Using the crowd for Top-k and Group-by Queries

Outline

- Problem Definition
- Related Work
- Error Model
- Max and Top-k
- Clustering
- Clustering with correlated types and values

The Problem

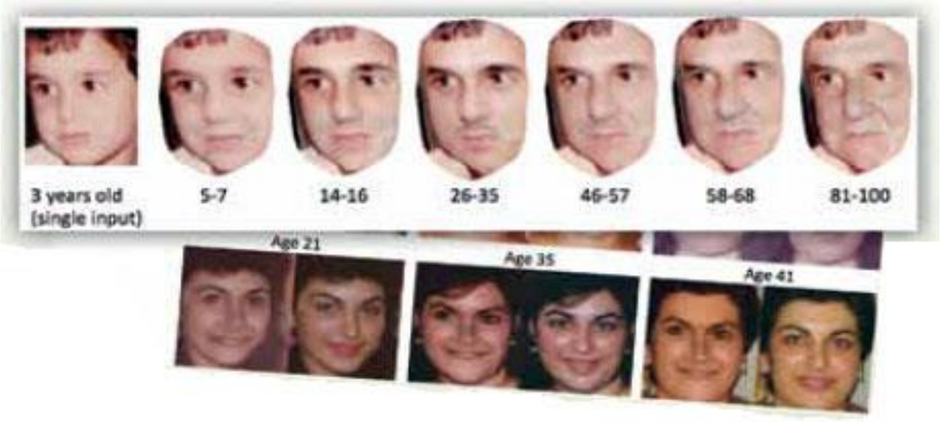
- Group photos of same person type

- Arrange photos within same group as per age(value)

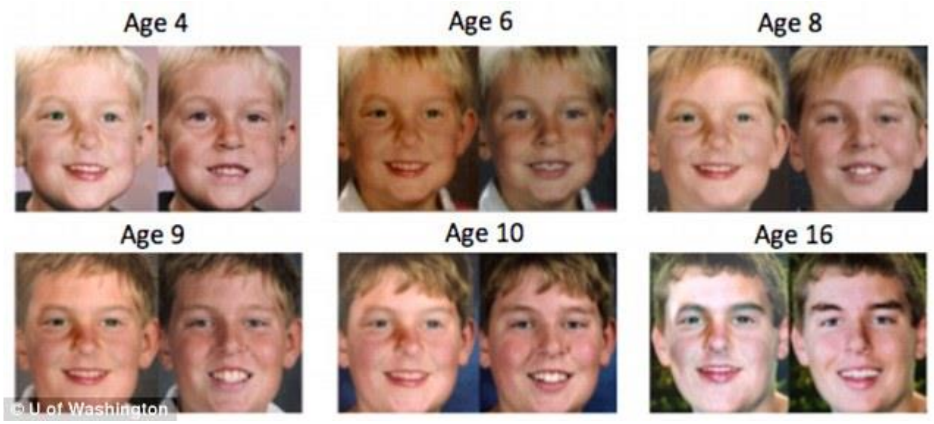
- Clustering – **Type** questions [Pic1=Pic2?]

- Max or Top-k – **Value** questions [Pic1>Pic2?]

SELECT most-recent(photo)
FROM photoDB
WHERE singlePerson(photo)
GROUP BY Person(photo)



photoDB



Why Crowdsource this

- Photo processing software can isolate distinct faces
- When tagged, software can group faces
- But, not all faces in photoDB are tagged
- Grouping of photos ,20 years apart difficult
- Timestamp on photos not trustworthy

Related Work – Focus Points

- Which element has maximum likelihood of being max?
- Which future comparisons are most effective?
- Evaluate heuristics by tuning time, cost and quality
- Sorting using Qurk – Batching and Rating pairs
- Focus on theoretical results

Formalising the Problem

- Consider n elements *x1,x2,…xn*
  - Each *xi* has type associated *type(xi)*
  - Each *xi* has value associated *value(xi)*
  - *J* distinct types or clusters
  - *J* clusters balanced if there are *n/J* elements per cluster

- **photoDB** - *J* is the number of distinct faces
- Since some people appear in more photos *J* clusters not balanced

Questions to the crowd


[type(x1)=type(x2)]

Do these pictures contain the same person? ✓

Who is this person? ✗


[val(x1)>val(x2)]

Which is the most recent picture of this person? ✓

What is the age in each picture? ✗

Handling Human Error

- **Constant Error Model** : Based on assumption that all questions are answered correctly with constant probability (> 0.5). Better than random Yes/No answer (=0.5)

- **Variable Error Model** : Based on how close data elements are in ordering of interest

Eg: Baby and high school graduate - probability of error low.

