Recommender Systems: Knowledge-Based Recommendation

Topic 6

Knowledge-Based Recommendation

- Some product recommendations require substantive domain knowledge more than item features or user ratings, especially in the context of products such as cars, computers, apartments, or financial services.
- Knowledge Sources vs. Recommendation Types
- No ramp-up problems exist for knowledge-based recommendation systems.
- A KBR system does not have to gather information about a particular user because its judgments are independent of individual tastes.
- Knowledge-based recommendation process is highly interactive ("conversational systems" as opposed to "information filtering systems").

Two Types of KBR Systems

- Constraint-based recommendation systems
  - Use explicitly defined set of recommendation rules to identify a set of items that fulfill the requirements
- Case-based recommendation systems
  - Use similarity metrics to retrieve items that are similar (within a predefined threshold) to the specified customer requirements.
- Both systems are similar in terms of the recommendation process:
  - the user must specify the requirements, and the system tries to identify a solution.
  - If no solution found, the user must change the requirements.
  - The system may also provide explanations for the recommended items.

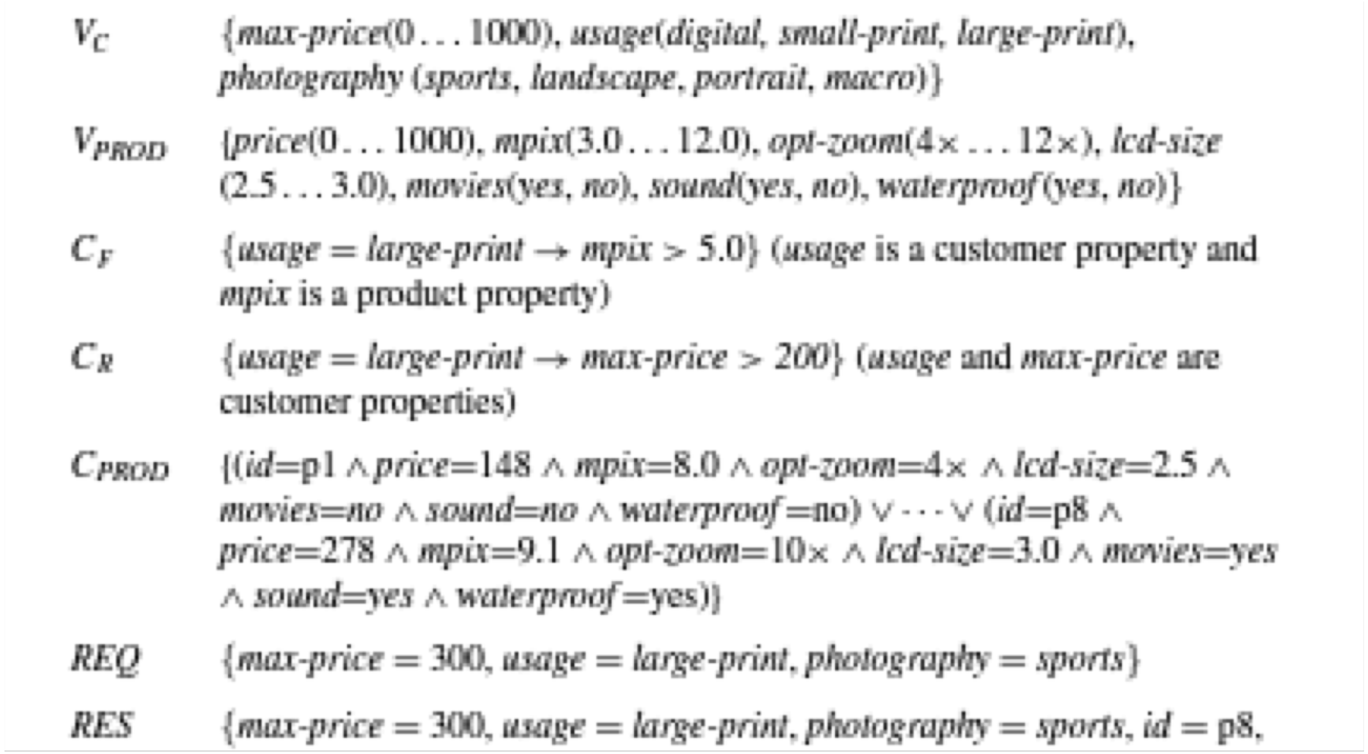
Example Product Assortment: Digital Cameras (Table 4.1)

