

NSWI166 – Introduction to Recommender Systems and User Preferences

Ladislav Peska & Peter Vojtas

Ladislav.peska@matfyz.cuni.cz, S208

<https://www.ksi.mff.cuni.cz/~peska/vyuka/NSWI166>

Lecture: Wed 9:00 S5 (:-/)

Labs: Wed 10:40 S6 (every two weeks)

*2/1, ZK+Z, 4
credits*

Organization

- **Active reading #1 results**
 - Amazon Item-Item KNN vs. EASE (shallow autoencoder) approx. 50% vs. 50%
 - Time-aware MF: 0%

- **Overall, good job**
 - Do not forget to add your own perspective
 - Do not forget to add weaknesses
 - Structure the output

Organization

- **Amazon Item-Item KNN**

- Old (*but recently won „Test of the Time“ award IEEE Internet Computing, 2017*)
- Lack of details in algorithm (*common for many industry papers, unfortunately*)
- No handling of temporality (*true; persistent issue; often only short-span of data for training*)
- No evaluation details (*common for many industry papers, unfortunately*)
- *In industry, this is often THE algorithm to start with*

- **EASE (Embarassingly Shallow Autoencoders)**

- Surprising Simplicity vs. Performance tradeoff (*general trend of not incorporating NLP stuff blindly; but note the volume of parameters:-)*
 - Linear not enough on complex domains? (*maybe, but sparsity issue; also explicit feedback not very common*)
 - Minimal effect of L2 hyperparam (*diag=0 is the main thing that prevents overfitting*)
- Evaluation issues (pre-processing time, fit into memory)
 - Problems arise with the number of items (*good for NetFlix, bad for large e-commerce*)
- *For me personally, this is THE algorithm to start with if you do not have too many items*

Organization

- **Funny notes:**
 - Not feeling competent to criticize (undertood, but do try it anyway)

Active reading

#2: Hybrid RS (choose one of the following)

- **Knowledge Graph Convolutional Networks for Recommender Systems** [Using knowledge graphs]
 - <https://dl.acm.org/doi/abs/10.1145/3308558.3313417>
- **VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback** [Incorporate raw content]
 - <https://cseweb.ucsd.edu/~jmcauley/pdfs/aaai16.pdf>
- **Ensemble Recommendations via Thompson Sampling: ...** [Intelligent switching hybrid]
 - <https://dl.acm.org/doi/abs/10.1145/3172944.3172967>
- **beeFormer: Bridging the Gap Between Semantic and Interaction Similarity in RS** [content-aware sparse EASE]
 - <https://dl.acm.org/doi/10.1145/3640457.3691707>

Deadline: 31.3.2024

- At most 1 page of text CZ/SK/EN (normal font and margins - be brief but thorough)
- Submit through Study Group Roaster in SIS

Q1. Your name and Paper's title

Q2. What is this paper about, and what contributions does it make?

Q3. What main new insights you received from the paper?

Q4. Does the paper has any notable weaknesses?



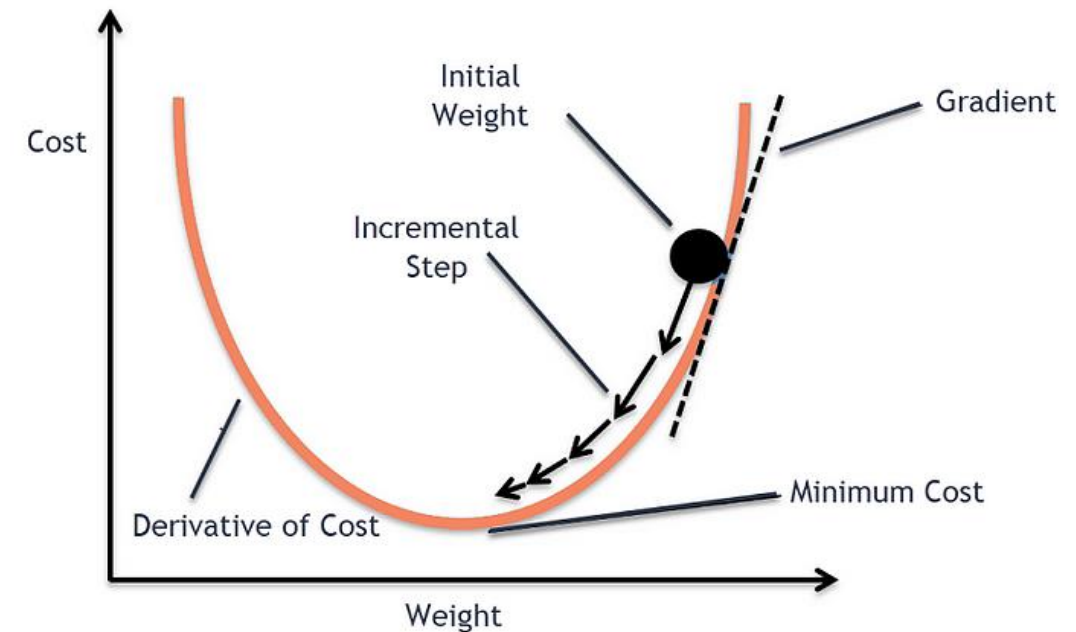
Questions?

Recap

What matrix factorization aims to do?

$$\sum_{i,j} \left(m_{i,j} - \sum_{k=0}^K u_{i,k} v_{k,j} \right)^2 + \lambda \left(\sum_{i,j} u_{i,j}^2 + \sum_{i,j} v_{i,j}^2 \right)$$

How stochastic gradient descend works?

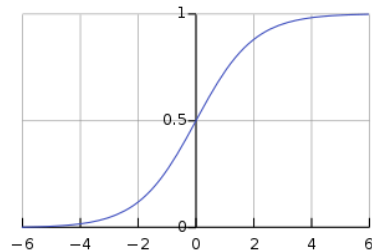


Recap

How BPR differs from „standard“ MF (FunkSVD)?

$$\text{maximize } \underbrace{\sum_{\forall(u,g,b)} \ln \sigma(\hat{r}_{u,g} - \hat{r}_{u,b})}_{\text{Ranking correctness}} - \underbrace{\lambda \|U\|^2 + \|V\|^2}_{\text{Regularization}}.$$

