Recommender Systems

Predictive Analytics: Week 12-13

Yicheng Song

Assistant Professor
Carlson School of Management
ycsong@umn.edu

Outline

- Introduction
- Collaborative Filtering
  - Memory based methods
  - Model based methods
- Content-based Filtering
- Evaluation Strategies

Examples

movielens
grouplens
University of Minnesota

NETFLIX

YAHOO! MUSIC

amazon.de

SNAP

twitter

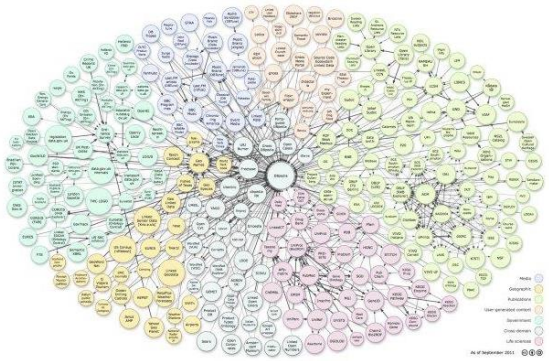
dblp
computer science bibliography

MusicBrainz

last.fm

the echonest

Linked open Data



Google

amazon sale driven by recommender system

People also ask

What percentage of Amazon's sales are due to its recommendation system?

35 percent

McKinsey estimated that 35 **percent** of consumer purchases on Amazon come from product **recommendations**, although **the** e-commerce giant itself has never revealed **its** own estimates. Oct 30, 2017

martechtoday.com › roi-recommendation-engines-mark...

The ROI of recommendation engines for marketing

Predictive

Y OF MINNESOTA
to Discover®

Why using Recommender System

- Value for the customer
  - Find things that are interesting
  - Narrow down the set of choices
  - Help me explore the space of options
  - Discover new things
  - Entertainment
  - …
- Value for the provider
  - Additional and probably unique personalized service for the customer
  - Increase trust and customer loyalty
  - Increase sales, click trough rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers
  - …

Purpose of RS

- Retrieval perspective
  - Reduce search costs
  - Provide "correct" proposals
  - ***Users know in advance what they want***

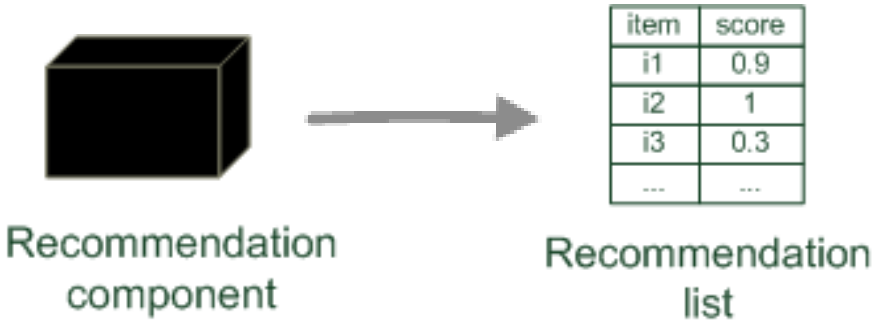
- Recommendation perspective
  - Serendipity – identify items from the Long Tail
  - ***Users did not know about the existence of item***