ID	Price	Mpix	Opt-zoom	LCD-size	Movies	Sound	Water-proof
P1	148	8.0	4x	2.5	No	No	Yes
P2	182	8.0	5x	2.7	Yes	Yes	No
P3	189	8.0	10x	2.5	Yes	Yes	No
P4	196	10.0	12x	2.7	Yes	No	Yes
P5	151	7.1	3x	3.0	Yes	Yes	No
P6	199	9.0	3x	3.0	Yes	Yes	No
P7	259	10.0	3x	3.0	Yes	Yes	No
P8	278	9.1	10x	3.0	Yes	Yes	yes

Constraints

- A constraint-based recommendation problem can be represented as a constraint satisfaction problem that can be solved by a constraint solver or a conjunctive query.
- A constraint satisfaction problem (CSP) can be described by a triple (*V*, *D*, C) where
  - V is a set of variables
  - D is a set of finite domains for these variables, and
  - C is a set of constraints that describes the combinations of values the variables can simultaneously take.
- A solution to a CSP corresponds to an assignment of a value to each variable in V in a way that all constraints are satisfied.

CSP Example (Table 4.2)



V_C	{*max-price*(0 . . . 1000), *usage*(*digital*, *small-print*, *large-print*), *photography* (*sports*, *landscape*, *portrait*, *macro*)}
V_{PROD}	{*price*(0 . . . 1000), *mpix*(3.0 . . . 12.0), *opt-zoom*(4× . . . 12×), *lcd-size* (2.5 . . . 3.0), *movies*(*yes*, *no*), *sound*(*yes*, *no*), *waterproof*(*yes*, *no*)}
C_F	{*usage* = *large-print* → *mpix* > 5.0} (*usage* is a customer property and *mpix* is a product property)
C_R	{*usage* = *large-print* → *max-price* > *200*} (*usage* and *max-price* are customer properties)
C_{PROD}	{(*id*=p1 ∧ *price*=148 ∧ *mpix*=8.0 ∧ *opt-zoom*=4× ∧ *lcd-size*=2.5 ∧ *movies*=*no* ∧ *sound*=*no* ∧ *waterproof*=no) ∨ · · · ∨ (*id*=p8 ∧ *price*=278 ∧ *mpix*=9.1 ∧ *opt-zoom*=10× ∧ *lcd-size*=3.0 ∧ *movies*=*yes* ∧ *sound*=*yes* ∧ *waterproof*=*yes*)}
REQ	{*max-price* = 300, *usage* = *large-print*, *photography* = *sports*}
RES	{*max-price* = 300, *usage* = *large-print*, *photography* = *sports*, *id* = p8,

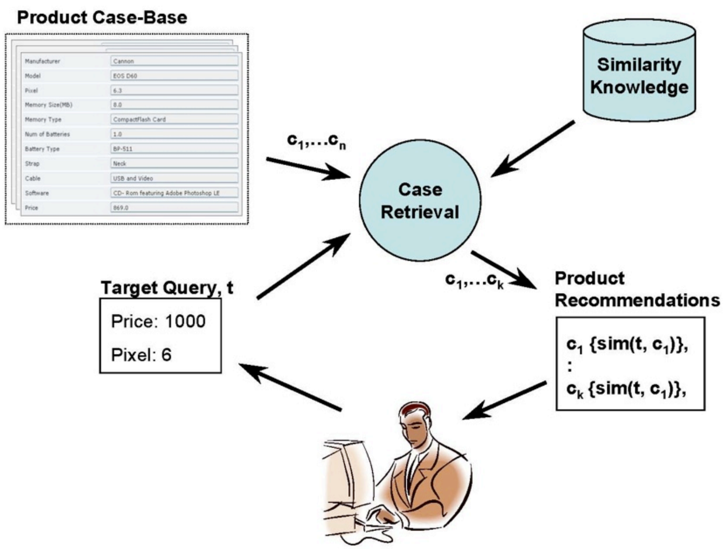
- Customer properties (V_C), Product properties (V_{PROD}), Compatibility constraints (C_R), Filter conditions (C_F), Product constraints (C_{PROD}).
- Recommendation Task: Find a set of solutions to CSP ($V = V_C \cup V_{PROD}$, D, $C = C_R \cup C_F \cup C_{PROD} \cup REQ$)

