Extração e Análise de Dados na Web

Recommender Systems

Departamento de Engenharia Informática
Instituto Superior Técnico

2^{o} Semestre
2012/2013

Outline

Bibliography

- Dietmar Jannach and Gerhard Friedrich, *Tutorial: Recommender Systems*, International Joint Conference on Artificial Intelligence Barcelona, 2011.
- Francesco Ricci, Lior Rokach and Bracha Shapira, *Recommender Systems Handbook*, Springer, 2011.

What is a Recommender System?

Recommender System

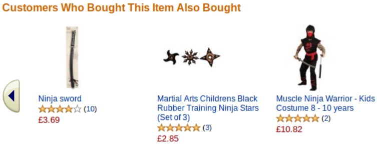
Recommender Systems (RS) are software tools and techniques providing suggestions for items to be of use to a user.

RS help to match users with items

- books, movies, music, travel, consumer products, news, ...

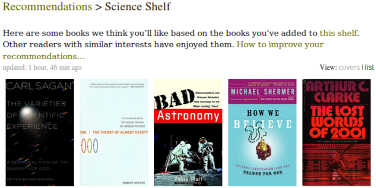
Examples

Amazon

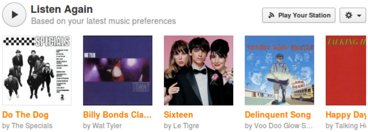


Criticker

Goodreads



Grooveshark



Why Use Recommender Systems?

For users:

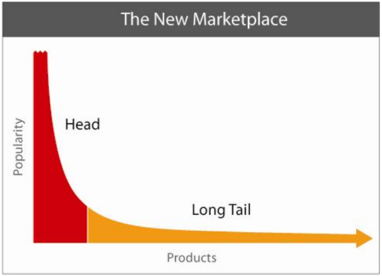
- Ease information overload
- Sales assistance

For providers:

- Can have high commercial value
  - See, for example, the Netflix Prize

Why Use Recommender Systems? (cont.)

Leverage the long tail



(...) our culture and economy is increasingly shifting away from a focus on a relatively small number of "hits" (mainstream products and markets) at the head of the demand curve and toward a huge number of niches in the tail.

Chris Anderson, *The Long Tail: Why the Future of Business is Selling Less of More*, 2004.

Goals of a Recommender System

- **Retrieval:**
  - Reduce search costs
  - Provide accurate proposals
- **Recommendation:**
  - Serendipity — identify items not known to the users
- **Prediction:**
  - Predict to what degree users like an item — most popular evaluation scenario in research
- **Interaction:**
  - Give users a "good feeling"
  - Educate users about the product domain
  - Convince/persuade users — explain
- **Commercial situations:**
  - Increase hit, clickthrough, lookers to bookers rates
  - Optimize sales margins and profit

How Recommendation Works

Recommender System

Input: User model: ratings, preferences, demographics, ...
Items (with or without description of item characteristics)

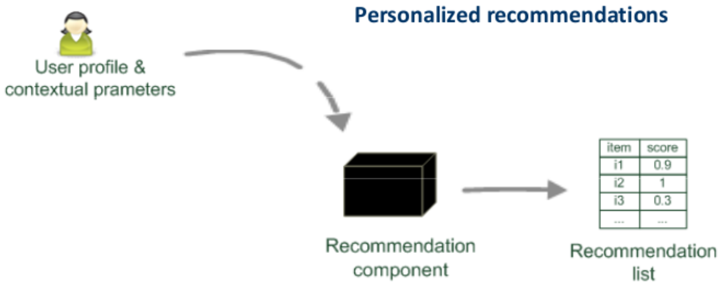
Output: Relevance score

In practice, usually not all items will be scored: the task is to find the most relevant ones.

Paradigms of Recommender Systems

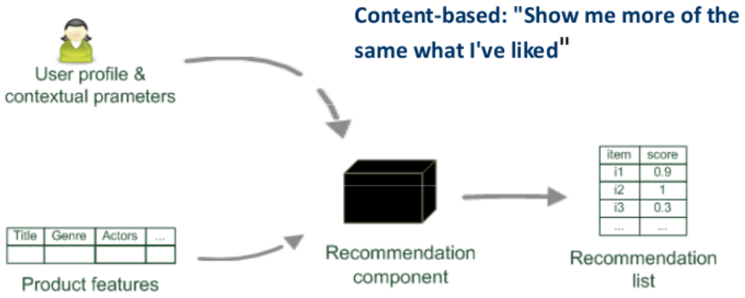
Recommender systems reduce information overload by estimating relevance

