	9	8	5	6	7	6

Preference list: A, E, F, (D, I), H, J, (B, G), C

Finally, the last chosen item is C

▼ Approval Voting (majority based)

Voters are allowed to vote for as many alternatives as they wish. This is intended to promote the election of moderate alternatives. We assume that the users vote for the items with a rating above a threshold of θ=5.

	A	B	C	D	E	F	G	H	I	J
Alice	10	4	3	6	10	9	6	8	10	8
Bob	1	9	8	9	7	9	6	9	3	8
Carl	10	5	2	7	9	8	5	6	7	6
Group	2	1	1	3	3	3	2	3	2	3

Preference list: (D, E, F, F, J), (A,G, I), (B, C)

▼ Additive/Average (consensus based)

Ratings are added, and the larger the sum the earlier the alternative appears in the sequence. Note that the resulting preference list is equivalent if we use the average.

	A	B	C	D	E	F	G	H	I	J
Alice	10	4	3	6	10	9	6	8	10	8
Bob	1	9	8	9	7	9	6	9	3	8
Carl	10	5	2	7	9	8	5	6	7	6
Group	21	18	13	22	26	26	17	23	20	22

Preference list: (E, F), H, (D, J), A, I, B, G, C

▼ **Multiplicative (consensus based)**

Ratings are multiplied, and the larger the result the earlier the alternative appears in the sequence

▼ **Fairness (consensus based)**

Items are ordered as the group members choose them in turn. The order in which they choose is crucial in this case. Let's assume the order is Carl, Bob, Alice (changing the order would lead to different results)

	A	B	C	D	E	F	G	H	I	J
Alice	10	4	3	6	10	9	6	8	10	8
Bob	1	9	8	9	7	9	6	9	3	8
Carl	10	5	2	7	9	8	5	6	7	6

Preference list: A, (B, D, F, H), (E, I), C, G

▼ **Dictatorship (borderline)**

One group member decides

▼ **Least Misery (borderline)**

The group rating is the minimum of the individual ratings. Items get selected based on such ratings, the higher the sooner.

	A	B	C	D	E	F	G	H	I	J
Alice	10	4	3	6	10	9	6	8	10	8
Bob	1	9	8	9	7	9	6	9	3	8
Carl	10	5	2	7	9	8	5	6	7	6
Group	**1**	**4**	**2**	**6**	**7**	**8**	**5**	**6**	**3**	**6**

Preference list: F, E, (D, H, J), G, B, I, C, A

▼ **Most Pleasure (borderline)**

The group rating is the maximum of the individual ratings. Items get selected based on such ratings, the higher the sooner.

	A	B	C	D	E	F	G	H	I	J
Alice	10	4	3	6	10	9	6	8	10	8
Bob	1	9	8	9	7	9	6	9	3	8
Carl	10	5	2	7	9	8	5	6	7	6
Group	**10**	**9**	**8**	**9**	**10**	**9**	**6**	**9**	**10**	**8**

Preference list: (A, E, I), (B, D, F, H), (C, J), G

Sometimes other factors might influence the individual's satisfaction:

- *Emotional contagion* (other group members being satisfied may increase a user's satisfaction. The opposite might happen)
- *conformity* (the opinion of others can influence your own): normative influence (the person knows that the others are wrong, but to be part of the group aligns his/her opinion to the others), informational influence (this person changes opinion believing that the groups must be right)

Incorporating group attributes: demographics (different type of people) , roles (parent/child), personality, expertise (based on the number of ratings provided by one user), relationship strength, relationship type

Recommending sequences to groups

Fairness balancing the relevance of the recommended items across the group members in a rank-sensitive way (each sub-sequence in the list should balance the preferences of the group members).

Greedy algorithm (GFAR) for finding a top-N set of recommendations satisfying the fairness definition.

Evaluating group recommenders

User studies: usually performed to evaluate user experience and system usability

Offline evaluations: difficult due to lack of datasets

Evaluations can suffer from popularity biases (if the individual RS tends to favor popular items, this will lead to overestimate performances for some aggregation strategies).