Conjunctive Queries

- A conjunctive query is a database query with a set of selection criteria that are connected conjunctively.
- Constraint-based item retrieval can be seen as a matter of constructing a conjunctive database query that is executed against the item catalog.
- Queries can be defined that select different item subsets from the set of available items depending on the requirements in REQ.
- SQL Example

Cases

- Case-based recommendation systems have their origins in case-based reasoning CBR) techniques.
- Items or products are represented as cases and recommendations are generated by retrieving those cases that are most similar to a user’s query or profile.



Manufacturer	Cannon
Model	EOS D60
Pixel	6.3
Memory Size(MB)	8.0
Memory Type	CompactFlash Card
Num of Batteries	1.0
Battery Type	BP-511
Strap	Neck
Cable	USB and Video
Software	CD- Rom featuring Adobe Photoshop LE
Price	869.0

Similarities

- Case-based recommendation systems mostly exploit similarity metrics for the retrieval of items from a catalog.
- In general, similarity metrics describe to which extent item properties match some given user’s requirements.
- Similarity is NOT a simple or uniform concept. What counts similar depends on what one’s goals are.

Similarities (Continued)

- The similarity of a given case C to a target query Q is typically defined as:

$$Sim(C, Q) = \frac{\sum_{a \in A_Q} w_a \, sim_a(C, Q)}{\sum_{a \in A_Q} w_a}$$

where for each $a \in A$, w_a is a numeric weight representing the importance of a and sim_a(C, Q) is a local measure of the similarity of π_a(C), the value of a in C, to π_a(Q), the preferred value of a.

Similarities (Continued)

- For MIB (more is better) attributes

$$sim_a(C,Q) = \frac{\pi_a(C) - \min(a)}{\max(a) - \min(a)}$$

- For LIB (less is better) attributes

$$sim_a(C,Q) = \frac{\max(a) - \pi_a(C)}{\max(a) - \min(a)}$$

- For NIB (nearer is better) attributes

$$sim_a(C,Q) = 1 - \frac{|\pi_a(C) - \pi_a(Q)|}{\max(a) - \min(a)}$$

Interacting with Constraint-Based Recommenders

- The general interaction flow consists of
  - The user specifies his or her initial preferences
  - When enough information about the user’s requirements and preferences has been collected, the user is presented with a set of matching items.
  - The user might revise his or her requirements to see alternative solutions or narrow down the number of matching items.
  - If none of the items in the catalog satisfies all user requirements, the recommender should support the user in resolving the problems.

Defaults

- Important means to support customers in the requirements specification process.
- Can be abused to manipulate consumers to choose certain options.
- Proposing default values
  - Static defaults
  - Dependent defaults
  - Derived defaults
    - 1-nearest neighbor
    - Weighted majority voter

Defaults

- Selecting the next question
  - User interaction logs are needed for the selection of questions.