Train set

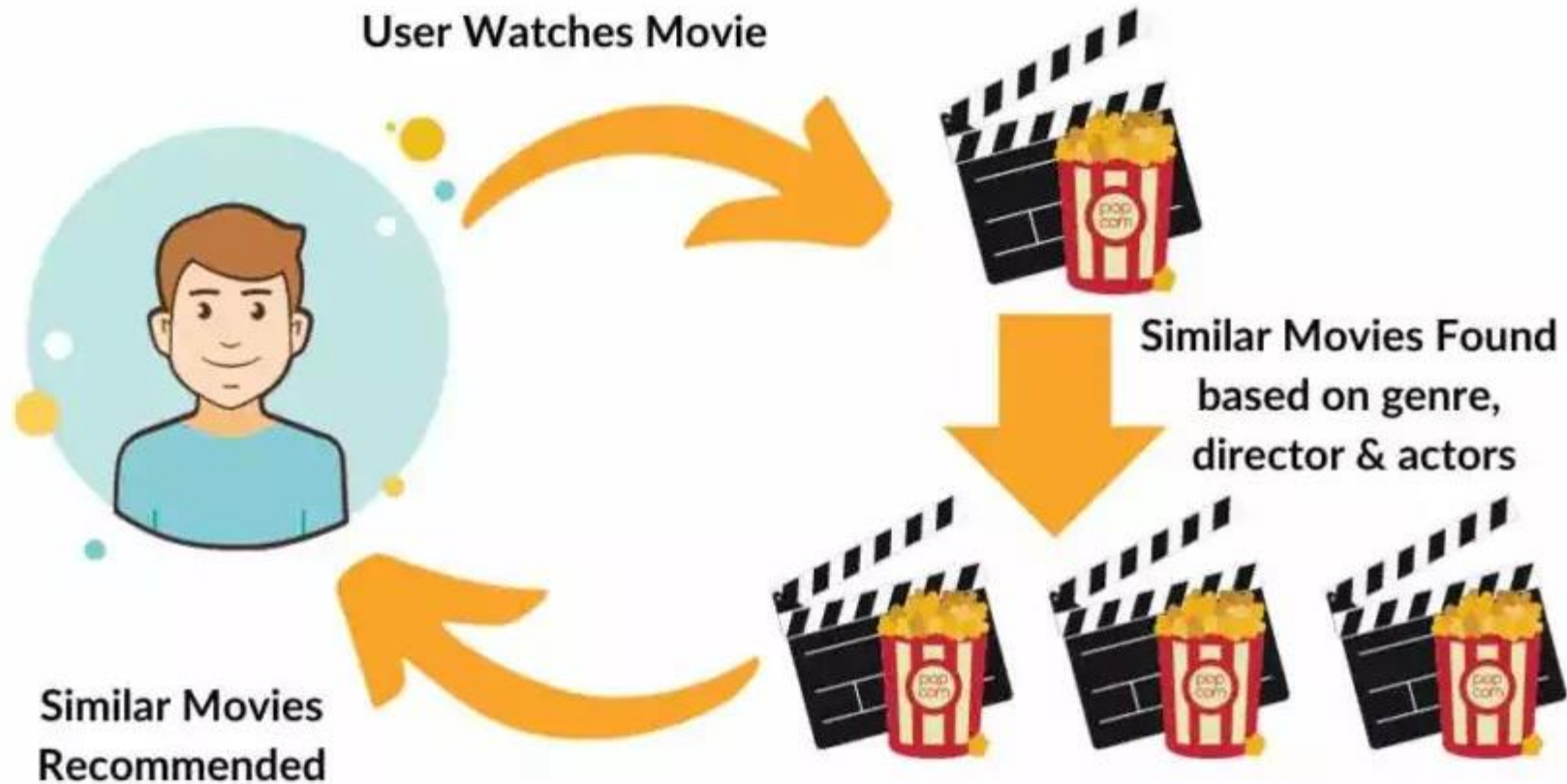


Why do we need user/item baseline predictors? [leftovers]

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$$

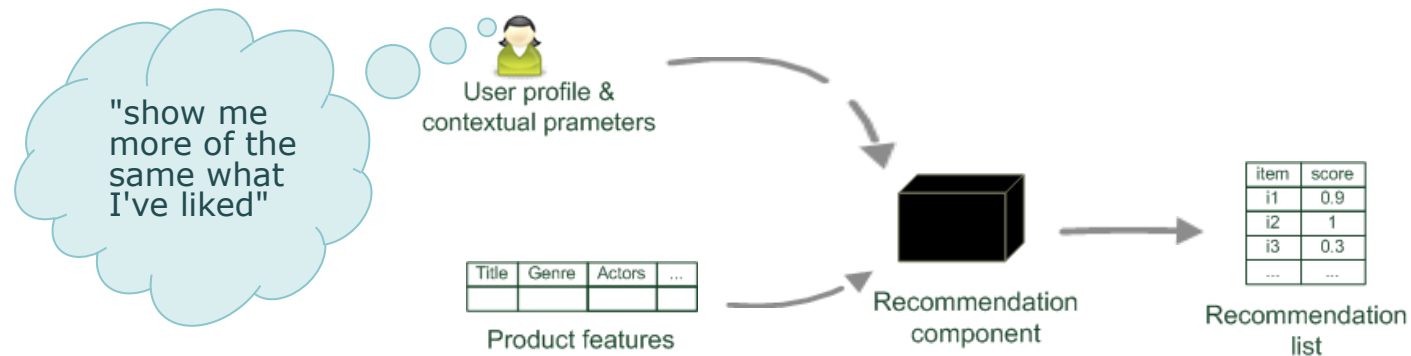
$$\min_{p^*, q^*, b^*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

Content-based recommendation



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



What is the "content"?

- **Most CB-recommendation techniques were applied to recommending text documents.**
 - Like web pages or newsgroup messages for example.
 - Now also multimedia content (fashion, music) or e-commerce
- **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 ↓



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

- Unstructured: free-text description.

What is the "content"?

- Most CB-recommendation techniques were applied to recommend documents.

- Like web pages or newsgroup messages for example
- Now also multimedia content (fashion, music)

- Content of items can also be represented by attributes.

- With textual descriptions of the items.
- Structured: Each item is described by a set of attributes ↓



Title	Genre	Author	Type	Price	Keywords
The Night		David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
	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

- Unstructured: free-text description.

