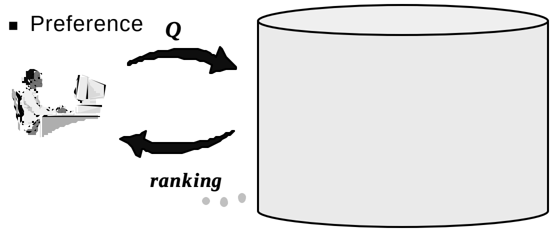
Ranking and Preference in Database Search: a) Similarity and Relevance

Kevin Chen-Chuan Chang

The Database and Information Systems Laboratory
at The University of Illinois at Urbana-Champaign
Large Scale Information Management

Ranking– Ordering according to the degree of some fuzzy notions:

- Similarity (or dissimilarity)
- Relevance
- Preference



Similarity!-- Are they similar?

- Two images

Similarity!-- Are they similar?

- Two images

So, similarity is not a Boolean notion– It is relatively ranking

5

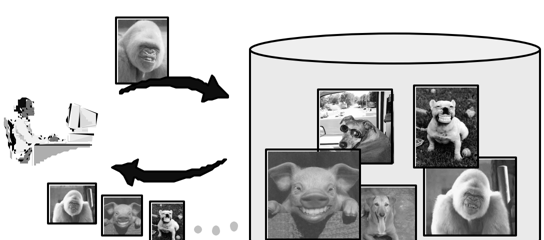
Similarity– Are they similar?

- Two strings

Virginia
Vermont

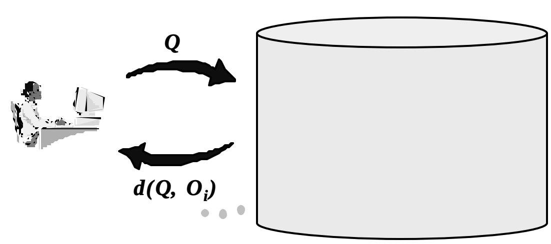
6

Ranking by similarity



7

Similarity-based ranking –- by a “distance” function (or “dissimilarity”)



8

The “space”– Defined by the objects and their distances

- Object representation– Vector or not?
- Distance function– Metric or not?

9

Vector space– What is a vector space?

(S, d) is a vector space if:

- Each object in S is a k-dimensional vector
  - $x = (x_1, ..., x_k)$
  - $y = (y_1, ..., y_k)$
- The distance $d(x, y)$ between any x and y is *metric*

10

Vector space distance functions – The L_p distance functions

- The general form:

$$L_P(x:(x_1, ..., x_k), y:(y_1, ..., y_k)) = \left(\sum_{i=1}^{k} |x_i - y_i|^P\right)^{\frac{1}{p}}$$

- AKA: p-norm distance, Minkowski distance
- Does this look familiar?

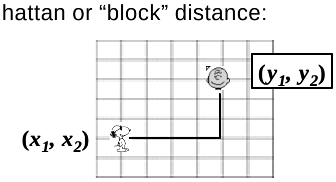
11

Vector space distance functions – L_1 : The Manhattan distance

- Let p=1 in L_p:

$$L_1(x:(x_1, ..., x_k), y:(y_1, ..., y_k)) = \sum_{i=1}^{k} |x_i - y_i|$$

- Manhattan or “block” distance:



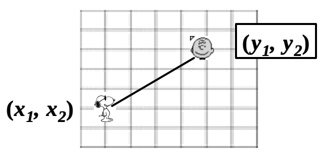
12

Vector space distance functions – L_2 : The Euclidean distance

- Let p=2 in L_p:

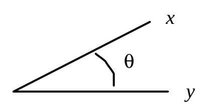
$$L_P(x:(x_1,...,x_k), y:(y_1,...,y_k)) = (\sum_{i=1}^{k} |x_i - y_i|^2)^{\frac{1}{2}}$$

- The shortest distance

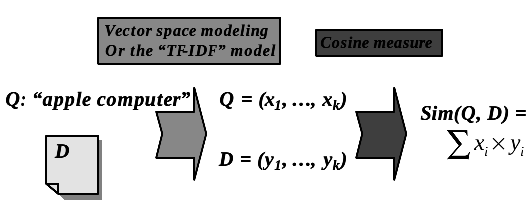


Vector space distance functions– The Cosine measure

$$sim(x, y) = \cos(q) = \frac{x \bullet y}{|x| \times |y|} = \frac{\sum x_i \times y_i}{\sqrt{\sum x_i^2} \times \sqrt{\sum y_i^2}}$$