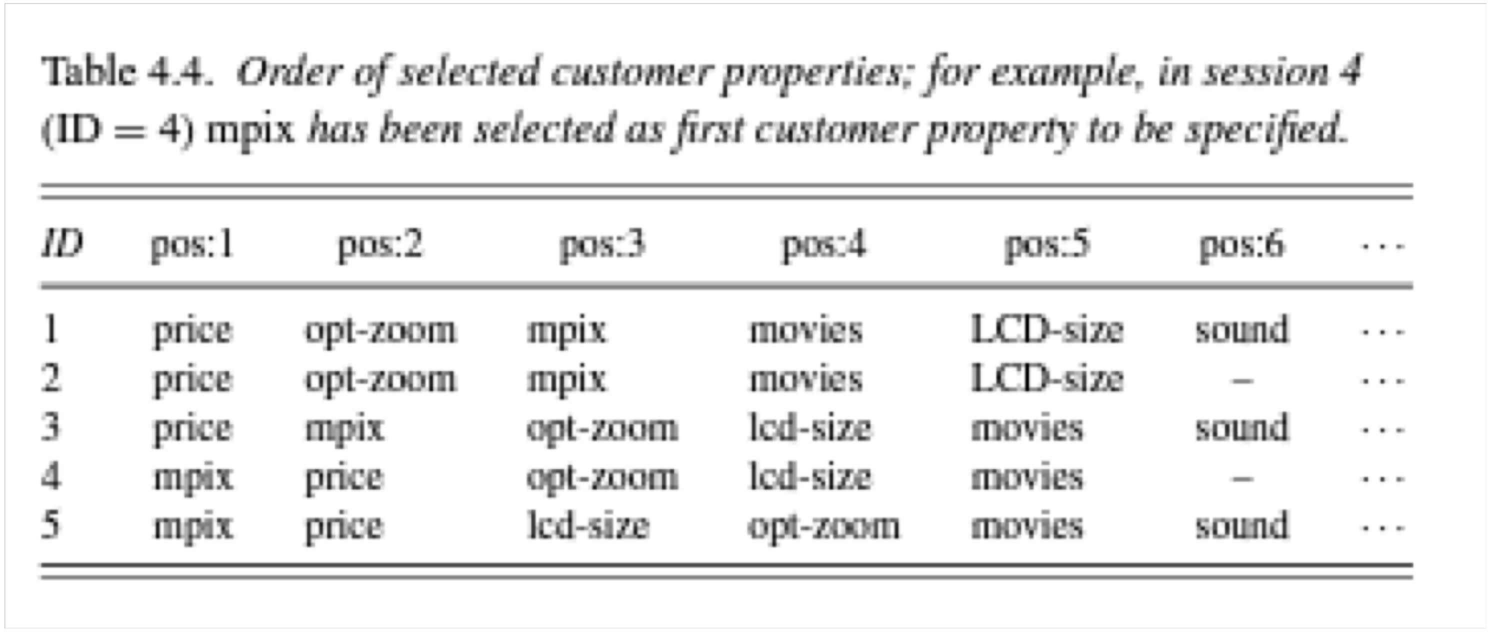


Table 4.4. *Order of selected customer properties; for example, in session 4* (ID = 4) mpix *has been selected as first customer property to be specified.*

ID	pos:1	pos:2	pos:3	pos:4	pos:5	pos:6	···
1	price	opt-zoom	mpix	movies	LCD-size	sound	···
2	price	opt-zoom	mpix	movies	LCD-size	–	···
3	price	mpix	opt-zoom	lcd-size	movies	sound	···
4	mpix	price	opt-zoom	lcd-size	movies	–	···
5	mpix	price	lcd-size	opt-zoom	movies	sound	···