Pictures taken 1 week apart - probability of error high

Given two distinct elements x_i, x_j such that $x_i > x_j$

Probability of error i.e..

Pr[x_j is returned as the larger element] $\leq \frac{1}{f(j-i)}$ **-e (1)**

..where *f*(1) >2 and e>0

When items are next to each other in order, (*j*-*i*) =1

For some *(j-i) , f(j-i) =2*

then Variable Error Model = Constant Error model

Finding max

- When no errors **(n-1)** questions asked to find max
- Constant error model : $\Pr \geq [1-\delta]$

$$\boldsymbol{O\left(n \log \frac{1}{\delta}\right)}$$

- Variable error model : $\Pr \geq [1-\delta]$

$$\boldsymbol{O\left(\frac{n}{\delta} \log \frac{1}{\delta}\right)}$$

Max and top-k – Tournament Approach

Algorithm 1 Algorithm for finding the maximum element.

1: – Choose a random permutation Π of the elements $x_1, \cdots, x_n$.
2: **for** levels $L = 1$ to $\log n$ in the comparison tree **do** { *leaves are in level 0, the root is in level* $\log n$* }
3: – If $L \leq \log X$ (lower $\log X$ levels), do one comparison at each internal node. Propagate the winners to the level above.
4: – If $L > \log X$ (upper $\log \frac{n}{X}$ levels), do N_L comparison at each internal node. Take majority vote and propagate the winners to the level above.
5: **end for**
6: **return** The element at the root node of the comparison tree.

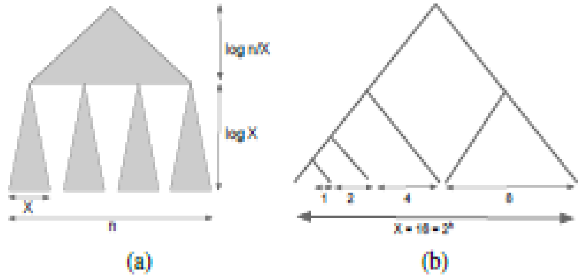
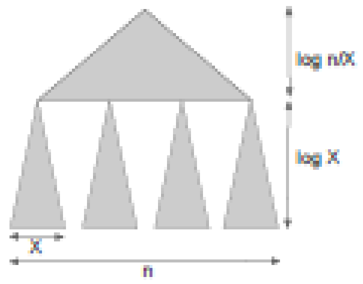


Figure 1: (a) General framework of Algorithm 1 with a comparison tree, (b) Amplified single X-tree with X nodes and (lower) $\log X$ levels.

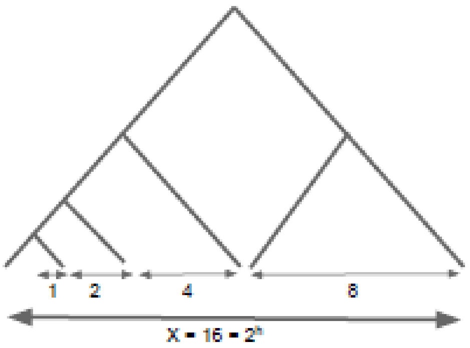
Analysis of Upper Levels

- Analyse upper $\log \frac{n}{X}$ levels
- Apply constant error model
- $\Pr \geq \frac{1}{2} + e$ finding max
- $O\left(\frac{n}{X} \log \frac{1}{\delta}\right)$ value questions



Analysis of Lower Levels

- Partition $\boldsymbol{x_1} \ldots \boldsymbol{x_n}$ into block of size $\boldsymbol{X}$
- Each block forms a comparison tree called ***X-Tree***
- $\frac{n}{X}$ such X-Trees
- Assuming $\boldsymbol{x_1}$ is max, consider the left most path
- $\Pr[x_1 \ \textit{never eliminated in comparison}] \geq [1 - 5\delta]$



