CS6120: Intelligent Media Systems

Dr. Derek Bridge
School of Computer Science & Information Technology
UCC

Motivation

- The recommender systems we have looked at so far
  - maintain a profile of the user's *long-term* interests
  - can be used either
    - reactively: user requests a recommendation ("user pull")
    - proactively: system makes recommendations unbidden ("system push")
  - but cannot (easily) respond to the user's *short-term* goals and interests

Approach

- We'll look at short-term goals and interests in the context of a user accessing an online product catalog
- The user will reveal her constraints and preferences in the form of a query (or sequence of queries)
- Products in the catalog will have descriptions
  - these will tend to be *structured* descriptions
- Matching goals/interests to product descriptions will use domain knowledge
  - knowledge of how well an item with a particular description will satisfy a particular goal/interest
  - hence these recommender systems are sometimes called *knowledge-based*

Running Example

Id	Address	Type	Bdrms	Bthrms	Rent	Furnished	Location
1	16 Oxford Road	Flat	1	1	265	Yes	Acton
2	2 Heathfield Road	House	3	2	370	Yes	Acton
3	101 Nassau Road	Flat	2	1	271	No	Barnes

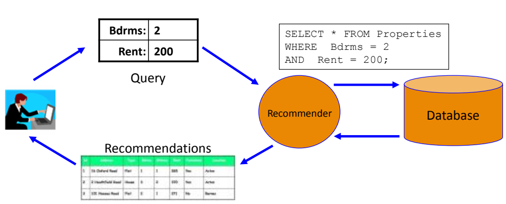
- Columns are called *attributes* and each piece of data is a *value*
  - so we have *attribute-value pairs*, e.g. Type = Flat

Eliciting the query: forms

- User can express constraints by submitting a form
  - fields can be left blank to indicate 'wild card'

Retrieval

- The query values are used to build an SQL query
  - executed against the database
  - its results are shown to the user



Filter-Based Retrieval

- Typically, SQL will be used to perform *filter-based retrieval*
  - exact matching
- This brings two problems
  - result set may be empty: query is *over-specified*
  - result set may be too large: query is *under-specified*
- Some systems will attempt to lessen the first of these two problems, e.g.
  - User's query: Rent = 200
  - SQL: `SELECT * FROM Properties WHERE Rent > 190 AND Rent < 210`
  - But this has at least two weaknesses! What are they?

Similarity-Based Retrieval

- An alternative is *similarity-based retrieval*
  - score each item (based on similarity to the query)
  - rank them on their scores
  - recommend those at the top of the ranking (in decreasing order of score)
- In this case,
  - result set is never empty (no matter how under-specified the query is)
  - result set can be a manageable length, and in any case is ordered

Similarity: $sim(q, i)$

- A global similarity function, $sim(q, i)$, is defined as a combination of local similarity functions, $sim_A(q, i)$, one for each attribute A in the query

	Id	Address	Type	Bdrms	Bthrms	Rent	Furnished	Location
i:	1	16 Oxford Road	Flat	1	1	265	Yes	Acton
q:			Flat	3		200	No	Hayes
$sim(q,i) = \Sigma$:			sim_{type}	sim_{bdrms}	sim_{bthrms}	sim_{rent}	$sim_{furnished}$	$sim_{location}$

Global and Local Similarities

- E.g. sum the local similarities

$$sim(q, i) = \sum_{A \in q} sim_A(q, i)$$

- E.g. take a weighted sum

$$sim(q, i) = \sum_{A \in q} w_A \times sim_A(q, i)$$

- E.g. can take averages or weighted averages

Local Similarity Functions

- E.g. one local similarity function is called the *overlap function*
  - very good for non-numeric attributes, especially ones with just two values, e.g. Type, Furnished

$$sim_A(q_A, i_A) = \begin{cases} 1 & \text{if } q_A = i_A \\ 0 & \text{otherwise} \end{cases}$$

Local Similarity Functions

- For numeric attributes, the absolute difference can form the basis:

$$abs(q_A - i_A)$$

- But, attributes with large range can overpower other attributes. So normalize:

$$\frac{abs(q_A - i_A)}{A_{max} - A_{min}}$$

- And this is a distance function but we need a similarity function so subtract from 1:

$$sim_A = 1 - \frac{abs(q_A - i_A)}{A_{max} - A_{min}}$$

Local Similarity Functions

- Human experts might define domain-specific similarity functions, esp. for non-numeric attributes
- E.g. $sim_{location}$

	Acton	Barnes	Chelsea	Ealing	Hayes
Acton	1.0	0.6	0.3	0.9	0.8
Barnes		1.0	0.2	0.8	0.7
Chelsea			1.0	0.6	0.5
Ealing				1.0	0.8
Hayes					1.0

Exercise

- Assuming
  - Global similarity is the sum of local similarities
  - Bdrms has range 0-8
  - Rent has range 100-750

  what is the global similarity of property number 1 to the query q?