Described method is outdated... But worth to know (baseline & simple yet working approach)

Content representation and item similarities

Item representation

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

User profile

Title	Genre	Author	Type	Price	Keywords
...	Fiction	Brunonia, Barry, Ken Follett	Paperback	25.65	Detective, murder, New York

$keywords(b_j)$
describes Book b_j
with a set of
keywords



Simple approach

- Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
- Or use and combine multiple metrics



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

Jaccard similarity

Term-Frequency - Inverse Document Frequency ($TF - IDF$)

- **Simple keyword representation has its problems**
 - in particular when automatically extracted as
 - 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
 - May not be relevant in some cases (e.g. Male vs. Female attribute on dating sites)

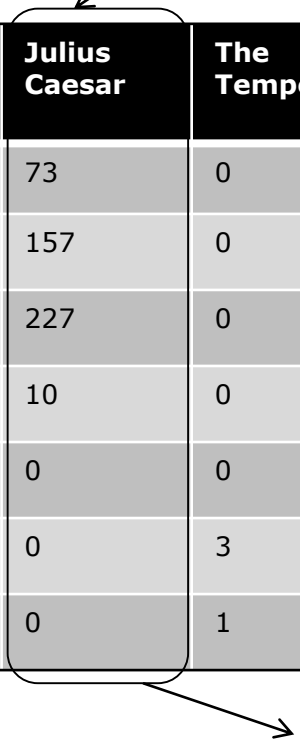
TF-IDF II

- **Given a keyword i and a document j**
- $TF(i, j)$
 - term frequency of keyword i in document 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: $TF-IDF(i, j) = TF(i, j) * IDF(i)$

Example TF-IDF representation

- **Term frequency:**

- Each document is a **count vector** in $\mathbb{N}^{|v|}$



A diagram consisting of a rounded rectangle that encloses the 'Julius Caesar' column of the table. An arrow points from the text 'count vector' in the bullet point above to this rectangle. Another arrow points from the bottom of the rectangle to the text 'Vector v with dimension |v| = 7'.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	1.51	0	3	5	5	1
worser	1.37	0	1	1	1	0

Vector v with dimension $|v| = 7$

Example taken from <http://informationretrieval.org>

Example TF-IDF representation

- Combined TF-IDF weights

- Each document is now represented by a real-valued vector of *TF-IDF* weights $\in \mathbb{R}^{|V|}$

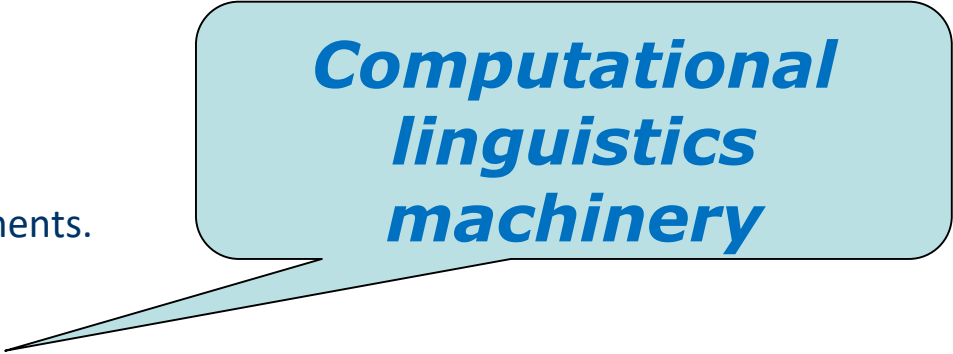
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4					
Caesar	232					
Calpurnia	0					
Cleopatra	57					
mercy	1.5					
worser	1.3					

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Example taken from <http://informationretrieval.org>

Improving the vector space model

- **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" \Rightarrow "go", "stemming" \Rightarrow "stem", ...
- **size cut-offs**
 - only use top n most representative words to remove "noise" from data
 - e.g. use top 100 words



***Computational
linguistics
machinery***

Improving the vector space model II

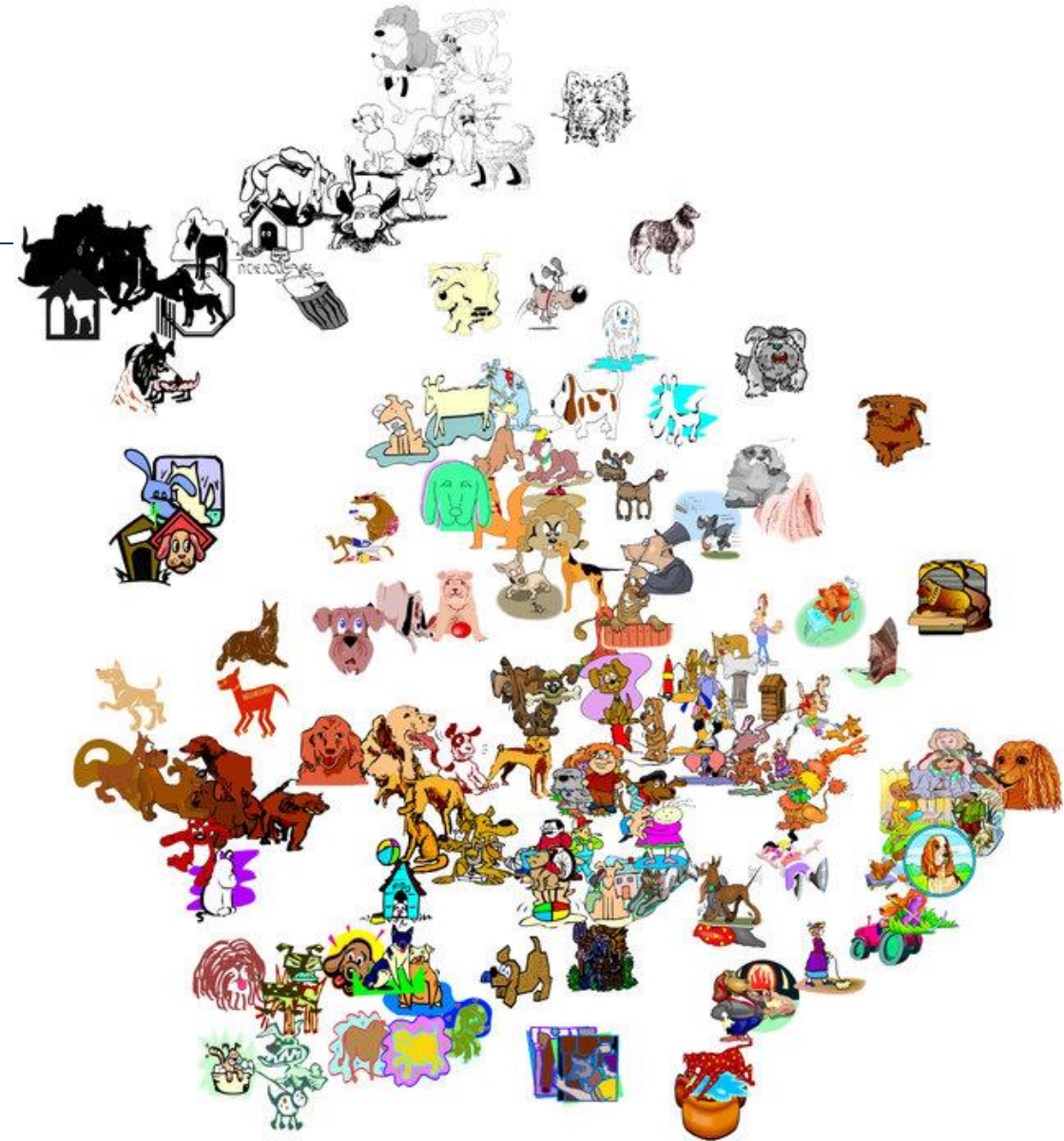
- **Use lexical knowledge, use more elaborate methods for feature selection**
 - Remove words that are not relevant in the domain
 - Remove features that are not relevant for a particular task
- **Detection of phrases as terms**
 - More descriptive for a text than single words
 - e.g. "United Nations,,
 - Named entity recognition (use e.g. Wikipedia)
- **Limitations**
 - semantic meaning remains unknown
 - example: usage of a word in a negative context
 - ➡ ▪ "there is nothing on the menu that a vegetarian would like.."
 - The word "vegetarian" will receive a higher weight then desired
 - an unintended match with a user interested in vegetarian restaurants