Recommender systems

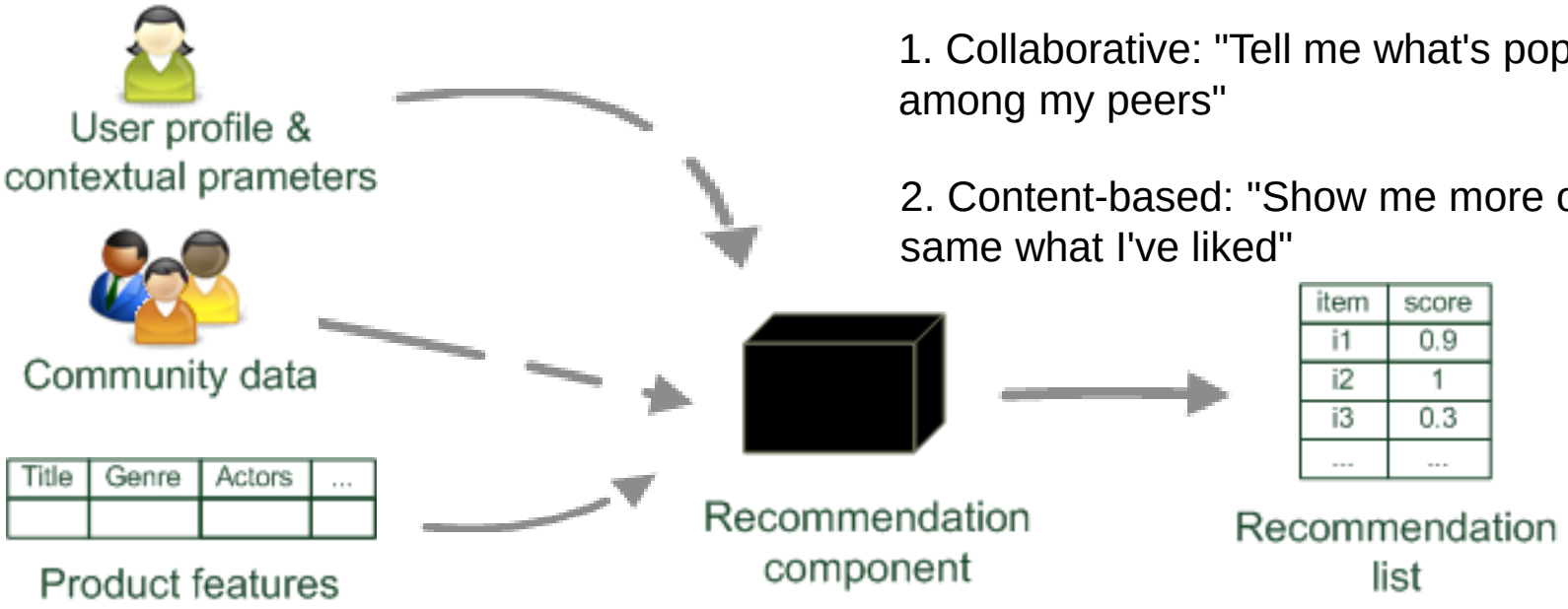
- RS seen as a function
- Given:
  - User data (e.g. ratings, history, demo, situational context)
  - Items (with or without description of item characteristics)
- Find:
  - Relevance score. Used for ranking.

At a high level

Recommender systems reduce information overload by estimating relevance



At a high level—Inputs



1. Collaborative: "Tell me what's popular among my peers"

2. Content-based: "Show me more of the same what I've liked"

3. Hybrid Recommendations

Recommender systems: basic techniques

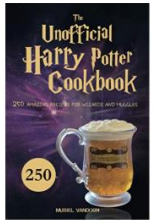
	Pros	Cons
Collaborative	No knowledge-engineering effort, serendipity of results, learns market segments	Requires some form of rating feedback, cold start for new users and new items
Content-based	No community required, comparison between items possible	Content descriptions necessary, cold start for new users, no surprises

Collaborative Filtering (CF)

Collaborative Filtering (CF)

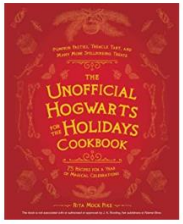
- The most prominent approach to generate recommendations
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- Approach
  - use the "wisdom of the crowd" to recommend items

Customers who viewed this item also viewed

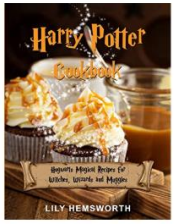


The Unofficial Harry Potter Cookbook: 250...
Muriel Vandorn
★★★★★ 14
Paperback
$10.99
prime FREE Delivery

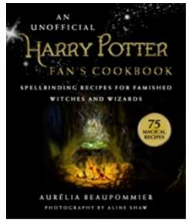
The Unofficial Harry Potter Spellbook:...
› Michael Gonzalez
★★★★½ 1,398
Paperback
$12.85
prime FREE Delivery



The Unofficial Hogwarts for the Holidays...
› Rita Mock-Pike
★★★★★ 22
Hardcover
$14.95
prime FREE Delivery

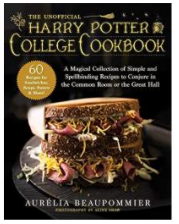


Harry Potter Cookbook: Hogwarts Magical...
› Lily Hemsworth
★★★★☆ 44
Paperback
$19.89
FREE Delivery
Usually ships within 1 to 2...



