CS-4053 Recommender System

Spring 2023

Lecture 6: Knowledge-based Recommender Systems

Course Instructor: Syed Zain Ul Hassan

National University of Computer and Emerging Sciences, Karachi

Email: zain.hassan@nu.edu.pk

History-based Techniques

- Use historical data to make recommendations
  - Collaborative Filtering
  - Content-based Filtering
- These techniques require data *(a lot of it!)*
- Are not adaptable
- Do not *ask* user for preferences

Knowledge-based Recommender System

- Knowledge-based recommender systems exploit user requirements

- These systems are appropriate when:
  - Customers want to explicitly specify their requirements
  - It is difficult to obtain ratings for a specific type of item
  - When ratings are time-sensitive

Knowledge-based Recommender System

- Application domain
  - Expensive items, not frequently purchased, few ratings (car, house)
  - Time span important (technological products)
  - Explicit requirements of user (buying property)

Conceptual Goals of Recommender Systems

Approach	Conceptual Goal	Input
Collaborative	Give me recommendations based on a collaborative approach that leverages the ratings and actions of my peers/myself.	User ratings + community ratings
Content-based	Give me recommendations based on the content (attributes) I have favored in my past ratings and actions.	User ratings + item attributes
Knowledge-based	Give me recommendations based on my explicit specification of the kind of content (attributes) I want.	User specification + item attributes + domain knowledge

Knowledge-based Systems: Example

- If you're looking for a house or a car online. You input price, how many rooms, how much total floor space, etc., and the website returns a list of houses based on those constraints

- **Issue**:
  - Is this recommendation or simple query search?

Knowledge-based Recommender System

- Knowledge-based systems can be categorized into
  - **Constraint-based Systems**

    - Explicitly defined conditions

  - **Rule-based Systems**

    - Similarity to specified requirements

Constraint-based Systems

- The user specifies his or her initial preferences
  - All at once
  - Incrementally using a wizard
  - Interactive dialog
- The user is presented with a set of matching items
  - With an explanation as to why a certain item was recommended

Components of Constraint-based Systems

- Customer properties — V_C
- Product properties — V_{PROD}
- Constraints — C_R **(on customer properties)**
- Filter conditions — C_F **– relationship between customer and product**
- Products — C_{PROD} **– possible instantiations**

Knowledge base: Example

$V_C = \{kl_c$: [expert, average, beginner] . . . . /* level of expertise */
wr_c: [low, medium, high] . . . . /* willingness to take risks */
id_c: [shortterm, mediumterm, longterm] . . . . /* duration of investment */
aw_c: [yes, no] . . . . /* advisory wanted ? */
ds_c: [savings, bonds, stockfunds, singleshares] . . . . . . /* direct product search */
sl_c: [savings, bonds] . . . . /* type of low-risk investment */
av_c: [yes, no] . . . . /* availability of funds */
sh_c: [stockfunds, singlshares] . . . . /* type of high-risk investment */ $\}$

$V_{PROD} = \{name_p$: [text] . . . . /* name of the product */
er_p: [1..40] . . . . /* expected return rate */
ri_p: [low, medium, high] . . . . /* risk level */
$mniv_p$: [1..14] . . . . /* minimum investment period of product in years */
$inst_p$: [text] . . . . /* financial institute */ $\}$

Recommender Systems Handbook; Developing Constraint-based Recommenders

Knowledge base: Example

$$C_R = \{CR_1: wr_c = high \rightarrow id_c \neq shortterm,$$
$$CR_2: kl_c = beginner \rightarrow wr_c \neq high\}$$

$$C_F = \{CF_1: id_c = shortterm \rightarrow mniv_p < 3,$$
$$CF_2: id_c = mediumterm \rightarrow mniv_p \geq 3 \land mniv_p < 6,$$
$$CF_3: id_c = longterm \rightarrow mniv_p \geq 6,$$
$$CF_4: wr_c = low \rightarrow ri_p = low,$$
$$CF_5: wr_c = medium \rightarrow ri_p = low \lor ri_p = medium,$$
$$CF_6: wr_c = high \rightarrow ri_p = low \lor ri_p = medium \lor ri_p = high,$$
$$CF_7: kl_c = beginner \rightarrow ri_p \neq high,$$
$$CF_8: sl_c = savings \rightarrow name_p = savings,$$
$$CF_9: sl_c = bonds \rightarrow name_p = bonds\ \}$$

$$C_{PROD} = \{CPROD_1: name_p = savings \land er_p = 3 \land ri_p = low \land mniv_p = 1 \land inst_p = A;$$
$$CPROD_2: name_p = bonds \land er_p = 5 \land ri_p = medium \land mniv_p = 5 \land inst_p = B;$$
$$CPROD_3: name_p = equity \land er_p = 9 \land ri_p = high \land mniv_p = 10 \land inst_p = B\}$$

Recommender Systems Handbook; Developing Constraint-based Recommenders

Constraint-based Systems: Example

- Consider a simple food recommendation system with a knowledge-based containing attributes and constraints for both restaurant and customer