Explaining group recommendations

Explanations are useful to achieve goals as:

- Transparency (explaining how the RS works)
- Effectiveness (helping the user in making good decisions)
- Usability of the system
- User satisfaction

In the context of group RS, explanations can achieve further goals: fairness, consensus

Participants prefer short, simple, informal and friendly explanations. Adding social component in explanations increases users' likelihood to accept the recommendations.

Explanation types:

1. Type 1: explain the aggregation strategy
2. Type 2: adds information on the decision history
3. Type 3: adds information about the future decision plan

explanations related to the Additive (ADD) and Majority (MAJ) strategies help the most to increase the fairness and consensus perception, and satisfaction. Users’ perceived fairness or consensus correlates with their satisfaction.

Privacy concerns with Group Explanations

> Information that lead the system to the choice could be really personal (some people might be not OK in sharing it with the other group members)

The users should be able to select the degree of granularity of information to disclose.

Lecture 7 - Hybrid recommender systems

Evaluating Explanations

Possible criteria: effectiveness, efficiency, usability, *system specific criteria*

Task-performance evaluation: extrinsic evaluation, measure whether system achieves its communicative goal (typically helping user perform a task - better decision making, or other possibilities e.g. behaviour change)

- *Evaluate in real-world or in controlled experiment:* most respected (especially in the NLP community), but very expensive and time-consuming. Evaluation is of specific system, not generic algorithm or idea.

Human ratings evaluation: intrinsic evaluation, ask human to assess texts on Likert scale

- *Probably most common type in NLG* (but less accepted outside NLP)
- *easier/quicker than task-based:* for laboratory evaluation can sometimes use crowd computing (e.g. Mechanical Turk - people doing tasks), can answer questions which are hard to fit into a task-based evaluation

▼ **Extrinsic vs intrinsic evaluation**

What is most meaningful?

	Task (extrinsic)	Ratings (intrinsic)
Real-world	**Most meaningful**	intermediate
Laboratory	intermediate	*Least meaningful*

Which is most costly?

	Task (extrinsic)	Ratings (intrinsic)
Real-world	**Most expensive**	intermediate

	Task (extrinsic)	Ratings (intrinsic)
Laboratory	intermediate	*Cheapest*

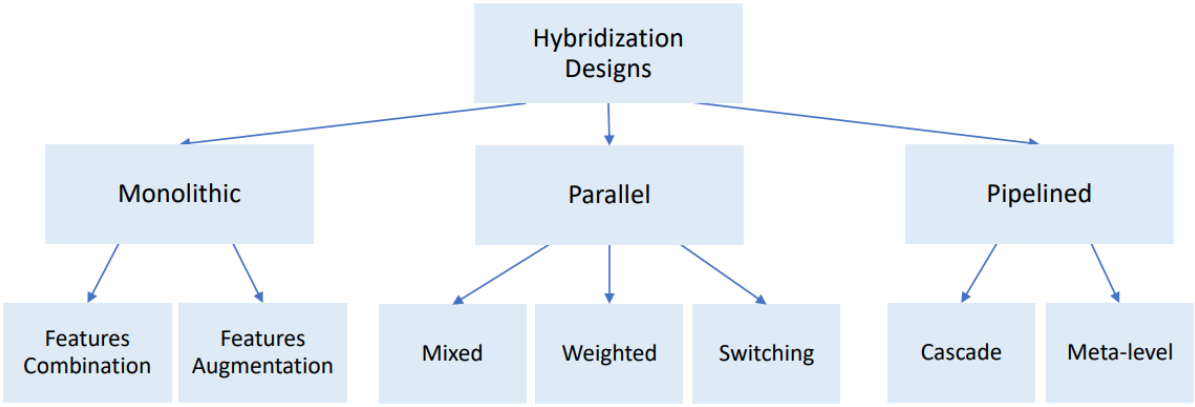
Metric-based evaluation: create a gold standard (input data for NLG system (scenarios), desired output text (usually human-written), run NLG system on above data sets, compare output to gold standard output. Widely used in machine translation, summarization...

- **Limited Confidence:** correlation of 0.8 (or more) with high-quality human evaluation? Clarity about scope, strength of validation evidence for metrics in other areas of NLP?