An Unofficial Harry Potter Fan's Cookbook: Spellbinding Recipes...
› Aurélia Beaupommier
★★★★½ 201
Hardcover
1 offer from $34.48

The Exclusive Harry Potter Cookbook – 30...
› Ina Deen
★★★★½ 314
Paperback
$12.95
prime FREE Delivery



The Unofficial Harry Potter College...
› Aurélia Beaupommier
★★★★½ 14
Hardcover
$14.96
prime FREE Delivery

Pure CF Approaches

- Input
  - Only a matrix of given user–item ratings
  - No other user or item information is needed
- 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

Nearest-Neighbors (kNN)

- A "pure" CF approach and traditional baseline
  - Uses a matrix of (explicit) ratings provided by the community as inputs
  - Returns a ranked list of items based on rating predictions
- Solution approach
  - Given an "active user" (Alice) and an item I not yet seen by Alice
  - Estimate Alice's rating for this item based on like-minded users (peers)
- Assumptions
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

User-based nearest-neighbor collaborative filtering

- Some first questions
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Measuring user similarity

- A popular similarity measure in user-based CF: **Pearson correlation**

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\ \sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

a, b : users

$r_{a,p}$: rating of user a for item p

P : set of items, rated both by a and b

Possible similarity values between -1 and 1; $\overline{r_a}$, $\overline{r_b}$ = user's average ratings

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,65
sim = 0,70
sim = -0,79

Making predictions

- A common prediction function:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

- Calculate, whether the neighbors' ratings for the unseen item *i* are higher or lower than their average
- Combine the rating differences – use the similarity as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

  - How many neighbors?
    - Only consider positively correlated neighbors (or higher threshold)
    - Can be optimized based on data set
    - Often, between 50 and 200

Improving the metrics / prediction function

- Not all item-ratings might be equally "valuable"
  - Which rating is more valuable? That on **Star Wars**, or **Clockwork Orange**?
  - Agreement on commonly liked items is not so informative
  - **Possible solution**: Give more weight to items that have a higher variance
- Similarity based on few vs many overlaps
  - Value of number of co-rated items
  - Use "significance weighting" by reducing the weight when the number of co-rated items is low
- Neighborhood selection
  - Use similarity threshold instead of fixed number of neighbors

6'th International Symposium on Telecommunications (IST'2012)

An Enhanced Significance Weighting Approach for Collaborative Filtering

Challenges with user-based CF

- Very simple scheme leading to quite accurate recommendations
  - Still today often used as a baseline scheme
- Possible issues
  - Scalability
    - Thinking of millions of users and thousands of items
    - Pre-computation of similarities possible but potentially unstable
    - Clustering techniques are often less accurate
  - Coverage
    - Problem of finding enough neighbors
    - Users with preferences for niche products

The ACM Conference Series on Recommender Systems

Item-Based Collaborative Algor

Badrul Sarwar, George Karypis, {sarwar, karypis, kons
GroupLens Research Group
Department of Computer
University of Minnesota

HOME RECSYS 2019 PAST CONFERENCES HONORS BLOG CO

1st ACM Conference on Recommender Systems
Minneapolis, Minnesota, USA, 19th-20th October 2007
ACM Recommender Systems 2007 has been built on a wonderful legacy of research workshop

[PDF] Item-based collaborative filtering recommendation algorith

B Sarwar, G Karypis, J Konstan, J Riedl - Proceedings of the 10th …, 2001 - dl.a

Recommender systems apply knowledge discovery techniques to the problem of personalized recommendations for information, products or services during a live interaction. These systems, especially the k-nearest neighbor collaborative filterin ones, are achieving widespread success on the Web. The tremendous growth in of available information and the number of visitors to Web sites in recent years p key challenges for recommender systems. These are: producing high quality …

☆ Cited by 9451 Related articles All 35 versions

Item-based collaborative filtering

- Basic idea:
  - Use the similarity between items (and not users) to make predictions
- Example:
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Making predictions

- A common prediction function:

$$pred(u,p) = \frac{\sum_{i \in ratedItem(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i,p)}$$

- An analysis of the MovieLens dataset indicates that "in most real-world situations, a neighborhood of 20 to 50 neighbors seems reasonable" (Herlocker et al. 2002)

Pre-Processing for item-based filtering

- Item-based filtering does not solve the scalability problem itself
- Pre-processing approach by Amazon.com (in 2003)
  - 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
- Memory requirements
  - Up to N^2 pair-wise similarities to be memorized (N = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)
  - Further reductions possible
    - Minimum threshold for co-ratings
    - Limit the neighborhood size (might affect recommendation accuracy)

Memory- and Model-based approaches

- kNN methods are often 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

Model Based Approach

- Variety of techniques proposed in recent years, e.g.,
  - Matrix factorization techniques
    - singular value decomposition, principal component analysis
  - Association rule mining
    - compare: shopping basket analysis
  - Probabilistic models
    - clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
  - Various other machine learning approaches
    - Regression-based techniques, deep neural networks, …