Sounds abstract? That's actually how Web search engines (like Google) work



How to evaluate vector-space queries? Consider *Lp* measure--

- Consider L_2 as the ranking function
  - Given object Q, find O_i of increasing $d(Q, O_i)$
- How to evaluate this query? What index structure?
  - As nearest-neighbor queries
  - Using multidimensional or spatial indexes. e.g., R-tree [Guttman, 1984]

How to evaluate vector-space queries? Consider Cosine measure--

- *Sim(Q, D)* = $\sum x_i \times y_i$
- How to evaluate this query? What index structure?
  - Simple computation: multiply and sum up
  - *Inverted index* to find document with non-zero weights for query terms

17

Is vector space always possible?

- Can you always express objects as k-dimensional vectors, so that
  - distance function compares only corresponding dimensions?
- Counter examples?

18

How about comparing two strings? Is it natural to consider in vector space?

- Two strings

Virginia
Vermont

19

Metric space– What is a metric space?

- Set *S* of objects
- Global *distance* function d, (the "*metric*")
- For every two points *x, y* in *S:*
  - *Positiveness:* $d(x, y) \geq 0$
  - *Symmetry* $d(x, y) = d(y, x)$
  - *Reflexivity* $d(x, x) = 0$
  - *Triangle inequity* $d(x, y) \leq d(x, z) + d(z, y)$

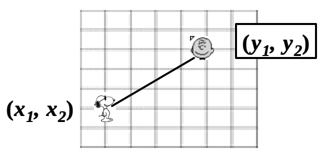
20

Vector space is a special case of metric space– E.g., consider L_2

- Let p=2 in L_p:

$$L_P(x:(x_1,\ldots,x_k), y:(y_1,\ldots,y_k)) = (\sum_{i=1}^{k} |x_i - y_i|^2)^{\frac{1}{2}}$$

- The shortest distance



21

Another example-- Edit distance

- The smallest number of edit operations (insertions, deletions, and substitutions) required to transform one string into another
  - **`Virginia`**
  - **`Verginia`**
  - **`Verminia`**
  - **`Vermonta`**
  - **`Vermonta`**
  - **`Vermont`**
- http://urchin.earth.li/~twic/edit-distance.html

22

Is edit distance metric?

- Can you show that it is symmetric?
  - Such that d(Virginia, Vermont) = d(Vermont, Virginia)?
    - **`Virginia`**
    - **`Verginia`**
    - **`Verminia`**
    - **`Vermonta`**
    - **`Vermonta`**
    - **`Vermont`**
- Check other properties

23

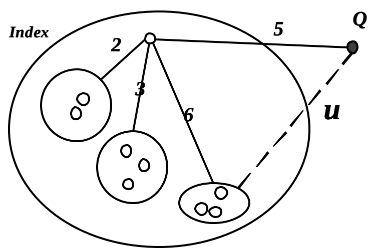
How to evaluate metric-space ranking queries? [Chávez et al., 2001]

- Can we still use R-tree?
- What property of metric space can we leverage to “prune” the search space for finding near objects?

24

Metric-space indexing

- What is the range of u?
- How does this help in focusing our search?



25

Relevance-based ranking – for text retrieval

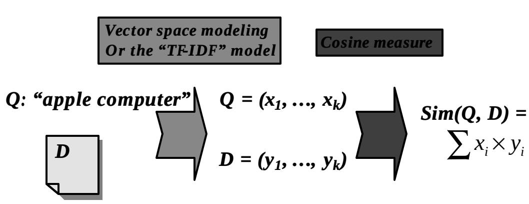
What is being "relevant"?

Many different ways modeling relevance

- *Similarity*
  - How similar is D to Q?
- *Probability*
  - How likely is D relevant to Q?
- *Inference*
  - How likely can D infer Q?

26

Similarity-based relevance -– We just talked about this "vector-space modeling" [Salton et al., 1975]



- TF-IDF for term weights in vectors
  - TF: *term frequency* (in this document)
    - the more term occurrences in this doc, the better
  - IDF: *inverse document frequency* (in entire DB)
    - the fewer documents contain this term, the better

27

Probabilistic relevance

- View: *Probability* of relevance
  - the "probabilistic ranking principle" [Robertson, 1977]

*"If a retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing **probability of usefulness** to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.*

- Initial idea proposed in [Maron and Kuhns, 1960] many models followed.