Dealing with Unsatisfiable Requirements and Empty Result Sets

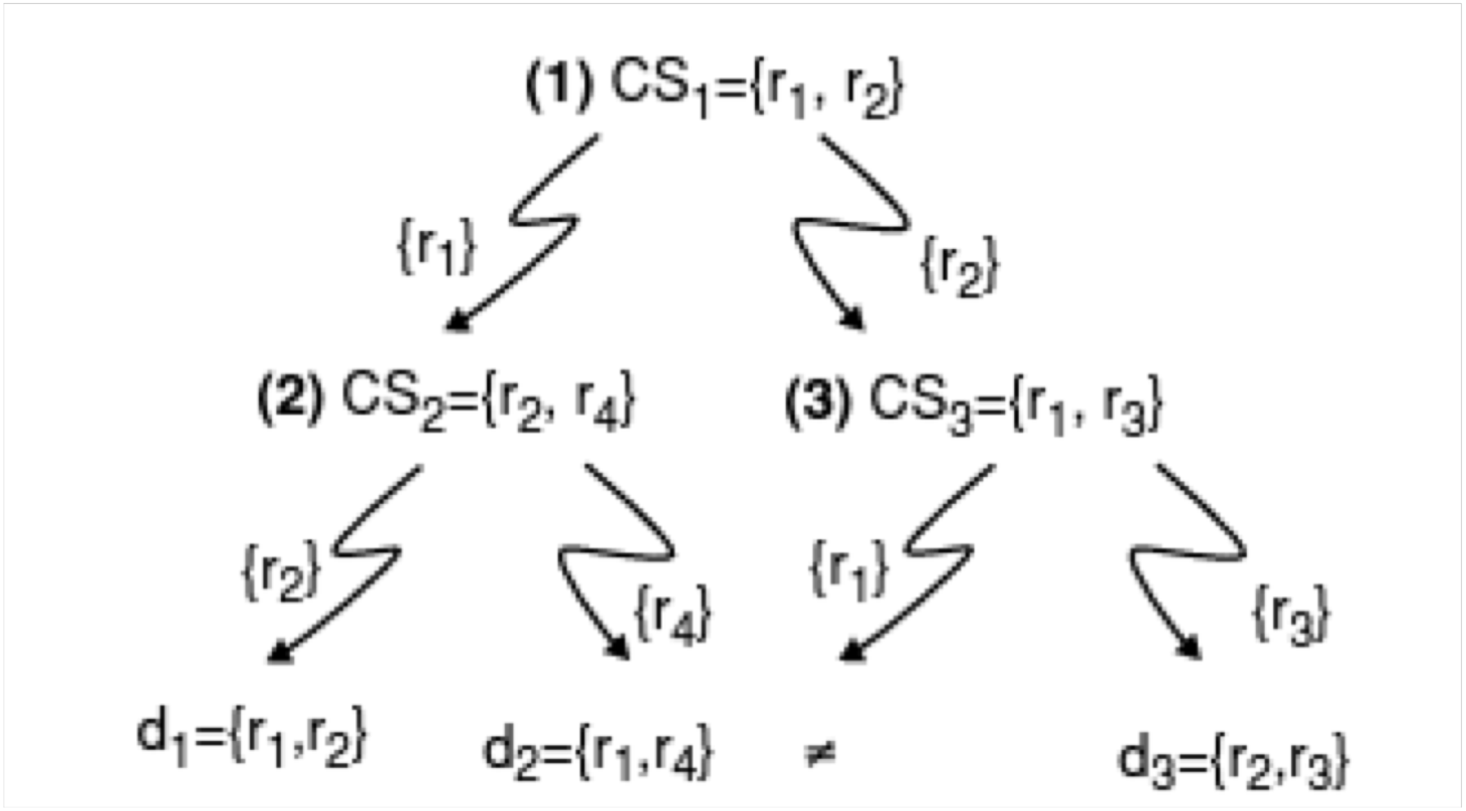
If REQ = {r_1: price <= 150, r_2: Opt-zoom = 5x, r_3: sound = yes, r_4: waterproof = yes}, it cannot be fulfilled by any product.