Probabilistic method

- Basic idea (simplistic version for illustration):
  - given the user/item rating matrix
  - determine the probability that user Alice will like an item i
  - base the recommendation on such these probabilities
- Calculation of rating probabilities based on Bayes Theorem
  - How probable is rating value "1" for Item5 given Alice's previous ratings?
  - Corresponds to conditional probability P(Item5=1 | X), where
    - X = Alice's previous ratings = (Item1 =1, Item2=3, Item3= …)
  - Can be estimated based on Bayes' Theorem

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)} \qquad P(Y|X) = \frac{\prod_{i=1}^{d} P(X_i|Y) \times P(Y)}{P(X)}$$

  - Assumption: Ratings are independent (?)

Probabilistic method

	Item1	Item2	Item3	Item4	Item5
Alice	1	3	3	2	?
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1

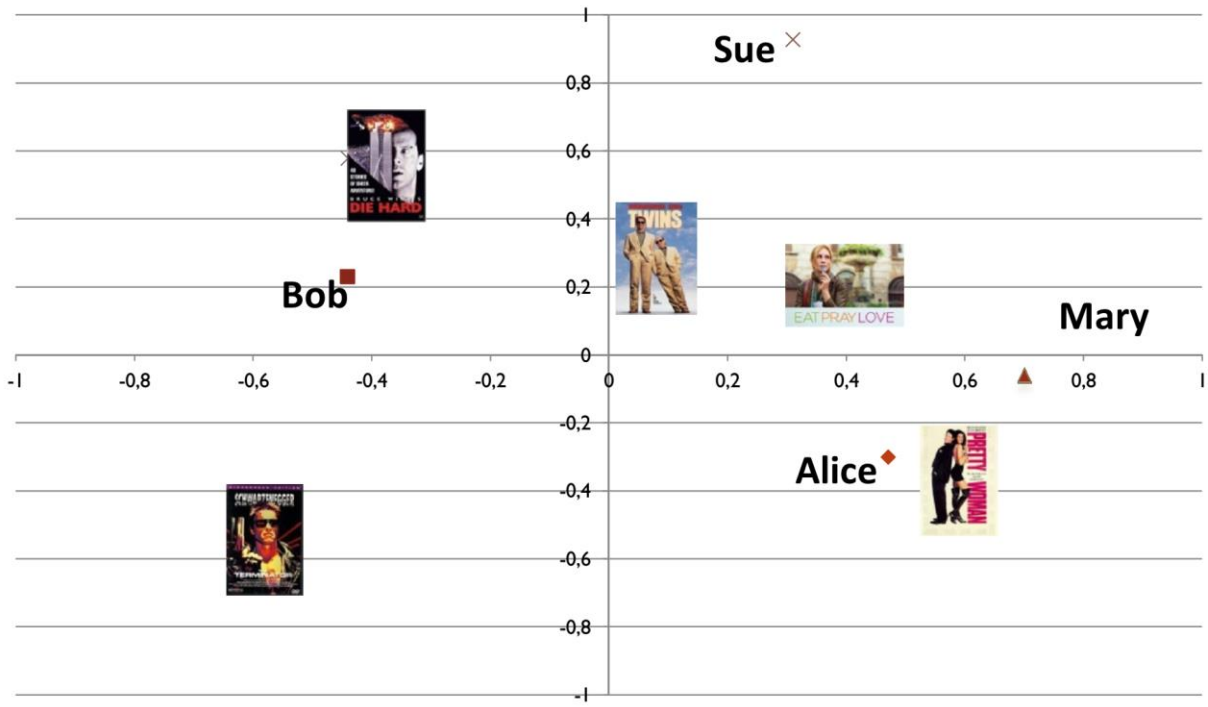
$X = (\text{Item1} = 1, \text{Item2} = 3, \text{Item3} = \dots)$

$$P(X|\text{Item5} = 1)$$
$$= P(\text{Item1} = 1|\text{Item5} = 1) \times P(\text{Item2} = 3|\text{Item5} = 1)$$
$$\times P(\text{Item3} = 3|\text{Item5} = 1) \times P(\text{Item4} = 2|\text{Item5} = 1) = \frac{2}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2}$$
$$\approx 0.125$$

$$P(X|\text{Item5} = 2)$$
$$= P(\text{Item1} = 1|\text{Item5} = 2) \times P(\text{Item2} = 3|\text{Item5} = 2)$$
$$\times P(\text{Item3} = 3|\text{Item5} = 2) \times P(\text{Item4} = 2|\text{Item5} = 2) = \frac{0}{0} \times \cdots \times \cdots \times \cdots = 0$$