💡 **LAYERED EVALUATION SUMMARY**
Main premises: break adaption process down into its constituents ("layers"), evaluate each of them separately where necessary and feasible, meant to provide guidance rather than prescribing a certain way of evaluation.
Benefits: offers guidance for possible studies ("separation of concerns"), helps to identify problems and wrong assumptions, guides development process (formative evaluation)

Hybrid Recommender Systems

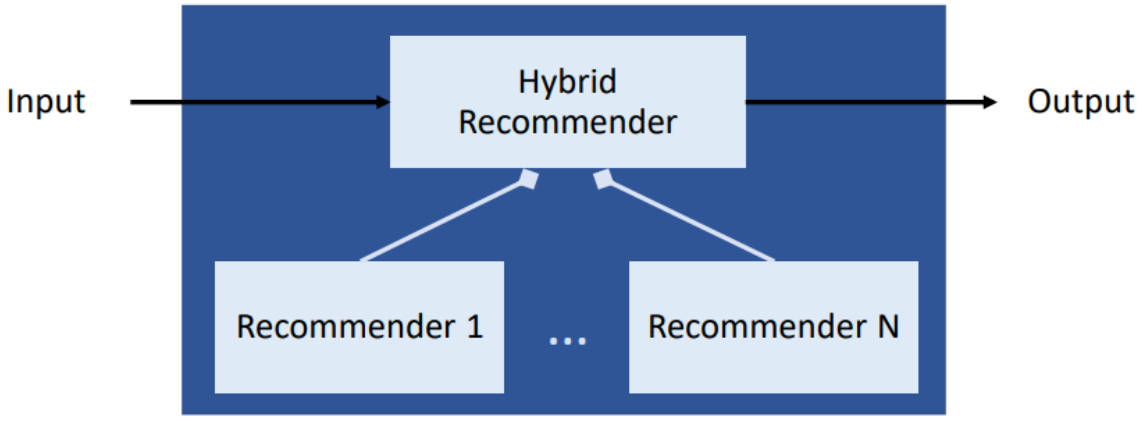
💡 **Monolithic:** hybridization design that incorporates aspects of several recommendation strategies in one algorithm implementation
Parallel: two separates RS operate independently and produce separate recommendations. In a subsequent step, their output is combined into a final set of recommendation
Pipelined: the output of one recommender becomes part of the input of the subsequent one



Hybridization Designs

Monolithic hybridization

A single recommender component that integrates multiple approaches (by preprocessing and combining several knowledge sources). Hybridization is thus achieved by a built-in modification of the algorithm behaviour to exploit different types of input data. Data-specific preprocessing steps are used to transform the input data into a representation that can be exploited by a specific algorithm paradigm.

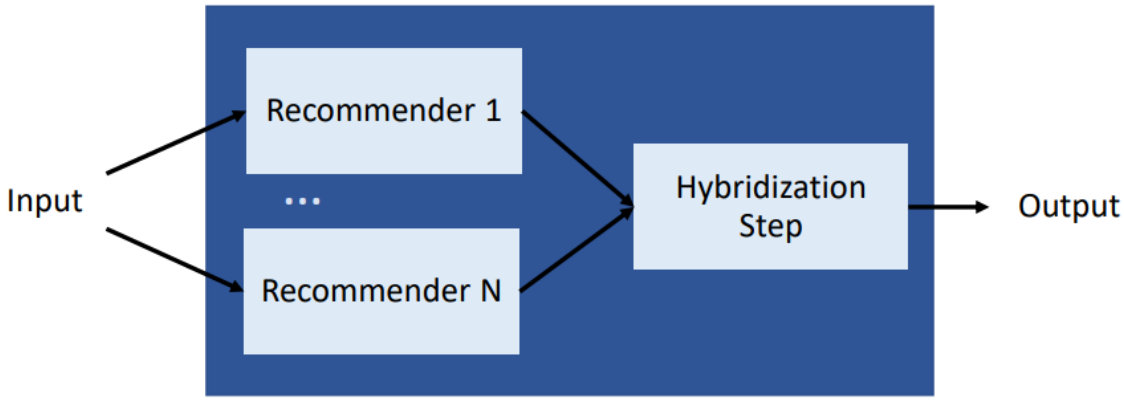


Feature combination: Monolithic recommendation component that uses a diverse range of input data. Combine features from different RS paradigms (e.g. CF + extra explicitly collected user feedbacks)