ID	Price	Mpix	Opt-zoom	LCD-size	Movies	Sound	Water-proof
P1	148	8.0	4x	2.5	No	No	Yes
P2	182	8.0	5x	2.7	Yes	Yes	No
P3	189	8.0	10x	2.5	Yes	Yes	No
P4	196	10.0	12x	2.7	Yes	No	Yes
P5	151	7.1	3x	3.0	Yes	Yes	No
P6	199	9.0	3x	3.0	Yes	Yes	No
P7	259	10.0	3x	3.0	Yes	Yes	No
P8	278	9.1	10x	3.0	Yes	Yes	yes

Dealing with Unsatisfiable Requirements and Empty Result Sets

- One approach of dealing with unsatisfiable requirements is to ask users to provide the indication of a minimal set of requirements.
- In this context, a diagnosis is a minimal set of user requirements whose repair (adaptation) will allow the retrieval of a recommendation.
- For REQ = {*r1*: price <= 150, *r2*: Opt-zoom = 5x, *r3*: sound = yes, *r4*: waterproof = yes}, the conflict sets are $CS_1 = \{r_1, r_2)$, $CS_2 = \{r_2, r_4\}$, and $CS_3 = \{r_1, r_3\}$.

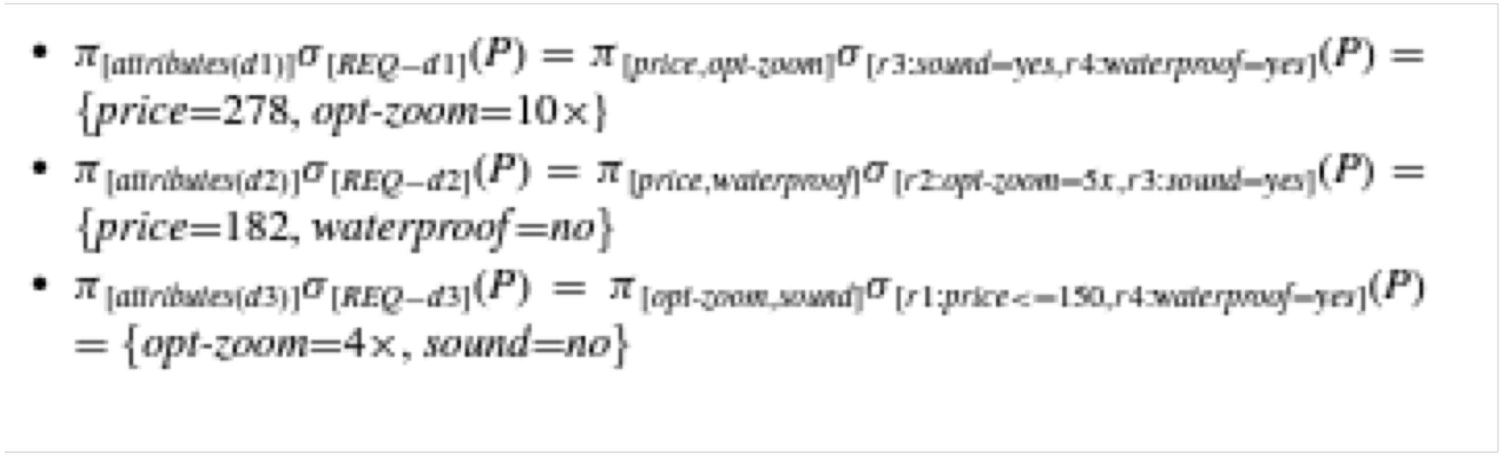
Dealing with Unsatisfiable Requirements and Empty Result Sets



$\Delta = \{d_1{:}\{r_1, r_2\}, d_2{:}\{r_1, r_4\}, d_3{:}\{r_2, r_3\}\}$

Proposing Repairs

- After having identified the set of possible diagnoses (Δ), we must propose repair actions for each of those diagnoses.
- Alternative repair actions can be derived by querying the product table *P* with $\pi_{[attributes(d)]}\sigma_{[REQ-d]}(P)$.