Sample X-Tree

Complexity at Lower Levels

- Number of questions in *log X* is bounded by n
- *It can be shown that* $\frac{n}{X} = \frac{n}{2^{\log X}} = o\left(\frac{n}{\partial}\right)$
- *Combining this for overall complexity, we get*

$$\boldsymbol{n + O\left(\frac{n}{\partial}\log\frac{1}{\partial}\right)}$$

..number of questions

- *Number of questions improve for linear and exponential functions in variable error model*

Finding top-k

- Approach similar as finding max (tournament approach)
- Assume k< n/2
- Split tree into lower log X levels and upper log n/X levels
- In the lower levels , do one comparison at each internal node
- Each top k element winner in their X-Tree

Number of Questions asked for top-k

- Use the corollary to the theorem defining complexity for finding max
- $\boldsymbol{n} + \boldsymbol{O}\left(\frac{nk}{\partial}\log\frac{k}{\partial}\right) + \boldsymbol{O}\left(\frac{k^2}{\partial}\log\frac{k}{\partial}\right)$ value questions
- Number of questions reduces when the function used in variable error model is linear or exponential

Clustering

- Motivated by Group – By queries in SQL
- Do two photos capture the same person or same place
- Finding bound for the number of questions necessary and sufficient to find ***J*** clusters
- We use the **Constant Error Model**

$$Pr[answering\ correctly] \geq \frac{1}{2}+e$$

Clustering Algorithm

- Pick first element y in list L
- Iterate over the list, asking question type(x)= type(y)
- $O\left(\frac{1}{e^2}\left(\log\frac{n}{\delta}\right)\right)$ such questions
- For all matching items , add to list P and delete them from L

Algorithm 2 Algorithm for clustering with only type questions (given n elements, and the values of $\epsilon, \delta > 0$))

1: – List the elements in arbitrary order L.
2: – Initialize a set for clusters $P = \emptyset$.
3: **while** L is not empty **do**
4: Let y be the first element in L.
5: Find elements with the same type as y among the remaining elements in L as follows: For each remaining element x in L, ask the type question $\texttt{type}(x) = \texttt{type}(y)$ $O(\frac{1}{\epsilon^2}(\log\frac{n}{\delta}))$ times. If the majority of the answers are "yes", x, y are decided to have the same type; otherwise they are decided to have different types.
6: Collect all elements of the same type, make a cluster C, add to P, and delete these elements from L.
7: **end while**
8: **return** the clusters in P.

Analysis of Clustering Algorithm

$$\Pr[\text{determining clusters correctly }] \geq 1 - \frac{\delta}{n}$$

- Outer while loop would be run *J* times and J $\leq n$ [or $O(J)$]
- Inner for loop at most n times . Hence total number of questions