Features augmentation: Does not simply combine and preprocess several types of input, but rather applied more complex transformation steps. The output of a contributing RS augments the feature space of the actual recommender by preprocessing its knowledge sources (e.g. CF with content-based predictions)

Parallelized hybridization

We use several recommenders side by side. Then, a specific hybridization mechanism is used to aggregate the outputs.



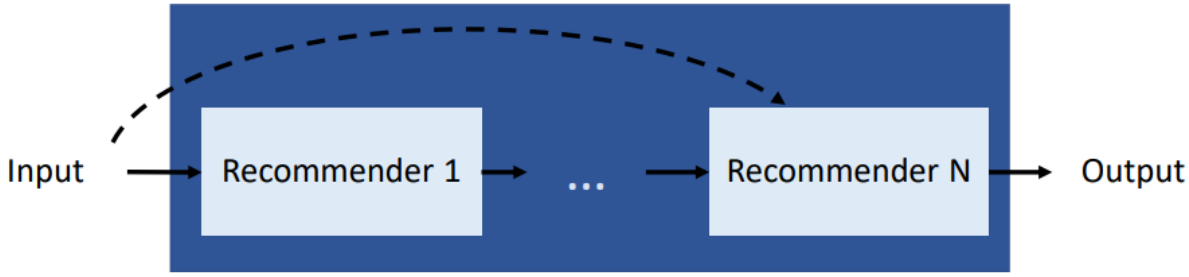
Mixed hybrids: combines the results of different RS at the level of the UI, in which results from different techniques are presented together. The top-scoring items for each recommender are then displayed next to each other.

Weighted hybrids: combines the recommendation of two or more RS by computing weighted sums of their scores. To evaluate the weights, an empirical bootstrapping approach can be applied (start from uniform and then adapt based on errors).

Switching hybrids: require an oracle that decides which recommender should be used in a specific situation. For instance, a KB system might be used until enough rating data are available (then we switch to CF).

Pipelined hybridization

Implement a staged process in which several techniques sequentially build on each other before the final one produces recommendations for the user. The pipelined hybrid variant differentiate themselves mainly according to the type of output they produce for the next stage (a preceding component may either preprocess input data to build a model that is exploited by the subsequent stage or deliver a recommendation list for further refinement).



Cascade hybrids: based on a sequenced order of techniques. Each succeeding recommender only refines the recommendations of its predecessor (all successor recommenders can only change the order or exclude some items).

Meta-level hybrids: one recommender builds a model that is exploited by the principal recommender to make recommendations (e.g. exploit a CF approach that builds on user models obtained through a CB recommender).

Lecture 8 - Context aware recommendations

💡 Conditions or circumstances which affect some thing

Dimensions of context

	Static	Dynamic
Explicit	Traditional approaches	E.g. whether to consider previous website
Latent (not directly observable)	Implicit context (e.g. mobile sensor data)	C.f. ongoing work in reinforcement learning

Explicit vs latent

Explicit: specified directly, e.g., through a questionnaire, through clicks

Latent: hidden/inferred. User behavior is affected by a context that is directly not observable)

Static vs dynamic

Static: predefined observable attributes. Stable, with (mostly) fixed structure or schema.

Dynamic: Different activities give rise to different contexts

Complete vs incomplete

Complete: all contextual factors are known explicitly (static, explicit)

Incomplete: only a subset of contextual factors are known (e.g. collected via user reviews)

Algorithmic paradigms

- Personalized models consistently outperform global models
- Individual preference models are more accurate models
- Users have different preferences for different conditions
- Computationally more expensive, but justified

Contextual pre-filtering

Contextual information drives data selection for that specific context. In other words, information about the current context *c* is used for selecting only the relevant set of data records. Then, ratings can be predicted using any traditional 2D recommender system on the selected data.

advantage: can use with existing techniques

example: if a person wants to see a movie on Saturday, only the Saturday rating data is used to recommend

Contextual post-filtering

The ratings are predicted using any traditional 2D recommender system on the entire data. Then, the resulting set of recommendations is adjusted (contextualized) for each user using the contextual information.