- The knowledge base presented is a minimal representation

Constraint-based Systems: Example

V_C =

{

v_{C1} : [vegan, no_dairy, no_requirements] …… */* dietary requirements */*

v_{C2} : [fast_food, traditional, continental, none] ….. */* food preference */*

v_{C3} : [high, medium, low] ….. */* spending style */*

…

}

Constraint-based Systems: Example

V_{PROD} =

{

v_{P1} : [pizza, BBQ, burger, biryani, italian] …… */* type of food */*

v_{P2} : [300, ... , 4000] ….. */* food price */*

v_{P3} : [*plaintext*] ….. */* name of food */*

…

}

Constraint-based Systems: Example

C_R =

{

c_{R1} : v_{C1} = vegan → v_{C2} ≠ fast food

c_{R2} : v_{C2} = continental → v_{C3} ≠ low

…

}

Constraint-based Systems: Example

C_F =

{

c_{F1} : v_{C1} = vegan → $v_{P1} \neq$ BBQ $\wedge$ $v_{P1} \neq$ burger

c_{F2} : v_{C3} = low → $v_{P2} < 1000$

…

}

Constraint-based Systems: Example

C_{PROD} =

{

c_{PROD1} : v_{P3} = afghani tikka pizza $\wedge$ v_{P2} = 1300
c_{PROD2} : v_{P3} = chicken burger $\wedge$ v_{P2} = 750
c_{PROD3} : v_{P3} = chicken biryani $\wedge$ v_{P2} = 400
c_{PROD4} : v_{P3} = alfredo pasta $\wedge$ v_{P2} = 1200
…

}

Constraint-based Systems: Example

- Now if the user sets their preference as:

{No dairy, none, low}

- The items recommended should be:

{chicken burger, chicken biryani}

Constraint-based Systems: Example

- Constraint-based recommendations can be seen as a subset selection problem

- Different techniques can be used to find subsets:
  - QuickXPlain
  - Linear Programming
  - Gradient Descent *(terrible idea!)*

Constraint-based Systems: Issues

- At times, users are presented with zero recommendations i.e., when no matching item is found
  - **Solution:** Relax some constraints

- Customers may be unsure about the values of their attributes
  - **Solution:** Suggest reasonable default values for attributes

- If business rules or items change often, knowledge-base has to be updated as well *(which can be an expansive task!)*

Case-based Systems

- In case-based systems, items are retrieved using similarity measures

- **sim(p, r)** expresses for each item attribute value, its distance to the customer requirement ***r* ∈ REQ**

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

- w_r is the importance weight for requirement r

Case-based Systems: Example

- Consider an example of a camera recommendation system

Item (Camera)	Price (in thousands)	Megapixel	LCD size	Waterproof
C1	12	8	3	No
C2	25	48	4	No
C3	55	60	6	Yes

- All of these available items are ***cases***

Case-based Systems: Example

- The allowed values of each attribute is given. We can use these values to normalize all the attributes

Item (Camera)	Price (in thousands)	Megapixel	LCD size	Waterproof
C1	12	8	3	No
C2	25	48	4	No
C3	55	60	6	Yes

- **Price :** { 12, 25, 55 }
- **Megapixel :** { 8, 10, 48, 54, 60 }
- **LCD size :** { 3, 4, 5, 6}
- **Waterproof :** { Yes, No }

Case-based Systems: Example

- The allowed values of each attribute is given. We can use these values to normalize all the attributes

Item (Camera)	Price (in thousands)	Megapixel	LCD size	Waterproof
C1	0.218	0.133	0.428	0
C2	0.454	0.8	0.571	0
C3	1	1	0.857	1

- **Price :** { 12, 25, 55 }
- **Megapixel :** { 8, 10, 48, 54, 60 }
- **LCD size :** { 3, 4, 5, 6, 7}
- **Waterproof :** { Yes, No }

Case-based Systems: Example

- Now if the user comes with the following query:

Price (in thousands)	Megapixel	LCD size	Waterproof
20	40	5	No

- We first normalize the attributes and then find similarity between each attribute in this query and the corresponding attribute in every available product

Case-based Systems: Example

- After normalizing user attributes (preferences):

Price (in thousands)	Megapixel	LCD size	Waterproof
0.363	0.666	0.714	0

- We first normalize the attributes and then find similarity between each attribute in this query and the corresponding attribute in every available product

Case-based Systems: Example

$$\boldsymbol{Sim}(\boldsymbol{Price_{REQ}}, \boldsymbol{PriceC_1}) = 1 - |\, 0.363 - 0.218 \,| = \mathbf{0.855}$$

$$\boldsymbol{Sim}(\boldsymbol{Megapixel_{REQ}}, \boldsymbol{Megapixel_{C1}}) = 1 - |\, 0.666 - 0.133 \,| = \mathbf{0.467}$$

$$\boldsymbol{Sim}(\boldsymbol{LCD_{REQ}}, \boldsymbol{LCDC_1}) = 1 - |\, 0.714 - 0.428 \,| = \mathbf{0.714}$$

- Assume the following weights for the attributes:
  - **w_{Price} = 1.5** *(price matters more to this user)*
  - **$w_{Megapixel}$ = 1**
  - **w_{LCD} = 1**

Case-based Systems: Example

- Now to find similarity of user requirements with **C1**:

$$\boldsymbol{Sim(User_{REQ}, C1)} = \frac{1.5 * 0.855 + 1 * 0.467 + 1 * 714}{1.5 + 1 + 1} = \mathbf{0.703}$$

- We can now find similarity of user requirements with **C2** and **C3** as well and recommend to the user a camera *(or top-k cameras)* with the highest similarity score

Knowledge-based Systems: Limitations

- Cost of knowledge acquisition and domain expertise
- Attributes (preferences) are not always independent