28

Probabilistic models (e.g.: [Croft and Harper, 1979])

- Estimate and rank by P(R | Q, D), or $\log \frac{P(R|Q,D)}{P(\overline{R}|Q,D)}$
  - I.e., $\log \prod_{t_i \in Q,D} \frac{p_i}{1-p_i} \cdot \frac{1-q_i}{q_i}$, where $p_i = P(t_i \mid R)$
    $q_i = P(t_i \mid \overline{R})$
- Assume
  - p_i the same for all query terms
  - $q_i = n_i/N$, where N is DB size
    - (i.e., "all" docs are non-relevant)
- $\log \prod_{t_i \in Q,D} \frac{p_i}{1-p_i} \cdot \frac{1-q_i}{q_i} \propto \log \prod_{t_i \in Q,D} \frac{1-q_i}{q_i} = \log \prod_{t_i \in Q,D} \frac{N-n_i}{n_i} = \sum_{t_i \in Q,D} \log \frac{N-n_i}{n_i}$
- Similar to using "IDF"
  - intuition: e.g., "apple computer" in a computer DB

29

This is how we derive the ranking function:

- To rank by $\log \frac{P(R|Q,D)}{P(\overline{R}|Q,D)}$

$$\frac{P(R|Q,D)}{P(\overline{R}|Q,D)} = \frac{P(Q,D|R)P(R)}{P(Q,D|\overline{R})P(\overline{R})} \propto \frac{P(Q,D|R)}{P(Q,D|\overline{R})}$$

$$P(Q,D|R) = \prod_{t_i \in Q,D} P(t_i|R) \prod_{t_j \in Q,D} (1-P(t_j|R)) = \prod_{t_i \in Q,D} p_i \prod_{t_j \in Q,\overline{D}} (1-p_j)$$

$$P(Q,D|\overline{R}) = \prod_{t_i \in Q,D} P(t_i|\overline{R}) \prod_{t_j \in Q,D} (1-P(t_j|\overline{R})) = \prod_{t_i \in Q,D} q_i \prod_{t_j \in Q,\overline{D}} (1-q_j)$$

$$\frac{P(R|Q,D)}{P(\overline{R}|Q,D)} = \frac{\prod_{t_i \in Q,D} p_i \prod_{t_j \in Q,\overline{D}} (1-p_j)}{\prod_{t_i \in Q,D} q_i \prod_{t_j \in Q,\overline{D}} (1-q_j)} \propto \frac{\prod_{t_i \in Q,D} p_i(1-q_i)}{\prod_{t_i \in Q,D} q_i(1-p_i)} = \prod_{t_i \in Q,D} \frac{p_i}{1-p_i} \cdot \frac{1-q_i}{q_i}$$

30

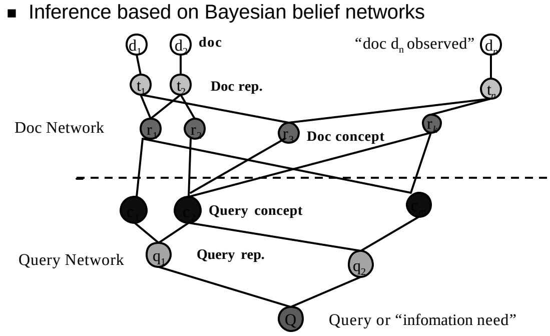
Inference-based relevance

- Motivation
  - Is there any "objective" way of defining relevance?
  - Hint from a logic view of database querying: retrieve all objects s.t., O → Q
    - E.g., O = (john, cs, 3.5) → gpa>3.0 AND dept=cs
  - What about "Retrieve D iff we can prove D→Q"?
- Challenges: Uncertainty in inference? [van Rijsbergen, 1986]
  - Representation of documents and queries
  - Quantify the uncertainty of inference P(D→Q) = P(Q|D)

31

Inference network [Turtle and Croft, 1990]

- Given doc as evidence, prove that info need is satisfied
- Inference based on Bayesian belief networks