1. Ignore context, use a regular 2D RS to generate a ranked list

2. Filter out recommendations (e.g. movies must have at least two of the preferred actors in a given context - heuristic)
3. Adjust the ranking (e.g. weighting the predicted rating with the probability of relevance (model-based)

Re-ranking: create recommendation list by reordering recommendations after they have been generated by an other recommendation algorithm. Achieve an optimal balance between similarity and diversity, or between general relevance and contextual relevance.

Maximal marginal relevance: reduce redundancy of results while maintaining query relevance for already ranked items.

Contextual modelling

Contextual information is used directly in the modeling technique as part of rating estimation.

Multidimensional recommender functions: heuristic-based, model-based

Heuristic-based: adapt nearest neighbor prediction - weighted sum of relevant ratings.

Model-based: Markov Chain Monte Carlo, support vector machine for contextual modeling (liked and disliked in various contexts, construct separating hyperplane in n-dim space, this hyperplane represents a classifier for future recommendation)

Conclusion

Control of contexts, and in particular mood, has positive effects on perceived recommendation quality and diversity. Users tend to modify context and user profile and in the relaxing scenarios.

1. Enabling user control of context leads to increase perceived quality
2. May consider offer fine-grained control for mood to increase perceived quality and diversity
3. Adapting control settings to a particular scenario, such as fewer control options when the user is performing a focused task

Lecture 9 - Ethics, bias and fairness in recommender systems

Fairness basic concepts

> *Are the benefits and resources it provides fairly allocated between different people or organizations it affects?*

Individual vs group fairness

Individual fairness: similar individuals should be treated similarly. Similarity defined according to a task.

Group fairness: Considers system's behavior with regard to group membership or identity. Usually, identify a subset of features called sensitive attributes who identify such group membership (like ethnicity or gender). We identify a protected group. The goal of group fairness is to address the impact of belonging to a protected group on the system final decision. The precise formulation of this goal rely on different sub-concepts of discrimination the system should avoid.

Disparate treatment: the sensitive attribute is a direct feature in the decision-making process

Disparate impact: e.g. one racial group has a higher loan approval than another

Disparate mistreatment: look at the distribution of errors (one group has a higher false positive rate than another, e.g. members of one racial group have highest chances to see their loan request rejected even if they are qualified).

Distributional vs representational harms

Harms connected to unfairness can be divided in:

distributional harm: a resource or opportunity is inequitably distributed

representational harm: people are represented in the system in a way that is inaccurate and systematically unfair

Anti-classification vs anti-subordination

According to the motivation of fairness:

Anti-classification: the influence of a protected attribute should not play a role in decisions

Anti-subordination: current decision-making processes should actively work to reverse the effects of historical patterns of discrimination

Stakeholder fairness

Customer fairness: treating different users of the system fairly. Ensure that they receive comparable service. no groups of user should systematically disadvantaged

Provider fairness: treating the creators or providers of the contents in the system fairly. Ensure equal opportunity to be recommended.

Subject fairness: fair treatment of the people or entities that the items are about. For instance, a new recommendation should not systematically under-represent topics or segments of society.

Studying fairness

Defining fairness

> Take a concern (usually, reducing a particular harm) and translate it intro specific framework and metric or objective function

For example: musicians of one ethnicity are more likely to be recommended than musicians of another?

Data collection

Problem of finding appropriate data for studying a fairness objective under consideration. For instance, gender and age might be difficult to include due to privacy regulations.

In come cases, a group can be synthesized from available data (e.g. look at fairness with the respect to the age of a movie)

Infer label - common ways of doing this include statistical gender recognition based on names or computer vision techniques for gender recognition based on portraits or profile picture. Generally discouraged.