- $\pi_{[attributes(d1)]}\sigma_{[REQ-d1]}(P) = \pi_{[price,opt\text{-}zoom]}\sigma_{[r3:sound=yes,r4:waterproof=yes]}(P) = \{price=278, opt\text{-}zoom=10\times\}$
- $\pi_{[attributes(d2)]}\sigma_{[REQ-d2]}(P) = \pi_{[price,waterproof]}\sigma_{[r2:opt\text{-}zoom=5x,r3:sound=yes]}(P) = \{price=182, waterproof=no\}$
- $\pi_{[attributes(d3)]}\sigma_{[REQ-d3]}(P) = \pi_{[opt\text{-}zoom,sound]}\sigma_{[r1:price<=150,r4:waterproof=yes]}(P) = \{opt\text{-}zoom=4\times, sound=no\}$

Repair Alternatives

Repair	Price	Opt-zoom	Sound	Waterproof
Rep1	278	10x	✓	✓
Rep2	182	✓	✓	No
Rep3	✓	4x	No	✓

Ranking the Items: Utility-Based Recommendation

- Primacy effects
- Multi-attribute utility theory (MAUT)
  - The theory provides a systematic way to handle the tradeoffs among multiple objectives
  - The customer-specific item utility is calculated on the basis of the following formula, in which the index *j* iterates over the number of predefined dimensions, *interest*(*j*) denotes a user's interest in dimension *j*, and *contributions*(p, *j*) denotes the contribution of item *p* to the interest dimension *j*.

$$Utilitity(p) = \sum_{j=1}^{\#(\dim ensions)} \text{int}\, erest(j) * contribution(p, j)$$