**Replaced by 2vec,
BERT, CLIP or similar
nowadays**

Content-based RS

- **What about other modality than text?**
 - Numeric and nominal (categorical) attributes
 - Multimedia (e.g., images)
- **Attributes**
 - Serialize into vectors & utilize the same approaches as for text
 - Nominal attributes are quite natural
 - Numeric attributes -> binning (multiple options), or leave them as is (normalization issues)
 - Usage of fixed weighting such as TF-IDF is questionable
 - (e.g. Gender or Age attributes on dating sites: high importance, but low IDF)
 - State-of-the-art: probably attention layer
- **Multimedia**
 - Embeddings (probably some deep convolutional neural networks or similar architecture)
 - Semantically similar items should have similar embeddings
 - Networks may be either trained to have this feature, or pre-trained models from similar tasks are used

Content-based RS



Multimedia

- **Embeddings** Semantically similar items should have similar embeddings
 - Networks may be either trained to have this feature, or pre-trained models from similar tasks are used

Content-based RS: Multimedia+attribute data, LineIT example

Homepage | SIRET research group | Charles University

LineIT: Lineups assembling Tool

A) Attribute-based filtering

Filter Candidates

- Gender
- Age
 - 0-18 (191)
 - 15-25 (489)
 - 20-30 (689)
 - 25-35 (1026)
 - 30-40 (1249)
 - 35-45 (1434)
 - 40-50 (1372)
 - 45-55 (1062)
 - 50-60 (651)
 - 55-65 (392)
 - 60-70 (191)
 - 65+ (140)
- Nationality
- Body shape
- Hairs
- Beards
- Scars

B) QBE search

Run multi-example, multi-patch query for all checked persons with following parameters:

Local vs. Global ratio: Local search weight

Queried objects

Query examples

Selected ROIs (local search)

C) Search results

Query Add to Lineup Select as Suspect sim:0.87727

Query Add to Lineup Select as Suspect sim:0.87649

Query Add to Lineup Select as Suspect sim:0.87643

Query Add to Lineup Select as Suspect sim:0.87535

Query Add to Lineup Select as Suspect sim:0.87218

D) Current lineup

Members

Query remove

Selected candidates

Suspect

E) Recommended candidates

Add to Lineup Query

Add to Lineup Query

Add to Lineup Query

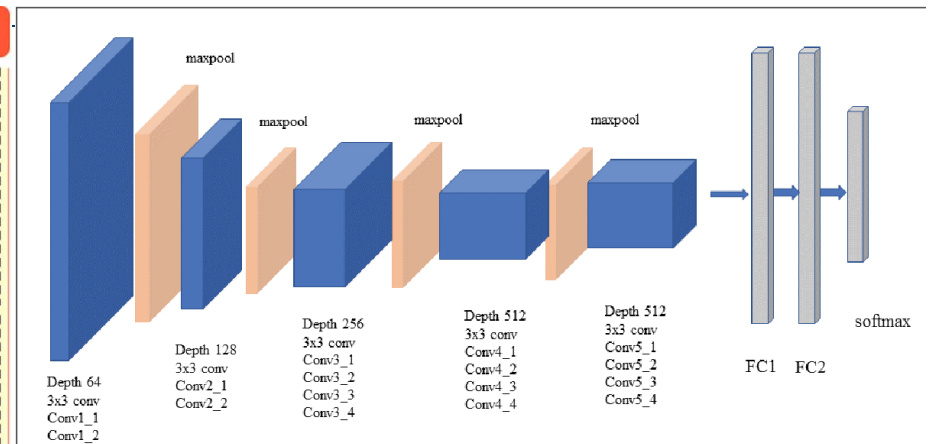
Add to Lineup Query

Add to Lineup Query

Add to Lineup Query

Add to Lineup Query

Add to Lineup Query



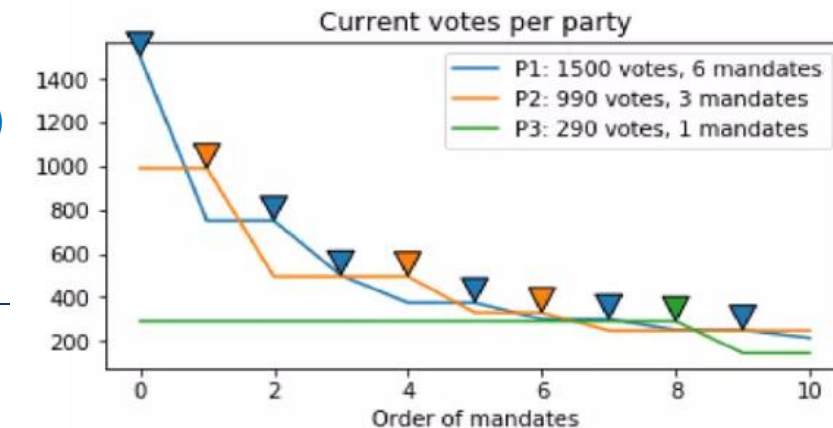
Fuzzy D'Hondt

$$c_{best} = \operatorname{argmax}_{v_{c_i} \in C} \left(\sum_{p_i \in P} v_{curr,i} \times r_{i,j} \right)$$

$$k_i += r_{i,best}, v_{curr,i} = \frac{v_{orig,i}}{k_i + 1}$$

Recommend based on an aggregation of attribute-based and visual similarity

- Combine similarities towards both the suspect and already selected candidates (i.e., lineup uniformity)
- For attribute-based similarity, employ users long-term preference on individual attributes
- Combine both recommenders using Fuzzy interpretation of D'Hondt mandate allocation algorithm



Recommending items

- **Simple method: nearest neighbors**
- **May be relevant for item-based recommendations**
 - Most similar items to the currently viewed one
 - Still used in smaller e-commerce (either based on content or collaborative similarity)
- **Other options?**
 - Any aggregation of user's preferences?