Latent Space

- Assume that both movies and users live in some **low-dimensional space** describing their properties
- **Recommend** a movie based on its **proximity** to the user in the latent space



Matrix factorization

- User vectors:

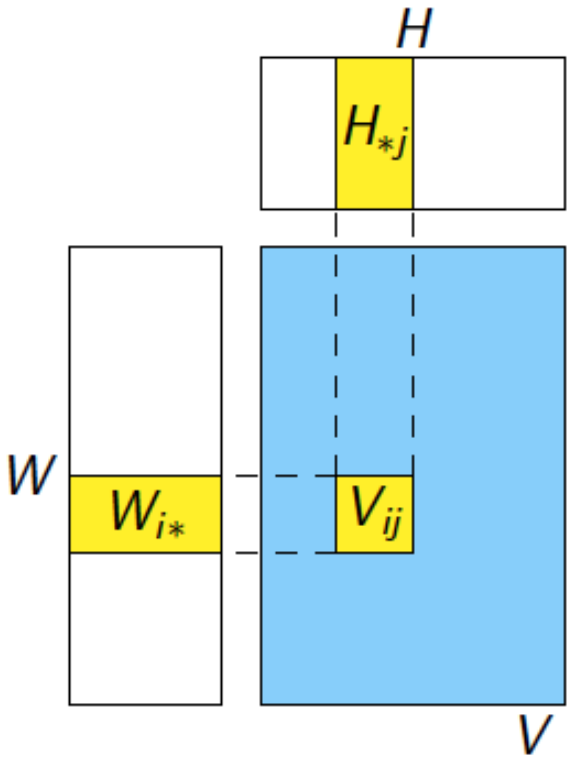
  $(W_{u*})^T \in \mathbb{R}^r$

- Item vectors:

  $H_{*i} \in \mathbb{R}^r$

- Rating prediction:

  $$V_{ui} = W_{u*} H_{*i}$$
  $$= [WH]_{ui}$$



Optimization problem

- Set of non-zero entries:

$$\mathcal{Z} = \{(u, i) : v_{ui} \neq 0\}$$

- Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$

Optimization problem

- Set of non-zero entries:

$$\mathcal{Z} = \{(u, i) : v_{ui} \neq 0\}$$

- Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$

- With Regularization:

$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2 + \lambda(\sum_i ||\mathbf{w}_i||^2 + \sum_u ||\mathbf{h}_u||^2)$$

Rating Prediction

- User vectors:

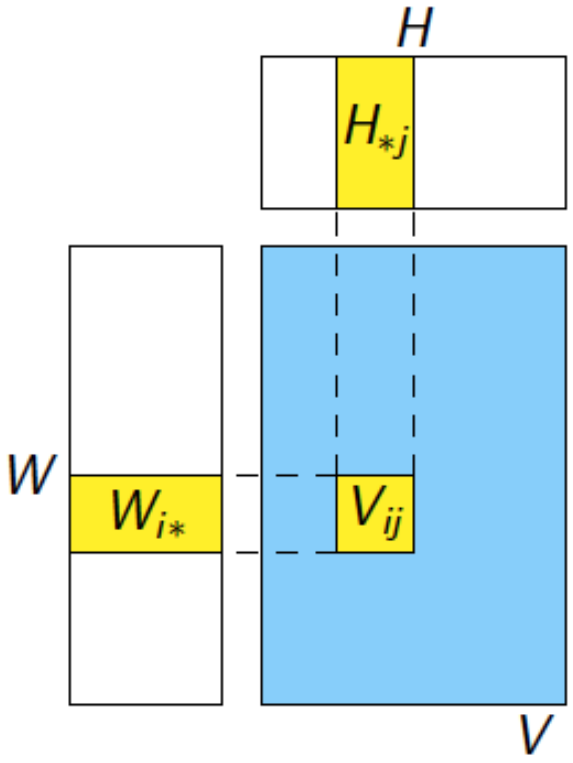
$$(W_{u*})^T \in \mathbb{R}^r$$

- Item vectors:

$$H_{*i} \in \mathbb{R}^r$$

- Rating prediction:

$$V_{ui} = W_{u*} H_{*i}$$
$$= [WH]_{ui}$$

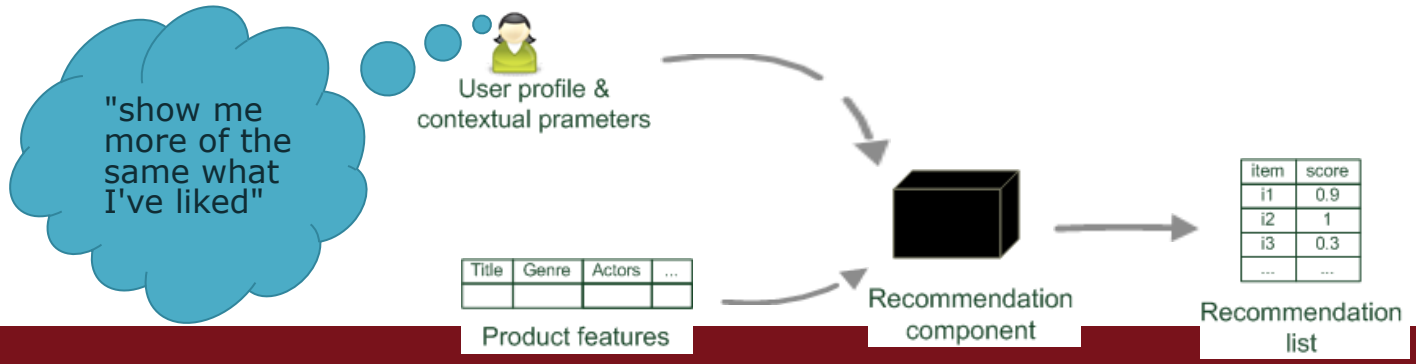


Collaborative Filtering Issues

- Pros: 👍
  - well-understood, works generally well, no knowledge engineering required
- Cons: 👎
  - requires user community, sparsity problems
- What is the best CF method?
  - In which situation and which domain?
  - differences between each well tuned methods are often very small (1/100)

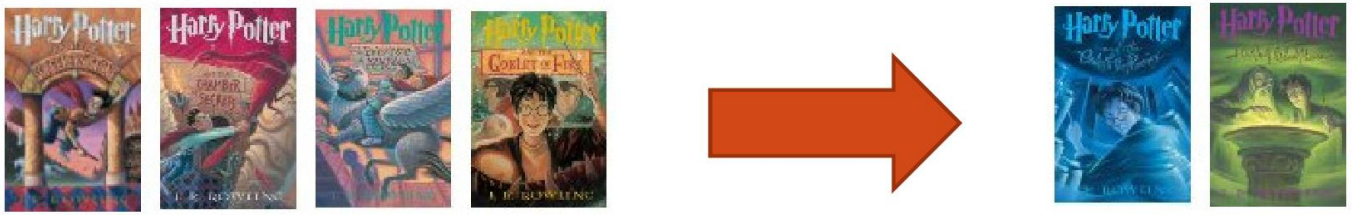
Content-based recommendation

- **While CF – methods do not require any information about the items,**
  - it might be reasonable to exploit such information; and
  - recommend fantasy novels to people who liked fantasy novels in the past
- **What do we need:**
  - some information about the available items such as the genre ("content")
  - some sort of *user profile* describing what the user likes (the preferences)
- **The task:**
  - learn user preferences
  - locate/recommend items that are "similar" to the user preferences



Content-based Filtering

- Again:
  - Determine preferences of user based on past behavior
- This time, however:
  - Look at what the current user liked (purchased, viewed, …)
  - Estimate the user's preference for certain item features
    - e.g., genre, authors, release date, keywords in the text
  - Alternative preference acquisition
    - ask the user, look at recently viewed items



What is content?

- Most CB-recommendation techniques were applied to recommending text documents.
  - Like web pages or newsgroup messages for example.
- Content of items can also be represented as text documents.
  - With textual descriptions of their basic characteristics.
  - Structured: Each item is described by the same set of attributes
  - Unstructured: free-text description.