💡 No matter the technique used, it is crucial to carefully document, Data sources, Integration or augmentation decisions, Known limitations of the data, Due to the length limitations of many publication venues, this may necessitate appendices or online supplements

Experiments design and results reporting

Experimental design

- Follow the structure of typical recommender system design
- Offline or Online settings
- Classic validation strategies (Holdout, K-fold....)
- Differentiate n the additional data and metrics used (for instance gender composition of user's recommendations, compared to the same evaluated on user's feedback history)

Reporting results

Results of fairness study need to be carefully described and contextualized. Special care needs to be taken to understand implications of data, experimental limitations of the results, results generalizability. It is important to clearly separate the social goal and the specific operationalization.

Consumer and provider fairness

Consumer fairness: concerns about how the recommendation impact consumers or subgroups of consumers

Provider fairness: providers are primarily the suppliers of information or items being recommended. The key benefit of a recommender system for providers is that the system provides exposure of their items to recommendation. Diversity of recommendations may be related to provider fairness

Provider utility: utility of recommendation to providers. A key notion in provider-side fairness is *exposure*. Provider utility must be measured cumulatively over a sequence of recommendation results delivered to users. Similar items should receive similar utility from the RS.

Other stakeholders

Multi-stakeholder recommender systems impose to consider others raising fairness concerns to address. In some settings, regulatory agencies may be the most important stakeholders for deciding the minimum legal standards with respect to fairness and / or non-discrimination that a system must meet. In other cases, the structure of a platform includes stakeholders who are neither consumers nor providers, but are still impacted by a specific parameters of transactions on the platform. For example: Uber eats.

Fairness mitigation

There is no one solution for fairness problems.

- **Fairness in data:** identify different strategies (if there is a group that is under-represented in the training data - > collect more labeled data for that group; modify the ranking or scoring model; change used features;)
- **Fairness through ranking:** re-ranking lists that are already optimized for relevance to improve fairness. We distinguish between Global (treating the problem as a global optimization task) and Local (single recommendation list) approaches.

Lecture 10 - Person to person and multi-stakeholder recommendations

Person to person recommendation

> Also called "social matching"

Types of connections: People to connect with (symmetric, known?), people to follow (asymmetric, not confirmed), strangers to get to know (symmetric, confirmed, unknown)

▼ **Recommending people to connect with**

Content: profile entries, status, photo text, shared lists, job title, location, description, tag

Content + Link (C+L): sequence 3-4 users where each connects (at least 1-way connection or comment)

Friend of a Friend (FoF): number of mutual friends

SONAR (familiarity signals): org-chart relationships (peers, manager-employee etc.), paper and patent co-authorship, project co-membership, blog commenting, person tagging, connection on another SNS, wiki co-editing and file sharing.

Content / Content + L produced mostly **unknown** people. SONAR (familiarity signals) and FoF produced mostly **known** individuals. The FoF algorithm was able to produce recommendations for only 57.2% of the users (compared to 87.7% for SONAR).

More geographically diverse: SONAR, most people who can recieve recs: SONAR, more unknown people: Content/Content+L

▼ **Centrality measure**

Degree centrality assigns an importance score based simply on the number of links held by each node (→ popularity)

Betweenness centrality measures the number of times a node lies on the shortest path between other nodes (for finding the individuals who influence the flow around the system)

Closeness centrality scores each nodes based on their "closeness" to all other nodes in the network (for finding individuals who are best placed to influence the entire network most quickly)

EigenCentrality measures a node’s influence based on the number of links it has to other nodes in the network. EigenCentrality then goes a step further by also taking into account how well connected a node is, and how many links their connections have, and so on through the network

Page rank same as eigen centrality but takes into account weight and rank

Multi-stakeholder recommendations

Stakeholders: consumer (aka users, e.g. users), providers (aka suppliers, e.g. writers), system (e.g. library)

Consumer fairness: treating different users of the system fairly by ensuring that users receive comparable service and no group of users is systematically disadvantaged by the system. For example, a system might be more inaccurate for some groups of users rather than others.