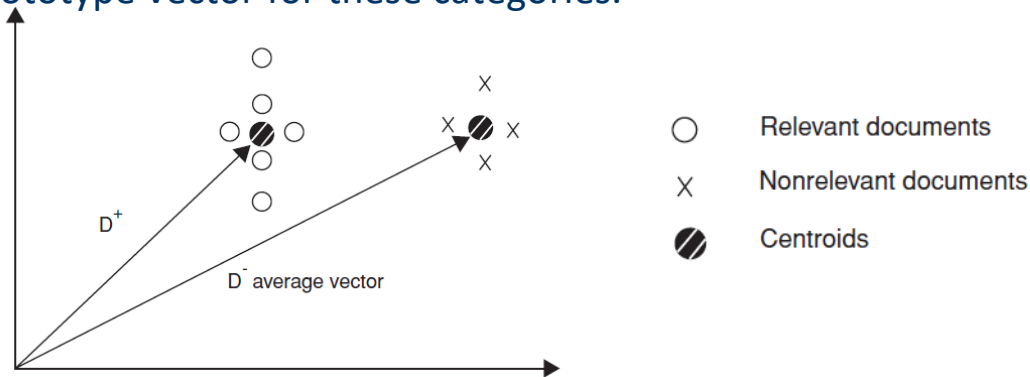
Query-based retrieval: Rocchio's method (Vector Space Model)

- Originally for „conversational“ (interactive/iterative) query retrieval systems
- Query-based retrieval: Rocchio's method
 - The SMART System: 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
- The paradigm fits well also for recommender systems
- Some modern loss functions are based on a similar principles (e.g. Contrastive loss for siamese networks)

Rocchio details

- **Document collections D^+ (liked) and D^- (disliked)**

- Calculate prototype vector for these categories.



- **Computing modified query Q_{i+1} from current query Q_i with:**

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

- **α, β, γ used to fine-tune the feedback**

- α weight for original query
- β weight for positive feedback
- γ weight for negative feedback

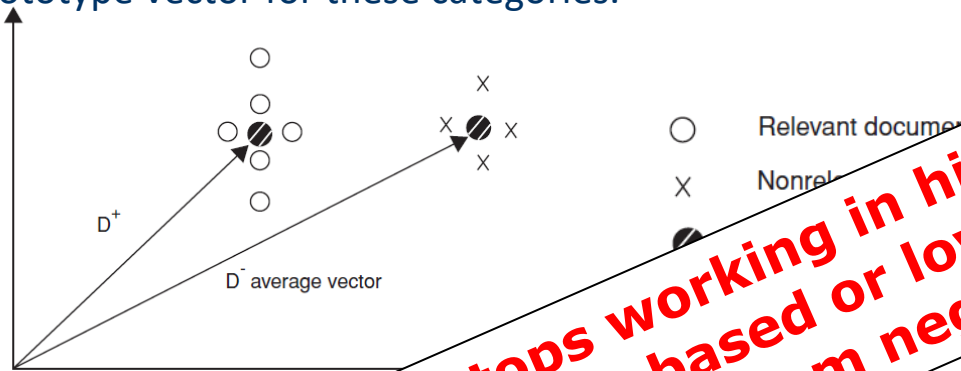
- **Often only positive feedback is used**

- More valuable than negative feedback

Rocchio details

- Document collections D^+ (liked) and D^- (disliked)

- Calculate prototype vector for these categories.



- Computing modified query Q_i from current query Q_i with:

- α, β, γ used to fine-tune the feedback

- α weight for original query
- β weight for positive feedback
- γ weight for negative feedback

$$\left(\sum_{d^+ \in D^+} d^+ \right) - \gamma \left(\frac{1}{|D^-|} \sum_{d^- \in D^-} d^- \right)$$

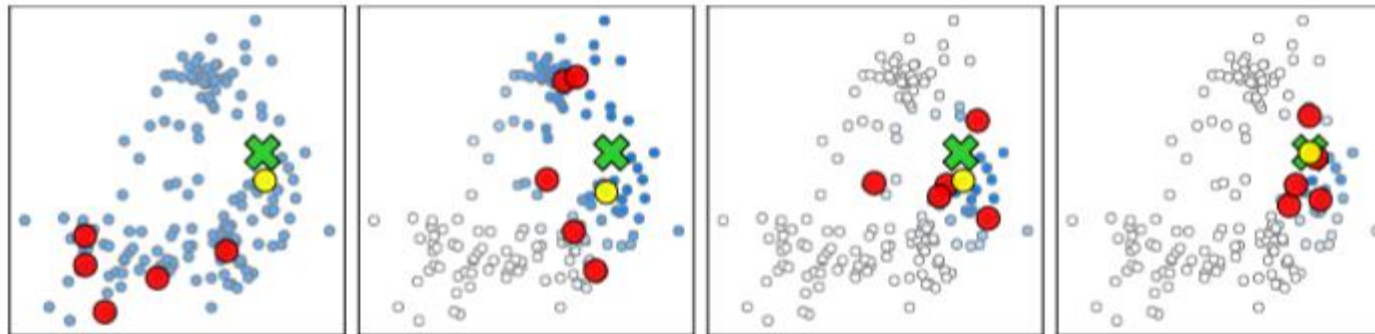
- only positive feedback is used

- More valuable than negative feedback

**This intuition partially stops working in high dimensions
But should be fine e.g. for attribute-based or low-dim representations
Clustering for diverse interests, margin from negative instead of distance**

Bayesian relevance feedback

- **Slight modification: Instead of query prototype evaluate suitability of all points in the solutions space**
 - Counting distances to each positive example separately
 - Exponential transformation (amplify highly similar objects)



$$s'_i = s_i \cdot \prod_{p \in P} \frac{\exp \frac{-\delta_{\cos}(o_i, o_p)}{\sigma}}{\sum_{x \in X \cup \{p\}} \exp \frac{-\delta_{\cos}(o_i, o_x)}{\sigma}} \quad (1)$$

where $o_i = f_{image}(O_i)$ is the vector representation of an image O_i , s and s' are vectors of size $|S|$ representing previous and new score values of images respectively, σ controls the importance of proximity to positive/negative examples, P denotes a list of indexes of positive examples, while X comprise indexes of all implicit negative examples and $\delta_{\cos}(x, y) = 1 - \sigma_{\cos}(x, y)$ is the cosine distance. After the update step, the final scores are specifically normal-

Practical challenges of Rocchio's method

- **Certain number of item ratings needed to build reasonable user model**
 - Can be automated by trying to capture user ratings implicitly (click on document)
 - Pseudorelevance Feedback: Assume that the first n documents match the query best. The set D^- is not used until explicit negative feedback exists.
- **User interaction required during retrieval phase**
 - Interactive query refinement opens new opportunities for gathering information and
 - Helps user to learn which vocabulary should be used to receive the information he needs

Explicit decision models

- **Decision tree for recommendation problems**
 - inner nodes labeled with item features (keywords)
 - used to partition the test examples
 - **existence or non existence of a keyword**
 - in basic setting only two classes appear at leaf nodes
 - **interesting or not interesting**
 - decision tree can automatically be constructed from training data
 - works best with small number of features
 - use meta features like author name, genre, ... instead of TF-IDF representation.