32

Using and constructing the network

- Using the network: Suppose all probabilities known
  - Document network can be pre-computed
  - For any given query, query network can be evaluated
  - P(Q|D) can be computed for each document
  - Documents can be ranked according to P(Q|D)
- Constructing the network: Assigning probabilities
  - Subjective probabilities
  - Heuristics, e.g., TF-IDF weighting
  - Statistical estimation
    - Need “training”/relevance data

33

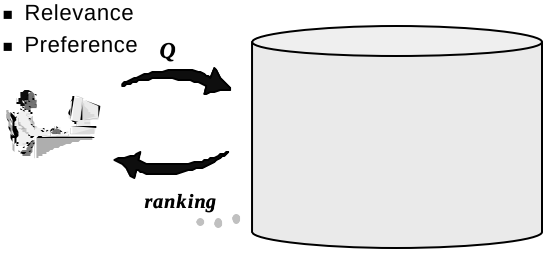
Ranking and Preference in Database Search: b) Preference Modeling

Kevin Chen-Chuan Chang

DAIS The Database and Information Systems Laboratory
at The University of Illinois at Urbana-Champaign
Large Scale Information Management

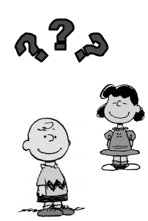
Ranking– Ordering according to the degree of some fuzzy notions:

- Similarity (or dissimilarity)
- Relevance
- Preference



35

What do you prefer? For a job.



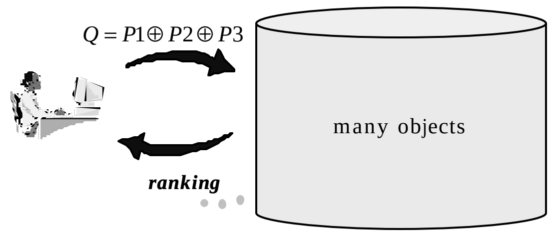
36

Stating your dream job? It's all about preferences

- **Expressing** preferences:
  - *P1*: Pay well – The more salary the better!
  - *P2*: Not much work – The less work the better!
  - *P3*: Close to home – The closer the better!
- **Combining** preferences:
  - How to combine your multiple wishes?
- **Querying** preferences:
  - How to then match the perfect job?

37

This setting is somehow different from typical voting scenarios



38

Different approaches

- **Qualitative**
  - Preferences are specified directly using relations
  - E.g., I prefer X to Y; you like Y better than X
- **Quantitative**
  - Preferences are specified indirectly using scoring functions
  - E.g., I like X with score .3, and Y with .5

39

Quantitative approach [Agrawal and Wimmers, 2000]

- Preference can be measured by "utility" values
  - Quantification of how useful things are
- Such quantification facilitates the search for optimal decisions as maximal utility scores

40

Expressing preference: Preference functions

- **Preference function**:
  - Mapping a record of a given type to a numeric score.

E.g. Laptop1('dell',1600,5.6,14,'P4 2GHZ')

Alice's preference function			
brand	price	weight	score
dell	>1500	*	0.3

A(laptop1)=0.3

Bob's preference function			
brand	price	weight	score
*	*	>5	veto

B(laptop1)=veto

Conflicts may arise between preferences

Consider a record
Laptop1: ('dell',1600,5.6,14, 'P4 2GHZ')

- Conflicts within one pref function
  - Alice's preference 3 → 0.3
  - Alice's preference 4 → 0.9
- Conflicts between two pref functions
  - Alice's preference 3 → 0.3
  - Bob's preference 4 → 0.6
- Need to find a way to reach a final decision!

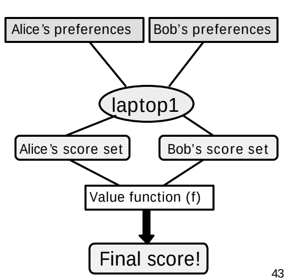
Alice's preference function			
brand	**price**	**weight**	**score**
*	*	< 3	0.8
ibm	<1500	*	0.8
*	>1500	*	0.3
dell	*	*	0.9

Bob's preference function			
brand	**processor**	**LCD size**	**score**
*	celeron	*	veto
*	*	<15	0.8
ibm	P4 2GHz	*	0.9
dell	P4 2GHz	*	0.6

Combining preferences: Value function that consider relevant scores and the record

$$combine(f)(p_1,\ldots,p_n)(r) = f(Scores(p_1,r),\ldots,Scores(p_n,r),r)$$

- Value function f
  - for merging scores
- Consider only
  - all relevant scores of r
  - the record r itself



Combining preferences: Example

- Considering the record Laptop1('dell',1600,5.6,14,'P4 2GHZ')
  - A(laptop1)={0.3,0.9}
  - B(laptop1)={0.6,0.8}

Rules:
- Bob has veto power over any laptop they buy.
- If price is higher than $1550, Bob will decide; otherwise listen to Alice.