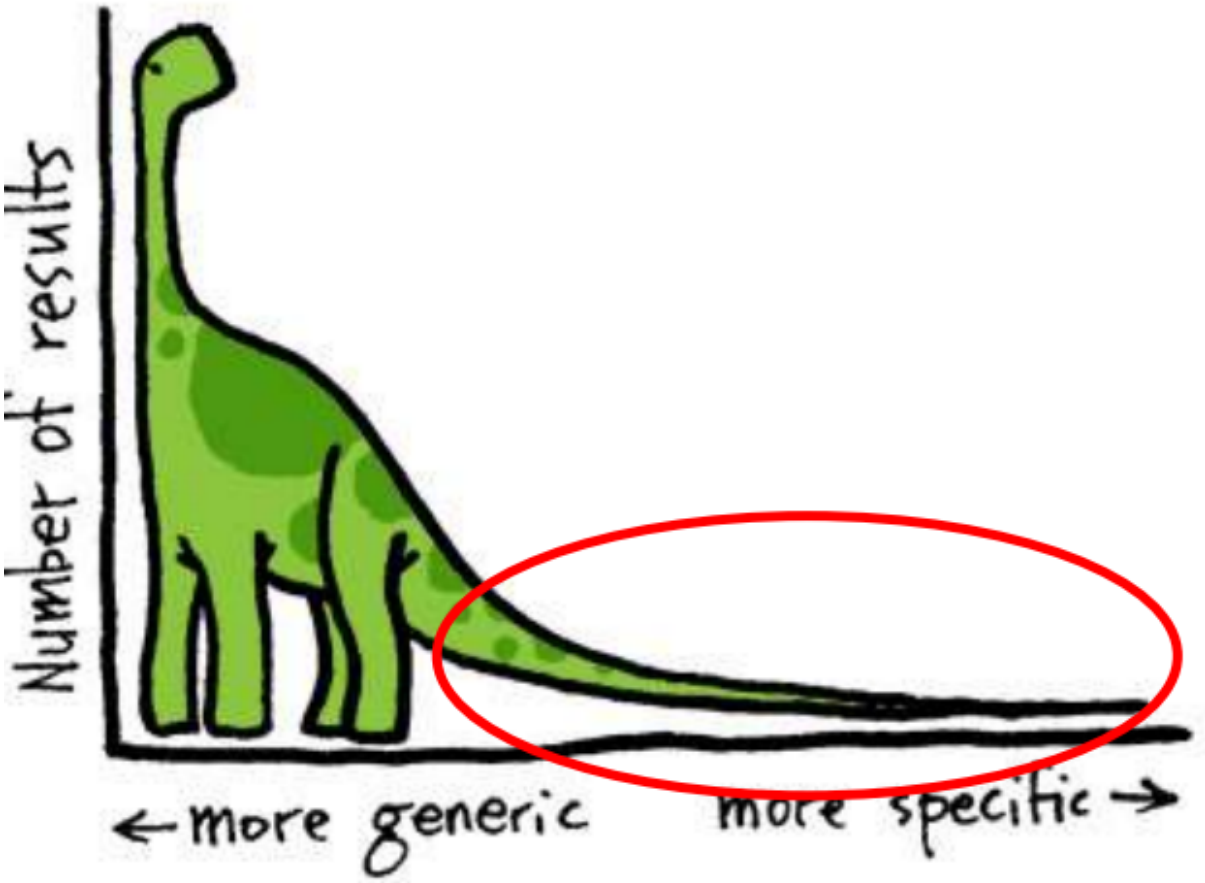
Provider fairness: treating the creators or providers of items fairly, for example by ensuring that musicians have equitable opportunity for their music to be recommended to users in a streaming music platform

Subject fairness: concerned with fair treatment of the people or entities that items are about. One example of a domain where this concern may arise is news recommendation, where recommendations that systematically under-represent topics or segments of society can be considered subject-unfair.

Traditionally: accuracy versus diversity

Long-tail novelty – Expected Popularity Complement - the expected number of (new) read relevant recommended items (not previously read)

Unexpectedness – Expected Profile Distance



Popularity-bias: RS often focus on items that are frequently rated, even though there are many others items of value to users. Small number of items take up the majority of rating interactions.

back to table of contents