Paradigms of Recommender Systems



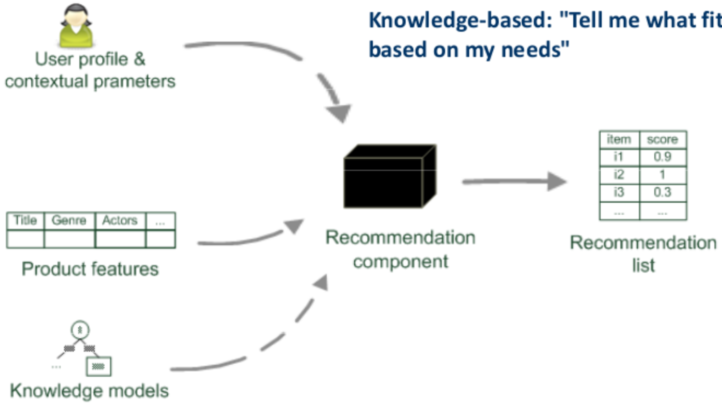
Paradigms of Recommender Systems

Paradigms of Recommender Systems



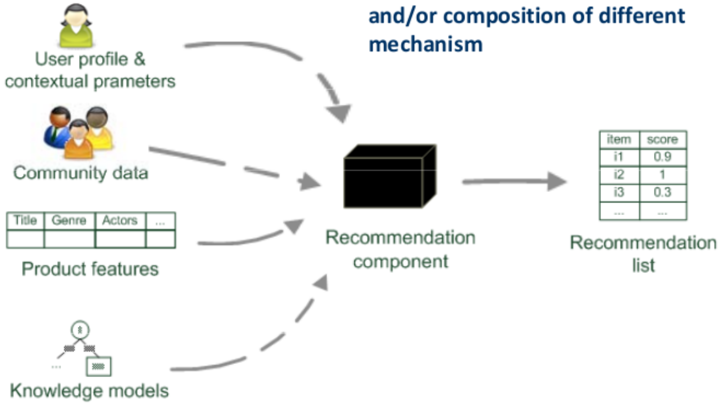
Paradigms of Recommender Systems

Knowledge-based: "Tell me what fits based on my needs"



Paradigms of Recommender Systems

Hybrid: combinations of various inputs and/or composition of different mechanism



Basic Techniques

	Pros	Cons
Collaborative	Nearly no ramp-up 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
Knowledge-based	Deterministic recommendations, assured quality, no cold-start, can resemble sales dialogue	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

Outline

Collaborative Filtering (CF)

- The most prominent approach to generate recommendations
  - Used by large 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
- 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

How CF works

An example: user-based nearest-neighbor collaborative filtering

Given an active user (Alice) and an item I not yet seen by Alice

- The goal is to estimate Alice's rating for this item, e.g., by
  - find a set of users (peers) who liked the same items as Alice in the past and who have rated item I
  - use, e.g. the average of their ratings to predict if Alice will like item I
  - do this for all items Alice has not seen and recommend the best-rated

	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

How CF Works (cont.)

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

An example: Pearson correlation

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

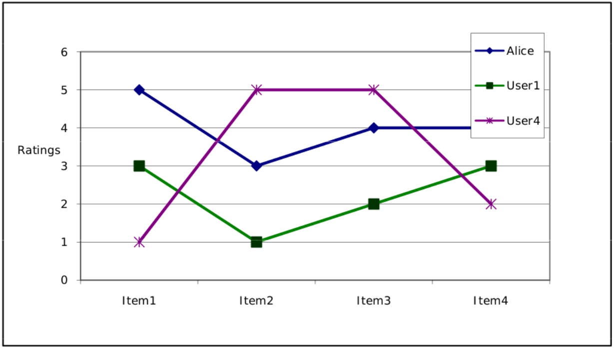
a, b: users
$r_{a,p}$: rating of user a for item p
P: set of items, rated both by a and b
$\overline{r}_a, \overline{r}_b$ user's average ratings
Possible similarity values between -1 and 1;

	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 = 0,85
sim = 0,70
sim = -0,79

Pearson Correlation

- Takes differences in rating behavior into account



- Works well in usual domains, compared with alternative measures
  - such as cosine similarity

Making Predictions

- A common prediction function:

$$pred(a,p) = \overline{r}_a + \frac{\sum_{b \in N} sim(a,b) \times (r_{b,p} - \overline{r}_b)}{\sum_{b \in N} sim(a,b)}$$

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

Improving the Metrics and Prediction

- Not all neighbor ratings might be equally valuable
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - Possible solution: Give more weight to items that have a higher variance
- Value of number of co-rated items
  - Use significance weighting, by e.g., linearly reducing the weight when the number of co-rated items is low
- Case amplification
  - *Intuition:* Give more weight to very similar neighbors, i.e., where the similarity value is close to 1.
- Neighborhood selection
  - Use similarity threshold or fixed number of neighbors