Ranking the Items: Utility-Based Recommendation

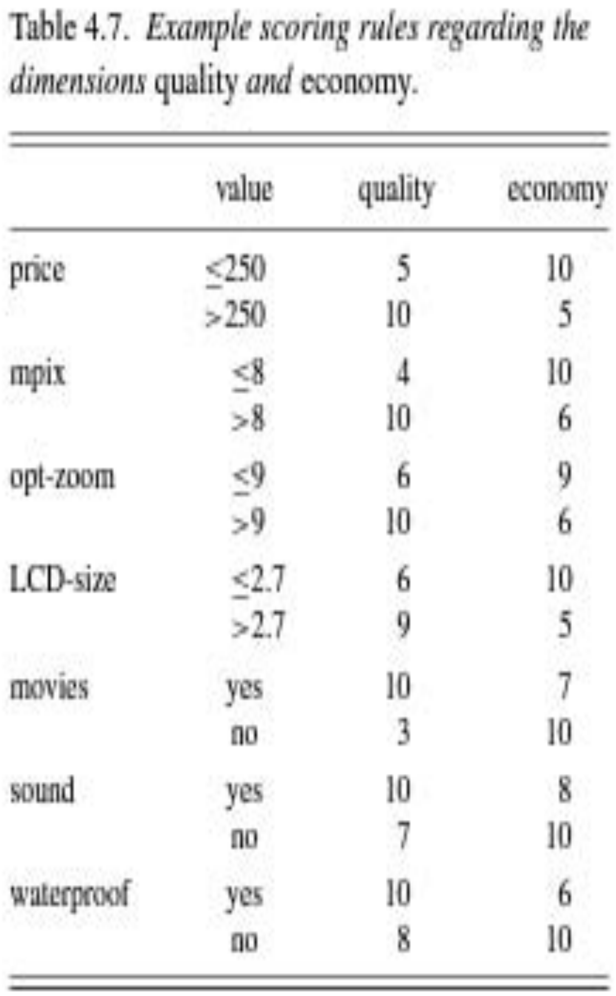


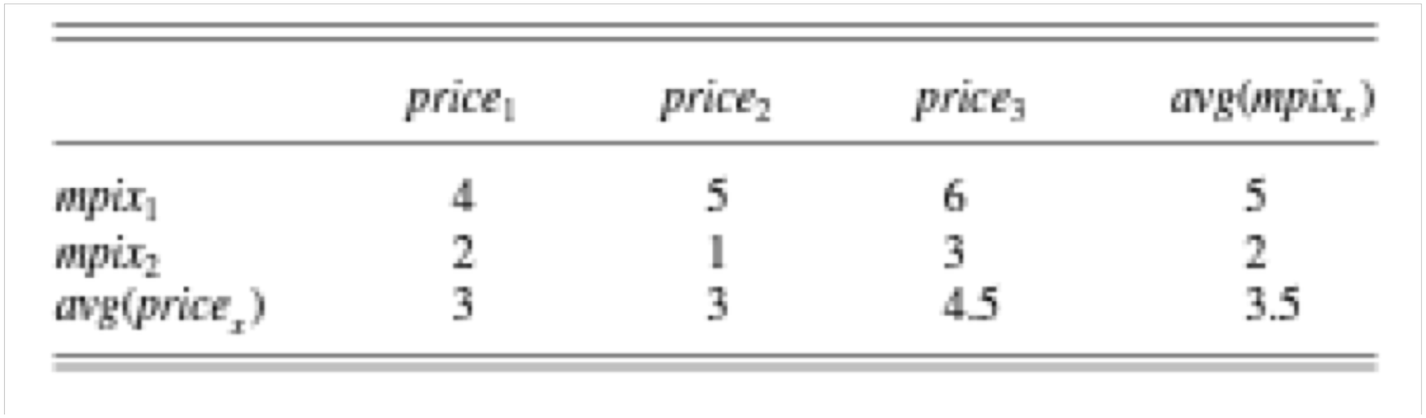
Table 4.7. *Example scoring rules regarding the dimensions* quality *and* economy.

	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

Assuming that customers cu1 and cu2 preferences are different (cu1: quality 80% and economy 20%; cu2: quality 40% and economy 60%), compute the utilities of items *p*1–*p*8 for cu1 and cu2.

User Preferences

- Different approaches for user preferences
  - User-defined preferences
  - Utility-based approaches
  - Conjoint analysis
    - Used to determine the relative importance of attributes



	$price_1$	$price_2$	$price_3$	$avg(mpix_x)$
$mpix_1$	4	5	6	5
$mpix_2$	2	1	3	2
$avg(price_x)$	3	3	4.5	3.5

Interacting with Case-Based Recommenders

- In addition to the query-based approach, used also in constraint-based recommenders, browsing-based approaches to item retrieval has been developed for case-based recommenders.
- Critiquing is an effective way to address the shortcomings of query-based approaches, in particular when users do not know exactly what they are seeking.
- State-of-the-art case-based recommenders are integrating query-based with browsing-based item retrieval.

Critiquing

- Simple Critiquing
- Compound Critiquing
- Dynamic Critiquing

Simple Critiquing

- Initial inputs are a user query, which specifies an initial set of requirements, and a set of candidate items that initially consists of all the available items.
- In the first critiquing cycle, the retrieval of items is based on a user query. Entry items are typically determined by calculating the similarity between the requirements and the candidate items. After the first critiquing cycle, recommended items are determined on the basis of the similarity between the currently recommended item and those items that fulfill the criteria of the critique specified by the user (e.g., cheaper). Procedure ItemRecommend is responsible for selecting entry/recommended items.
- The user reviews the recommended item and either accepts the recommendation or selects another critique, which triggers a new critiquing cycle. Procedure UserReview is responsible for accepting a critique request and reducing the scope of candidate items.

Compound Critiquing

- While simple critiquing (also called unit critiquing) permits a single item property to be specified, compound critiquing allows multiple properties to be changed.
- Example: “cheaper” (unit critique) vs. “cheaper and more mpx” (compound critique)
- Its main advantage is a faster progression through the item space (in other words, reduction of critiquing cycles).

Dynamic Critiquing

- Dynamic critiquing exploits patterns, which are generic descriptions of differences between the recommended item and the candidate items. These patterns are used for the derivation of compound critiques.
- Critiques are denoted as dynamic as they are derived on the fly in each critiquing cycle.
- Dynamic critiques are calculated using the concept of association rule mining (see Table 4.13).
- The user reviews the recommended item and either accepts the recommendation or selects a critique (unit or compound), starting a new critiquing cycle.