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

TF-IDF

- Simple keyword representation has its problems
  - in particular when automatically extracted:
    - not every word has similar importance
    - longer documents have a higher chance to have an overlap with the user profile
- Standard measure: TF-IDF
  - Encodes text documents in multi-dimensional Euclidian space
    - weighted term vector
  - TF: Measures, how often a term appears (density in a document)
    - assuming that important terms appear more often
    - normalization has to be done in order to take document length into account
  - IDF: Aims to reduce the weight of terms that appear in all documents

TF-IDF

- Given a keyword i and a document j
- $TF(i,j)$
  - term frequency of keyword i in document j
  - Term frequency is relative to most frequent term z in document j

$$TF(i,j) = \frac{Frequency(i,j)}{max_z\, Frequency(z,j)}$$

- $IDF(i)$
  - inverse document frequency calculated as $IDF(i) = log\frac{N}{n(i)}$
    - N : number of all recommendable documents
    - $n(i)$: number of documents from N in which keyword i appears
- $TF - IDF$
  - is calculated as: $TF\text{-}IDF(i,j) = TF(i,j) * IDF(i)$
- Normalization
  - Vector of length 1

$$TF - IDF(i,j) = \frac{TF - IDF(i,j)}{\sqrt{\sum_s TF - IDF(s,j)^2}}$$

Term Frequency

- Absolute term frequency:
  - Each document is a count vector in $\mathbb{N}^{|v|}$

	Poem A	Poem B	Poem C
Caesar	232	0	2
Calpurnia	0	10	0
Cleopatra	57	0	0

Vector v with dimension $|v| = 3$

$$TF(i,j) = \frac{Frequency(i,j)}{max_z\, Frequency(z,j)}$$

TF	Poem A	Poem B	Poem C
Caesar	1	0	1
Calpurnia	0	1	0
Cleopatra	0.24	0	0

Inverse Document Frequency

	Poem A	Poem B	Poem C
Caesar	232	0	2
Calpurnia	0	10	0
Cleopatra	57	0	0

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

IDF	Poem A	Poem B	Poem C
Caesar	0.58	0	0.58
Calpurnia	0	1.58	0
Cleopatra	1.58	0	0

$$TF\text{-}IDF(i,j) = TF(i,j) * IDF(i)$$

TF-IDF	Poem A	Poem B	Poem C
Caesar	0.58	0	1
Calpurnia	0	1.58	0
Cleopatra	0.39	0	0

$$TF - IDF(i,j) = \frac{TF - IDF(i,j)}{\sqrt{\sum_s TF - IDF(s,j)^2}}$$

Norm. TF-IDF	Poem A	Poem B	Poem C
Caesar	0.83	0	1
Calpurnia	0	1	0
Cleopatra	0.55	0	0

Improving the vector space mode

- Vectors are usually long and sparse
- Remove stop words
  - They will appear in nearly all documents.
  - e.g. "a", "the", "on", …
- Use stemming
  - Aims to replace variants of words by their common stem
  - e.g. "went" ⟶ "go", "stemming" ⟶"stem", …
- Size cut-offs
  - only use top n most representative words to remove "noise" from data
  - e.g. use top 100 words
- Tuning of representation
  - Logarithmic instead of linear TF count

Limitations of content-based methods

- Keywords alone may not be sufficient to judge quality/relevance of a document or web page
  - up-to-date-ness, usability, aesthetics, writing style
  - content may also be limited / too short
  - content may not be automatically extractable (multimedia)
- Ramp-up phase required
  - Some training data is still required
  - Web 2.0: Use other sources to learn the user preferences
- Overspecialization
  - Algorithms tend to propose "more of the same"
  - Or: too similar news items

Evaluating Recommender Systems

Evaluation Approach

- Test with real users
  - A/B tests
  - Example measures: sales increase, click through rates
- Laboratory studies
  - Controlled experiments
  - Example measures: satisfaction with the system (questionnaires)
- Offline experiments
  - Based on historical data
  - Example measures: prediction accuracy, coverage

Metrics: MAE and RMSE

- Recommendation is concerned with learning from noisy observations (x,y), where $f(x)=\hat{y}$ has to be determined such that $\sum_{\hat{y}}(\hat{y}-y)^2$ is minimal.
- Experimental setup
  - Historic user ratings constitute ground truth (e.g., MovieLens movie ratings, 100k ratings to 10 million; 100 mio. ratings for Netflix Prize)
  - Predict hidden ratings
  - **Mean Absolute Error** (*MAE*) computes the deviation between predicted ratings and actual ratings

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

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

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

Example

Nr.	UserID	MovieID	Rating (r_i)	Prediction (p_i)	$\lvert p_i - r_i \rvert$	$(p_i - r_i)^2$
1	1	134	5	4.5	0.5	0.25
2	1	238	4	5	1	1
3	1	312	5	5	0	0
4	2	134	3	5	2	4
5	2	767	5	4.5	0.5	0.25
6	3	68	4	4.1	0.1	0.01
7	3	212	4	3.9	0.1	0.01
8	3	238	3	3	0	0
9	4	68	4	4.2	0.2	0.04
10	4	112	5	4.8	0.2	0.04
					4.6	5.6