Memory-based and Model-based Approaches

- User-based CF is said to be memory-based
  - the rating matrix is directly used to find neighbors and 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 periodically (can be computationally expensive)
  - large variety of techniques used

Issues with User-based CF

- Scalability issues arise with if many more users than items
  - e.g. amazon.com
  - Space complexity $O(m^2)$, when pre-computed
  - Time complexity for computing Pearson: $O(m^2n)$
- High sparsity leads to few common ratings between two users
- A solution: Item-based CF
  - exploits relationships between items, instead of relationships between users

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

The Cosine Similarity Measure

- Produces better results in item-to-item filtering
  - (for some datasets)
- Ratings are seen as vector in n-dimensional space
- Similarity is calculated based on the angle between the vectors

$$sim(a,b) = \frac{\sum_{u \in U} r_{u,a} r_{u,b}}{\sqrt{\sum_{u \in U} r_{u,a}^2}\sqrt{\sum_{u \in U} r_{u,b}^2}}$$

- Adjusted cosine similarity
  - take average user ratings into account, transform the original ratings

$$sim(a,b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r}_u)(r_{u,b} - \overline{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r}_u)^2}}$$

  - U: set of users who have rated both items a and b

Pre-processing for Item-based CF

- Item-based CF 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 (items, which are rated at least by n users)
    - Limit the size of the neighborhood (might affect recommendation accuracy)

Data sparsity problems

- Cold start problem
  - How to recommend new items? What to recommend to new users?
- Straightforward approaches
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
- Alternatives
  - Use other algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be to small to make good predictions
    - Assume transitivity of neighborhoods

Outline

Content-based recommendation

- While CF methods do not require any information about the items
  - it might be reasonable to exploit such information; and
  - e.g. 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 (preferences)
- The task:
  - learn user preferences
  - recommend items that are similar to the user preferences

Content Representation and Item Similarities

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
...					

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follet, ..	Paperback	25.65	detective, murder, New York

- Simple approach: compute the similarity between the user profile and an unseen item using keyword overlap
  - E.g. use the vector space model (from IR)

Recommending Items

- Simple method: nearest neighbors
  - Given a set of documents D already rated by the user (like/dislike)
    - Find the n nearest neighbors of a not-yet-seen item $i \in D$
    - Take these ratings to predict a rating/vote for i
  - Good to model short-term interests/follow-up stories
  - Used in combination with method to model long-term preferences
- Query-based retrieval: Rocchio's method
  - Users are allowed to rate (relevant/irrelevant) retrieved documents (feedback)
  - The system then learns a prototype of relevant/irrelevant documents
  - Queries are then automatically extended with additional terms/weight of relevant documents

Rocchio Relevance Feedback Method

$$Q_{i+1} = \alpha \times Q_i + \beta\left(\frac{1}{|D^+|} \sum_{d^+ \in D^+} d^+\right) - \gamma\left(\frac{1}{|D^-|} \sum_{d^- \in D^-} d^-\right)$$

D^+, D^-: positive/negative items
Q: query
α, β, γ: used to fine tune the process

Other Methods

- Classifiers
  - E.g. using 2 classes: relevant/non-relevant
- Other information retrieval methods
  - As used by search engines (L2R, PageRank, etc.)

Limitations of Content-based Recommendation Methods

- Keywords alone may not be sufficient to judge quality/relevance of a document or web page
  - content may be limited/too short
  - content may not be automatically extractable (e.g. multimedia)
- Ramp-up phase required
  - Some training data is still required
- Over-specialization
  - Algorithms tend to propose "more of the same"
  - E.g. too similar news items

Outline

Knowledge-Based Recommendation

- Explicit domain knowledge
  - Requirements elicitation from domain experts
  - System mimics the behavior of experienced sales assistant
  - Best-practice sales interactions
  - Can guarantee "correct" recommendations (determinism) with respect to expert knowledge
- Conversational interaction strategy
  - Opposed to one-shot interaction
  - Elicitation of user requirements
  - Transfer of product knowledge ("educating users")

Constraint-based Recommendation

- Knowledge base
  - Usually mediates between user model and item properties
  - Variables
    - User model features (requirements), item features (catalogue)
- Set of constraints
  - Logical implications (IF user requires A THEN proposed item should possess feature B)
  - Hard and soft/weighted constraints
  - Solution preferences
- Derive a set of recommendable items
  - Fulfilling a set of applicable constraints
  - Applicability of constraints depends on current user model
  - Explanations: transparent line of reasoning

Example

Knowledge base

True $\Rightarrow$ brand = brand prefs
Motives = Landscape $\Rightarrow$
Low foc Length $\leq$ 28
True $\Rightarrow$ Price $\leq$ Max cost
...