Never saw them really working...

Explicit decision models II

- **Rule induction**
 - built on RIPPER algorithm
 - good performance compared with other classification methods
 - **elaborate postpruning techniques of RIPPER**
 - **extension for e-mail classification**
 - takes document structure into account
- **main advantages of these decision models:**
 - inferred decision rules serve as basis for generating explanations for recommendation
 - existing domain knowledge can be incorporated in models

Never saw them really working...
But may be fine for possible results pre-processing

On feature selection

- **process of choosing a subset of available terms**
- **different strategies exist for deciding which features to use**
 - feature selection based on domain knowledge and lexical information from WordNet (Pazzani and Billsus 1997)
 - frequency-based feature selection to remove words appearing "too rare" or "too often" (Chakrabarti 2002)
- **Not appropriate for larger text corpora**
 - Better to
 - evaluate value of individual features (keywords) independently and
 - construct a ranked list of "good" keywords.
- **Typical measure for determining utility of keywords: e.g. X^2 , mutual information measure or Fisher's discrimination index**

Still important even today (less garbage to crawl for deep learning models)

- Think on the meta-level: can user preference be based on this feature?

Limitations of content-based recommendation 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)
 - Not so big issue today
- **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
 - Multicriterial optimization (diversity, novelty), fairness-aware approaches

Overspecialization

Recommended reading:



What is wrong here?

Overspecialization

Recommended reading:



Possibly lacking:

- Novelty (w.r.t. user's currently known interests)
- Diversity (of recommended items from each other)
- Serendipity (i.e. combination of relevance & surprise for the user)

Overspecialization

Recommended reading:



Simple diversity re-ranking: maximal marginal relevance

- Select first item by relevance
- Select next item by $\max(\alpha \cdot \text{relevance} - (1 - \alpha) \cdot \max(\text{similarity to already selected item}))$

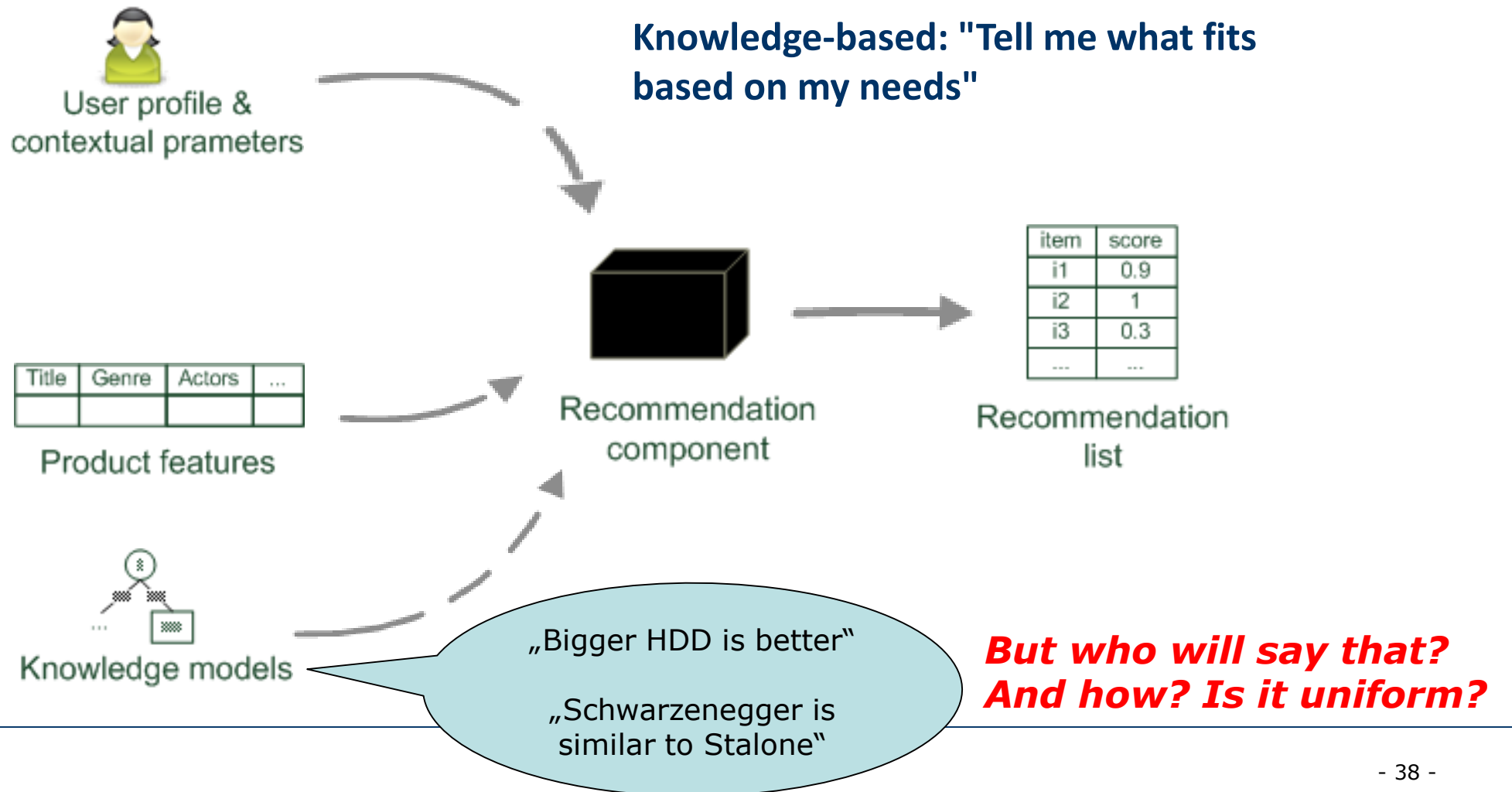
Not good for too long lists (iterative calculations over increasingly large sets)
– or have similarity matrix ready (might be too large to store)

Discussion & summary

- In contrast to collaborative approaches, content-based techniques **do not require user community** in order to work
- Presented approaches aim to learn a model of user's interest preferences based on explicit or implicit feedback
 - Deriving implicit feedback from user behavior can be problematic
- Evaluations show that a good recommendation accuracy can be achieved with help of machine learning techniques
 - These techniques do not require a user community
- **Danger exists that recommendation lists contain too many similar items**
 - All learning techniques require a certain amount of training data
 - Some learning methods tend to overfit the training data
- Pure content-based systems are rarely found in commercial environments
 - **Well, for small-scale projects (cold-start) they'll do...**
 - **Better than building collaborative systems without any users so far**

Knowledge-based recommendation

Basic I/O Relationship



Why do we need knowledge-based recommendation?