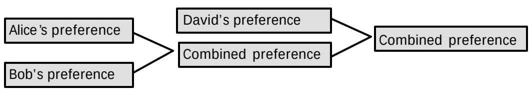
- f(Alice's score set, Bob's score set, laptop1)

```
{     if (veto in Bob's score set) then return veto
        else if price>1550 then return max(Bob's score set)
               else return average(Alice's score set)
}
```

- combine(f)(A,B)(laptop1) =
  f(A(laptop1), B(laptop1), laptop1) = 0.8

Properties of combining functions: Closure

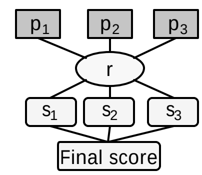
- Closure



- Why is this desirable?
  - Allow flexible compositions of preferences

Properties of combining functions: Modular

- Modular
  - Combined score of r only depends on the scores of r
- Why is this desirable?
  - Pref are autonomous:
    - Change IBM will no affect Dell
  - Ease of implementation
    - "Context free", or "first order"
- Counter example?



Querying preferences – Ranking by preference scores

- Top-k queries–
  - Finding top k answers with highest scores
- Much research effort in this area
  - We will see next time

Quantitative model: Advantages

- Advantages:
  - Discriminative scoring and tie resolution
  - Efficient implementation
- Problems?

Quantitative model: Problems

- Problems:
  - Not obvious how to specify scores
  - Not obvious how to decide combining functions
  - Total ordering by scores is not always reasonable

49

Qualitative approach: Specify pairwise ordering relation between objects

Book No.	ISBN	Vender	Price
1	0679726691	BooksForLess	$14.75
2	0679726691	LowestPrices	$13.50
3	0679726691	QualityBooks	$18.80
4	0062059041	BooksForLess	$7.30
5	0374164770	LowestPrices	$21.88

Preference 1. (Preference on Best Price)
If the same ISBN, prefer lower Price to higher price

⇨ Preference 1 can be expressed as a binary relation (b1,b2) such that:
b1.ISBN = b2.ISBN $\wedge$ b1.Price < b2.Price

50

Quantitative approach? [Chomicki, 2003]

Book No.	ISBN	Vender	Price
1	0679726691	BooksForLess	$14.75
2	0679726691	LowestPrices	$13.50
3	0679726691	QualityBooks	$18.80
4	0062059041	BooksForLess	$7.30
5	0374164770	LowestPrices	$21.88

Preference 1. (Preference on Best Price)
If the same ISBN, prefer the one with lower Price

⇨ Score(Book2) > Score(Book1) > Score(Book3)
Score(anyof Book 1, 2, 3) = Score(Book4) = Score(Book 5)
$\Rightarrow$ Score(Book 1) = Score(Book2) = Score(Book3)

There is no score function that captures Preference 1

51

Qualitative ⊃ Quantitative

- Qualitative: Preference relation
- Quantitative: Scoring function

- Scoring-based ordering can be captured by preference relations
- But, not every intuitively plausible preference relation can be captured by scoring function

52

Preference as ordering [Kießling, 2002; Chomicki, 2003]

- It is natural, intuitive that people express their wishes: "**I like X better than Y**" or "**I prefer X to Y"**
- Better-than can be captured by a binary relation
- X and Y can be any records, as a set of attributes
  - E.g., Book (ISBN, Vender, Price)
- E.g., Let <P1 be the relation for Preference 1 in Book

(0679726691,BooksForLess,$14.75)

<P1 (0679726691,LowestPrices,$13.50)

53

Preference: Strict partial order

- Given a set **A** of attribute names with value domain **dom(A)**
  - A **preference P** is a **strict partial order** P=(A, <P) on dom(A)
  - x <P y is interpreted as "I like y better than x",
  - x and y are indifferent iff
    - neither x <P y nor y <P x
- Properties of preferences
  - Irreflexive : x (not <P) x
  - Transitive: x <P y and y <P z ➔ x <P z
  - Asymmetric: x <P y ➔ y (not <P) x
- Strict partial order
  - Strict:
    - Since if x<P y hold then y <P x doesn't, like "less than" (asymmetric)
  - Partial:
    - Since <P not enforced on every pair of objects

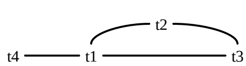
54

Preference graph, or the "better than" graph

Directed, acyclic graph (why acyclic?)