- MAE = 0.46
- RMSE = 0.75

Removing line nr. 4

- MAE = 0.29
- RMSE = 0.42

Removing lines 1,2,4,5

- MAE = 0.1
- RMSE = 0.13

Metrics: Precision and Recall

- Recommendation is viewed as information retrieval task:
  - Retrieve (recommend) all items which are predicted to be “good”.
- Precision: a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved
  - E.g. the proportion of recommended movies that are actually good

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|\text{all recommendations}|}$$

- Recall: a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items
  - E.g. the proportion of all good movies recommended

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

F_1 Metric

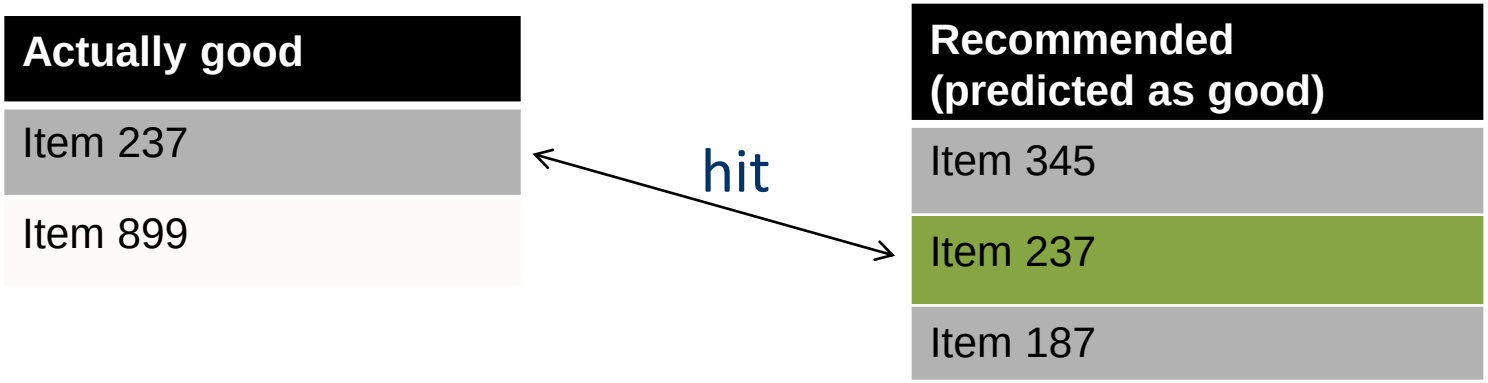
- The F_1 Metric attempts to combine Precision and Recall into a single value for comparison purposes.
  - May be used to gain a more balanced view of performance

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

- The harmonic mean of precision and recall

Metrics: Rank position matters

For a user:



- Rank metrics extend recall and precision to take the positions of correct items in a ranked list into account
  - Relevant items are more useful when they appear earlier in the recommendation list
  - Particularly important in recommender systems as lower ranked items may be overlooked by users

Metrics: Rank Score

- Measure top-k ***item recommendation*** performance
  - i.e., the quality of the top-k recommendations produced by the system
    - In contrast to **rating prediction** performance (the quality of rating predictions for the unknown items)
  - Basic idea: do relevant items appear in top-k recommendations
  - Such recommendation techniques can work with 0/1 relevance information (i.e., where more precise rating information may not be available, e.g., purchase/no-purchase)
- Precision-at-top-k
  - Percentage of relevant items among top-k recommendations
- Mean average precision (MAP)
  - Averaging precision across different levels of recall
- Normalized discounted cumulative gain (NDCG)
  - Discounted cumulative gain (DCG): penalty for low-relevance documents appearing in high positions of the list (discounted as we go down the list)
  - NDCG is the DCG but normalized based on ideal ranking (ideal DCG), so that NDCG $\in$ [0,1]

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \qquad \mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

NDCG Example

- There are 6 items to rank: I1 to I6
- Relevance scores (0-3) scale:
  - 3,2,3,0,1,2
- DCG at 6:

$$\mathrm{DCG}_6 = rel_1 + \sum_{i=2}^{6} \frac{rel_i}{\log_2 i} = 3 + (2 + 1.892 + 0 + 0.431 + 0.774) = 8.10$$

- An ideal ordering IDCG:
  - 3,3,2,2,1,0 would lead to an DCG of 8.69
- The nDCG
  - DCG/IDCG = 8.10/8.69 = 0.932