- Products with low number of available ratings



- Time span plays an important role
 - five-year-old ratings for computers
 - user lifestyle or family situation changes
- **Customers want to define their requirements explicitly**
 - "the color of the car should be black"

Knowledge-based recommender systems

- **Constraint-based**
 - based on explicitly defined set of recommendation rules
 - *(partially)* fulfill recommendation rules
- **Case-based (critiquing)**
 - Item-based: give me similar items, however with larger display
- **Both approaches are similar in their **conversational** recommendation proces** *(edge of query retrieval and recommender systems)*
 - users specify the requirements
 - systems try to identify solutions
 - if no solution can be found, users change requirements (or partial solution is given)
 - *Not always, we may learn knowledge RS rules from collaborative data*

Constraint-based recommender systems

- **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 set of applicable constraints
- applicability of constraints depends on current user model
- explanations – transparent line of reasoning

Constraint-based recommendation tasks

- **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
- **Find a subset of items that satisfy the maximum set of weighted constraints**
 - similar to find a maximally succeeding subquery (XSS)
 - all proposed items have to fulfill the same set of constraints
 - compute relaxations based on predetermined weights
- **Rank items according to weights of satisfied soft constraints**
 - rank items based on the ratio of fulfilled constraints
 - does not require additional ranking scheme

Ranking the items

- **Multi-attribute utility theory**
 - each item is evaluated according to a predefined set of dimensions that provide an aggregated view on the basic item properties
- ***E.g. quality and economy are dimensions in the domain of digital cameras***

id	value	quality	economy
price	≤250	5	10
	>250	10	5
mpix	≤8	4	10
	>8	10	6
opt-zoom	≤9	6	9
	>9	10	6
LCD-size	≤2.7	6	10
	>2.7	9	5
movies	Yes	10	7
	no	3	10
sound	Yes	10	8
	no	7	10
waterproof	Yes	10	6
	no	8	10

Item utility for customers

■ Customer specific interest

Customer	quality	economy
Cu ₁	80%	20%
Cu ₂	40%	60%

■ *Calculation of Utility*

quality	economy	cu ₁	cu ₂
P ₁ $\Sigma(5,4,6,6,3,7,10) = 41$	$\Sigma(10,10,9,10,10,10,6) = 65$	45.8 [8]	55.4 [6]
P ₂ $\Sigma(5,4,6,6,10,10,8) = 49$	$\Sigma(10,10,9,10,7,8,10) = 64$	52.0 [7]	58.0 [1]
P ₃ $\Sigma(5,4,10,6,10,10,8) = 53$	$\Sigma(10,10,6,10,7,8,10) = 61$	54.6 [5]	57.8 [2]
P ₄ $\Sigma(5,10,10,6,10,7,10) = 58$	$\Sigma(10,6,6,10,7,10,6) = 55$	57.4 [4]	56.2 [4]
P ₅ $\Sigma(5,4,6,10,10,10,8) = 53$	$\Sigma(10,10,9,6,7,8,10) = 60$	54.4 [6]	57.2 [3]
P ₆ $\Sigma(5,10,6,9,10,10,8) = 58$	$\Sigma(10,6,9,5,7,8,10) = 55$	57.4 [3]	56.2 [5]
P ₇ $\Sigma(10,10,6,9,10,10,8) = 63$	$\Sigma(5,6,9,5,7,8,10) = 50$	60.4 [2]	55.2 [7]
P ₈ $\Sigma(10,10,10,9,10,10,10) = 69$	$\Sigma(5,6,6,5,7,8,6) = 43$	63.8 [1]	53.4 [8]

Explicit constraint-based GUI (Bronislav Vaclav, Master thesis 2011)

Categories | **Parameters** | **Notebooks** » **New Search** | Cancel Search

EXISTING CONDITIONS




Parameter	Condition	Value	Condition Weight
Price	is less than	12000	High priority
RAM size	is greater than	2048 MB	Medium priority
CPU frequency	is greater than	2.2 GHz	Medium priority
Manufacturer	is	ACER or ASUS or FUJITSU	Low priority
LCD size	is between	16 and 19 inches	Low priority




[Show Results](#)

RESULTS

Sort by: Rating | Show: Detailed, 9

1 2 3 4 5 6 7 8 ... Next

Notebook 115	Notebook 110	Notebook 240
 <p>Always on the move? Pick our compact laptop. We also offer a 17.3" model if you want more room, or think you might replace your desktop computer. And if you need...</p> <ul style="list-style-type: none">RAM size: 2048 MBCPU frequency: 2 GHzManufacturer: ASUSLCD size: 15.4 inchesProcessor: Core 2 Duo T7250 <p>• 12314.4,- incl. VAT 14654.14,- In Stock</p>	 <p>Always on the move? Pick our compact laptop. We also offer a 17.3" model if you want more room, or think you might replace your desktop computer. And if you need...</p> <ul style="list-style-type: none">RAM size: 2048 MBCPU frequency: 2 GHzManufacturer: ASUSLCD size: 15.4 inchesProcessor: Core 2 Duo T7250 <p>• 12314.5,- incl. VAT 14654.26,- In Stock</p>	 <p>If you're into high-def movies, games, music and photography, this laptop is for you. New Windows gives you more ways than ever of savoring your digital...</p> <ul style="list-style-type: none">RAM size: 4096 MBCPU frequency: 2 GHzManufacturer: ASUSLCD size: 15.4 inchesProcessor: Core 2 Duo T7250 <p>• 13115.6,- incl. VAT 15607.56,- In Stock</p>

Notebook 61	Notebook 167	Notebook 229
 <p>If you're into high-def movies, games, music and photography, this laptop is for you. New Windows gives you more ways than ever of savoring your digital...</p> <ul style="list-style-type: none">RAM size: 2048 MBCPU frequency: 2.2 GHz <p>• 12314.4,- incl. VAT 14654.14,- In Stock</p>	 <p>Always on the move? Pick our compact laptop. We also offer a 17.3" model if you want more room, or think you might replace your desktop computer. And if you need...</p> <ul style="list-style-type: none">RAM size: 2048 MBCPU frequency: 2 GHz <p>• 12314.5,- incl. VAT 14654.26,- In Stock</p>	 <p>Always on the move? Pick our compact laptop. We also offer a 17.3" model if you want more room, or think you might replace your desktop computer. And if you need...</p> <ul style="list-style-type: none">RAM size: 4096 MBCPU frequency: 2.2 GHz <p>• 13115.6,- incl. VAT 15607.56,- In Stock</p>

CREATE / UPDATE CONDITIONS

OBJECT RELEVANCE

Case-based recommender systems

- Items are retrieved using similarity measures
- Distance similarity

$$\text{similarity}(p, REQ) = \frac{\sum_{r \in REQ} w_r * \text{sim}(p, r)}{\sum_{r \in REQ} w_r}$$