	Id	Address	Type	Bdrms	Bthrms	Rent	Furnished	Location
i:	1	16 Oxford Road	Flat	1	1	265	Yes	Acton
		q:	Flat	3		200	No	Hayes
		$sim(q,i) = \Sigma$:						

Recap

- In contrast to filter-based retrieval, similarity-based retrieval and utility-based retrieval
  - compute a score for each item
    - typically, a sum or average of local similarities/utilities, one per attribute in the query
    - typically, for local utility, similarity is used as a proxy – but not always
  - ranks the item in order of descending score
  - recommends the top-ranking items (in descending order of score)
- A surprisingly under-used idea
  - but, where used, highly successful
- In all cases,
  - should consider enhancing diversity

CONVERSATIONAL SYSTEMS

Single-Shot Systems

- We've been assuming a *single-shot system*
  - submit query, view results, end of story
- But
  - seldom are we able to specify all our requirements up-front
  - seldom are we satisfied with the initial set of results (irrespective of whether the system uses filter- or similarity-based retrieval)
    - if not satisfied, our only option is to revise the query and submit again
      - typically with no guidance
      - can lead to 'stonewalling'
  - seldom are queries within a session independent

Conversational Systems

- A *conversational recommender system*
  - an iterative approach
  - users can elaborate their requirements as part of an extended recommendation dialog
- Techniques
  - *Navigation-by-asking*
    - recommender selects and asks questions
    - user may or may not answer the questions
  - *Navigation-by-proposing*
    - recommender makes interim recommendations
    - user provides feedback on these recommendations (e.g. critiques)

Navigation-By-Asking: Desiderata

- Questions should be few in number
- Questions should have a comprehensible ordering/grouping
- Each question should be comprehensible
- Each question should have low answering cost
- …

Navigation-By-Asking

- Let's focus on minimizing the number of questions
- *Statically-defined dialog*
  – will not minimize the number of questions since next question is fixed → insensitive to user's answers to previous questions
- *Dynamically-defined dialog*
  – next question is chosen based on an analysis of the distribution of remaining candidate items
- For simplicity, let's assume filter-based retrieval
  – i.e. exact-matching

Check your intuitions

- Suppose these are the candidate items

Id	Colour	Size	Weight
1	red	small	light
2	red	small	light
3	red	large	heavy
4	blue	small	heavy
5	blue	small	heavy
6	red	small	light
7	red	small	light
8	blue	small	heavy
9	blue	large	heavy
10	blue	large	medium

- You can ask the user to supply a preferred *colour* or a preferred *size* or a preferred *weight*.

Which would you ask for first?

Check your intuitions, continued

- We'll suppose the user gives us an answer to our first question. In the lecture, delete parts of the table that are no longer relevant:

Id	Colour	Size	Weight
1	red	small	light
2	red	small	light
3	red	large	heavy
4	blue	small	heavy
5	blue	small	heavy
6	red	small	light
7	red	small	light
8	blue	small	heavy
9	blue	large	heavy
10	blue	large	medium

- What would you ask next?

Information Gain

- Let C be the remaining candidate items
- Suppose attribute A has a set of possible values, V
  - e.g. for $A = Colour$, $V = \{red, blue\}$
- Let $C_{A=v}$ be those members of C for which $A = v$
- The information gain for an attribute A, $Gain(A)$:

$$Gain(A) = -\sum_{v \in V} \frac{size(C_{A=v})}{size(C)} \times \log(\frac{size(C_{A=v})}{size(C)})$$

- Log? Should be log_2 but you can use the button on your calculator labeled log, which is log_{10}. This will not change the outcome here

Worked Example

- Let's compute, $Gain(Colour)$:

$$Gain(A) = -\sum_{v \in V} \frac{size(C_{A=v})}{size(C)} \times \log(\frac{size(C_{A=v})}{size(C)})$$

Worked Example

- Compute $Gain(Size)$ and $Gain(Weight)$ in your own time
- But here are the answers, so that you can check yours against mine:
  - $Gain(Colour) = 0.3$
  - $Gain(Size) = 0.26$
  - $Gain(Weight) = 0.37$