User model

Motives = Landscape
Brand pref = Canon
Max cost = 350

Catalogue

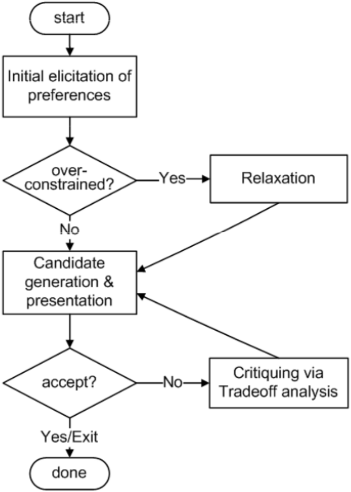
Brand = Canon
Low foc length = 35
Up foc length = 140
Price = 420

Brand = Panasonic
Low foc length = 28
Up foc length = 112
Price = 319

...

Conversational Strategies

- Process consisting of multiple conversational moves
  - Resembles natural sales interactions
  - Not all user requirements known beforehand
  - Customers are rarely satisfied with the initial recommendations
- Different styles of preference elicitation:
  - Free text query interface
  - Asking technical/generic properties
  - Images/inspiration
  - Proposing and Critiquing



Limitations of Knowledge-based Recommendation Methods

- Cost of knowledge acquisition
  - From domain experts
  - From users
  - From web resources
- Accuracy of preference models
  - Very fine granular preference models require many interaction cycles with the user or sufficient detailed data about the user
  - Whereas collaborative filtering models the preference of a user implicitly
- Independence and stability assumption can be challenged
  - Preferences are not always independent from each other and stable

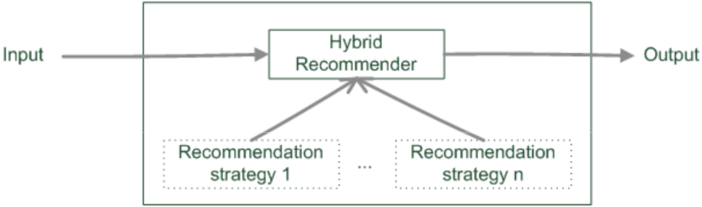
Outline

Different hybridization designs

- Monolithic exploiting different features
- Parallel use of several systems
- Pipelined invocation of different systems

Monolithic Hybridization Design

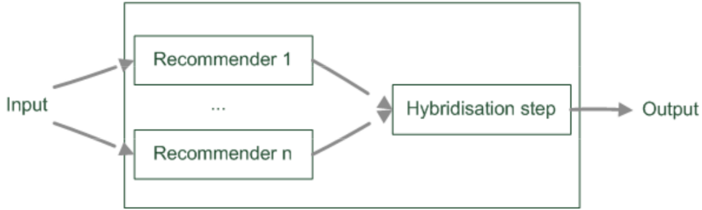
- Only a single recommendation component



- Combination of several knowledge sources
  - E.g. ratings and user demographics or explicit requirements and needs used for similarity computation
- Hybrid features:
  - Social features: Movies liked by user
  - Content features: Comedies liked by user, dramas liked by user

Parallelized Hybridization Design

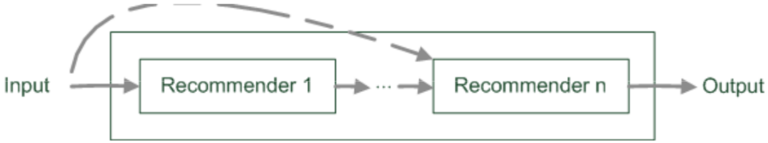
- Output of several existing implementations combined
- Least invasive design
- Weighting or voting scheme applied



  - Weights can be learned dynamically

Pipelined hybridization designs

One recommender system pre-processes some input for the subsequent



- Cascade: Refinement of recommendation lists
  - E.g. first recommender excludes items (e.g. knowledge-based); second recommender assigns score (e.g. collaborative)
- Meta-level: Learning of model
  - E.g. collaborative filtering identifies similar users; knowledge base is built using behavior of similar users

Limitations and Success of Hybridization Strategies

- Most datasets do not allow to compare different recommendation paradigms
  - i.e. ratings, requirements, item features, domain knowledge, critiques rarely available in a single dataset
- Thus few conclusions that are supported by empirical findings
  - Monolithic: some preprocessing effort traded-in for more knowledge included
  - Parallel: requires careful matching of scores from different predictors
  - Pipelined: works well for two antithetic approaches
- Netflix competition—stacking recommender systems
  - Weighted design based on >100 predictors (recommendation functions)
  - Adaptive switching of weights based on user model, parameters (e.g. number of ratings in one session)

Outline

Other Topics

- Explanations
  - A selling agent may be interested in promoting particular products
  - A buying agent is concerned about making the right buying decision
- Evaluation
  - Common metrics: precision, recall, F1, etc.
  - However: real-value lies in increasing conversions and satisfaction with bought items (low churn rate)

Questions?