- **Def.**
 - $\text{sim}(p, r)$ expresses for each item attribute value $\phi_r(p)$ its distance to the customer requirement $r \in REQ$.
 - w_r is the importance weight for requirement r
- **In real world, customer would like to**
 - maximize certain properties. i.e. resolution of a camera, "more is better"(MIB)
 - minimize certain properties. i.e. price of a camera, "less is better"(LIB)
 - Target within some values, e.g. Price between x,y

Knowledge-based recommender systems (work of Alan Eckhardt circa 2008-2014)

- Transform known rating on items into
 - Rating (preference regression) of item features



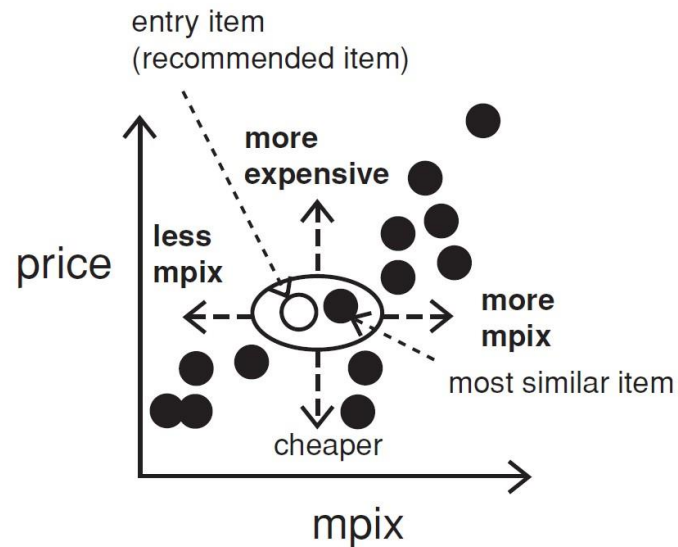
- Learning combination of item feature's ratings
 - Based on goodness of fit on features

$$\mathbb{Q}(o) = \frac{(2 * f_{Price}(o) + 1 * f_{Display}(o) + 1 * f_{RAM}(o))}{4}$$

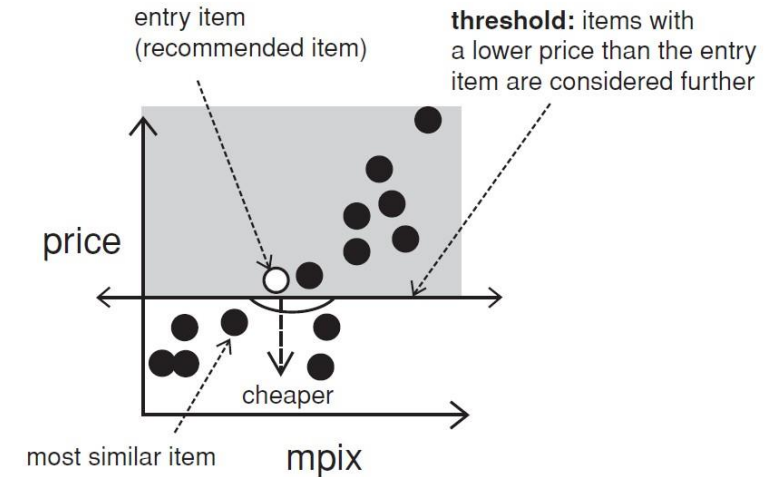
- Evaluate the learned rating function on all other objects
 - *Recommend better instead of similar objects*

Interacting with case-based recommenders (perhaps the key thing to remember)

- Customers maybe not know what they are seeking
- Critiquing is an effective way to support such navigations
- Customers specify their change requests (*price or mpix*) that are not satisfied by the current item (*entry item*)

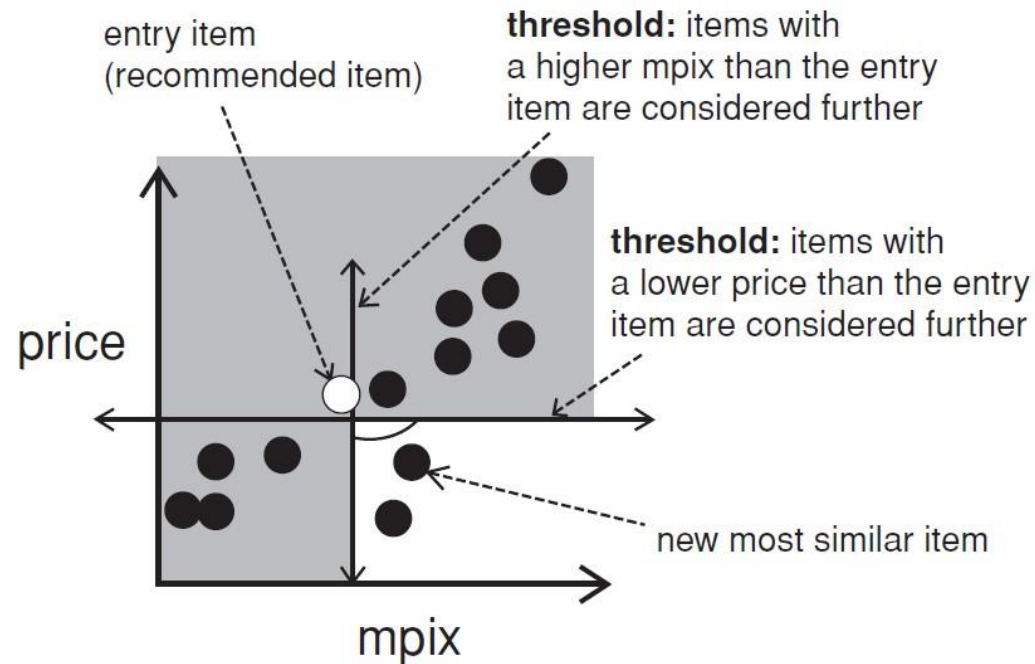


Critique on price



Compound critiques

- Operate over multiple properties can improve the efficiency of recommendation dialogs
 - You can try to learn attribute-level preferences from the interaction data (if you have them), or apply general policies (item A is better than item B for most settings)



Summary

- **Knowledge-based recommender systems**
 - Move from recommending *similar* to recommending *better* objects
- **Limitations**
 - cost of knowledge acquisition
 - from domain experts / users / external resources
 - accuracy of preference models
 - very fine granular preference models require many interaction cycles
 - collaborative filtering models preference implicitly
 - independence assumption can be challenged
 - preferences are not always independent from each other
 - *No known commercial usage*
 - *Experiments with LLM + text input goes in this direction*
 - *The generic concept of price per value with optional personalization is viable*