Dynamic Dialog

let $Candidates$ be the entire product catalog

repeat the following until $Candidates$ is small enough to display on the screen or all candidates have the same values for all attributes

Compute the information gain of each unasked attribute

Choose the attribute with highest information gain

Ask the user for her preferred value for this attribute

Remove from $Candidates$ all products which do not have this value for this attribute

Discussion

- Our treatment assumes filter-based retrieval
  - however, a variation has been defined that works for similarity-based/utility-based retrieval
  - S.Schmitt (2002): *simVar: A similarity-influenced question-selection criterion for e-sales dialog,* Artificial Intelligence Review, vol.18(304), pp.195-221
- We have only considered minimizing dialog length
  - it is easy to incorporate question costs, if they are known (which they rarely are)
  - comprehensible ordering/grouping might be achievable by incorporating a similarity measure *between questions*
  - if users have the option of declining to answer a question, we have the opportunity to learn answering preferences in order to personalize dialogs

Navigation-by-Proposing: intuition

Problem?

- Asking the user questions, whether up-front (e.g. form-filling) or incrementally (navigation-by-asking) still requires that she
  - knows her own mind
  - is able to articulate her preferences

Solution?

- On the other hand, if we show the user one or more items (interim recommendations), she may more easily be able to say
  - what she likes about them
  - what she dislikes about them

Critiquing

- *Critiquing* is one form of navigation-by-proposing
- How it works (roughly)
  - the system shows the user an item
  - the user supplies a critique of the item (e.g. *"cheaper", "more bedrooms",…*)
  - the system retrieves all items that satisfy the critique
  - of these items, it shows the user the one that is most similar to the one being critiqued
- This captures the idea of *"like this but…"*

Entrée: Restaurant Recommender

Worked Example

Id	Address	Type	Bdrms	Bthrms	Rent	Furnished	Location
1	16 Oxford Road	Flat	1	1	265	Yes	Acton
2	2 Heathfield Road	House	3	2	370	Yes	Acton
3	101 Nassau Road	Flat	2	1	271	No	Barnes
4	78 Moscow Road	Flat	3	1	850	Yes	Bayswater

Worked Example

- Suppose the system shows the user the following item:

Id	Address	Type	Bdrms	Bthrms	Rent	Furnished	Location
2	2 Heathfield Road	House	3	2	370	Yes	Acton

- The user selects the "cheaper" critique
- So she want to see the items that are
  - "like this property but cheaper"

Worked Example

- Since the item has Rent = 370, the user's critique can be expressed as Rent < 370
- The system finds all items that satisfy the critique
  - `SELECT * FROM Properties WHER Rent < 370;`

Id	Address	Type	Bdrms	Bthrms	Rent	Furnished	Location
1	16 Oxford Road	Flat	1	1	265	Yes	Acton
3	101 Nassau Road	Flat	2	1	271	No	Barnes

- Call these the *Candidates*

Worked Example

- For each candidate item i, compute $sim(s, i)$ where s is the selected item

2	2 Heathfield Road	House	3	2	370	Yes	Acton
1	16 Oxford Road	Flat	1	1	265	Yes	Acton
$sim(id2, id1) = \Sigma$:		0	0.25	0.875	0.838	1	1

2	2 Heathfield Road	House	3	2	370	Yes	Acton
3	101 Nassau Road	Flat	2	1	271	No	Barnes
$sim(id2, id3) = \Sigma$:		0	0.875	0.875	0.848	0	0.6

$sim(id2, id1) = 3.963$ $sim(id2, id3) = 3.198$

Worked Example

- Show the user the highest scoring item:

Id	Address	Type	Bdrms	Bthrms	Rent	Furnished	Location
1	16 Oxford Road	Flat	1	1	265	Yes	Acton

  – "like this but cheaper"!

Broader Issues

- Both navigation-by-asking and navigation-by-proposing require the user to have a lot of knowledge/understanding
- Both impose a burden on the user
  – sh/e must interact with the system
- In both, the fixation has been on minimizing dialog length
  – why might this be wrong? In other words, why might a user prefer a longer dialog than is strictly necessary

Broader Issues

- This lecture has been about short-term interests/goals
  - explored in the context of a knowledge-based recommender
- How do we build content-based and collaborative recommenders that can elicit and respond to short-term goals/interests?
  - to balance short- and long-term preferences