- An edge (y $\rightarrow$ x) exists for x <P y

t2 <P t1, t2 <P t3, t1 <P t4, t1 <P t3



- Nodes in G without a predecessor are ***maximal elements*** of P (max(P)), being at ***level 1***
- x is on level *j*, if the ***longest*** path from x to a maximal node has *j*-1 edges
- x, y are ***unranked*** If no directed path exists between x and y

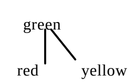
55

Expressing preference: Base preference constructors

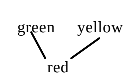
- Non-numerical base preferences

dom(Color)={red, yellow, green}

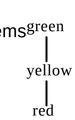
  - Specify the items which is preferred
    - POS(color, {green})



  - Specify the items which is not preferred
    - NEG(color, {red})



  - Explicitly specify the preference between pairs of items
    - EXP(color, {(yellow,green), (red,yellow)})



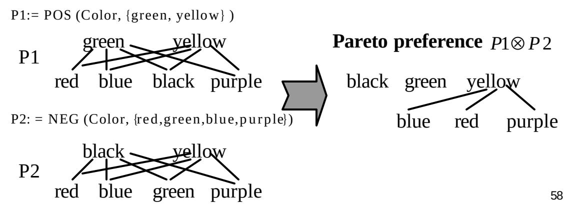
56

Expressing preference:
Base preference constructors

- Numerical base preferences
  - Prefer the value around a specific value
    - AROUND (price, 40000)
  - Prefer the value within a specific range
    - BETWEEN (mileage, [20000,30000])
  - Prefer the value as low (high) as possible
    - LOWEST (price)
  - Preference is based on some scoring function
    - f(price)
    - x <P y iff f(x) < f(y)

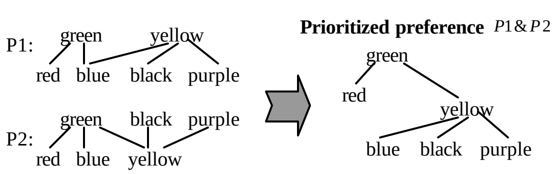
Combining preferences:
Complex Preference Constructors -- Pareto

- If P1 and P2 are considered equally important, how to combine then?
- Pareto: Only preserve those orders in consensus



Combining preferences:
Complex Preference Constructors -- Priority

- If P1 is more important than P2, how to combine?
- Priority: P1 first then P2



Querying preferences

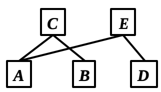
Given P=(A,<P) and a relation R, R[A] $\subseteq$ dom(A)
A preference query σ[P](R) is a soft selection operation on R

- Best-Matches-Only (BMO) query model
  - Retrieve perfect choices, if present in R
    - Perfect choices are maximal elements of P
  - Otherwise deliver best-matching alternatives (tuples with lowest level), but nothing worse
- Ranking ("top-k") or iterated preferences
  - Order tuples according their level value

The BMO query model

Laptop	price	weight
A	4000	5.4
B	3200	5
C	3000	4.8
D	1200	5.8
E	1000	5.2

- Suppose base preferences:
  - P1: LOWEST(price)
    - E → D → C → B → A
  - P2: LOWEST(weight)
    - C → B → E → A → D
- Combined preference: P1Ä P2
  - Better-than Graph:



Level 1

Level 2

- BMO answers: σ[P](R)={*C*, *E*}
- Challenge: Answer BMO without fully computing P1Ä P2 (Next time)

61

Qualitative or quantitative?

- Consider different aspects:
  - Query expression?
  - Query processing?
  - Result presentation?
- What do you suggest?

62

Conjecture– Perhaps a hybrid…

- Front-end: Rank expression
  - Let user specify preference in partial orders
- Back-end: Rank processing
  - Process with an approximate score-based ordering

63

Thank You!

64