$$O\left(\frac{n}{e^2}\left(\log\frac{n}{\delta}\right)\right)$$

Total Complexity $\boldsymbol{O\left(nJ\left(log\frac{n}{\delta}\right)\right)}$

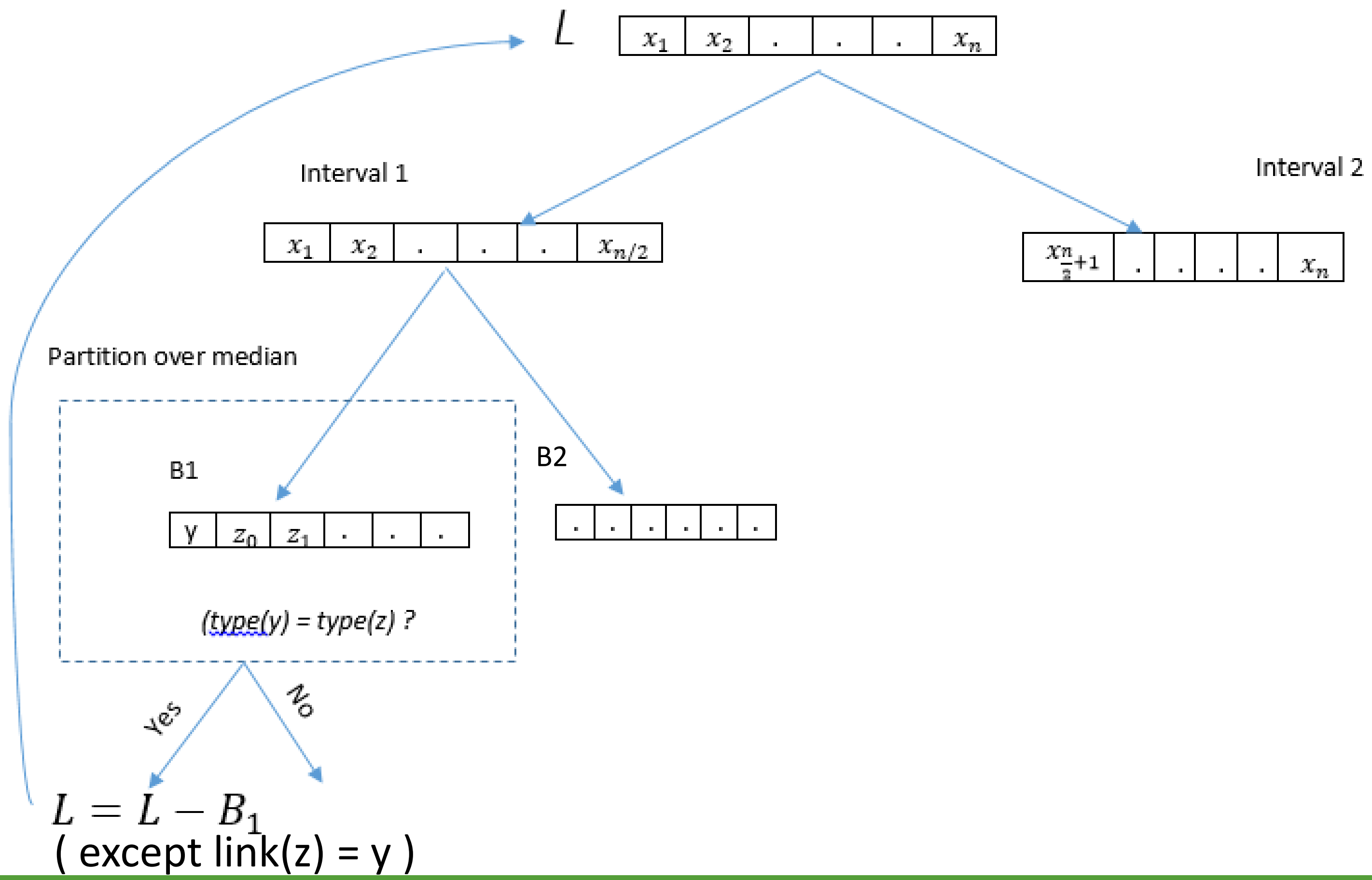
Clustering with Correlated Types and Values

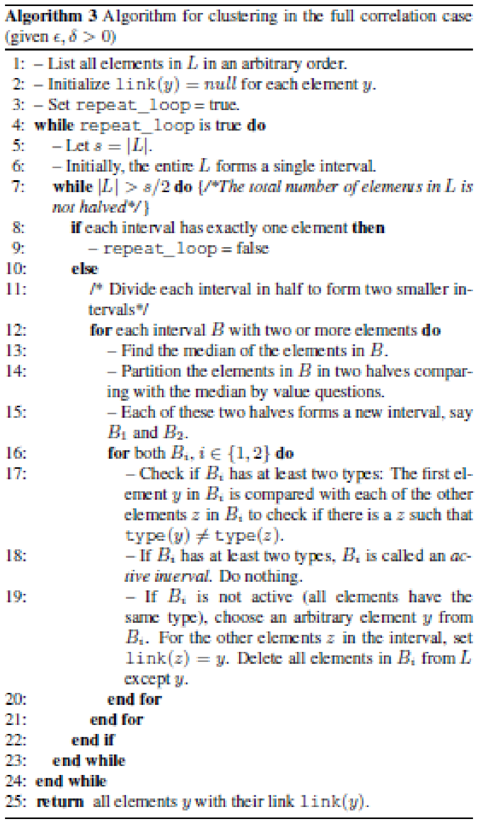
- Exploiting the fact that values and types could be correlated for datasets .
- Eg: Prices(value) of hotels could be correlated to Ratings (type)

Name	Hotel A1	Hotel A2	Hotel A3	Hotel B1	Hotel B2	Hotel B3	Hotel C1	Hotel C2	Hotel C3
Quality	3 -Star	3 -Star	3 -Star	4 -Star	4 -Star	4 -Star	5- Star	5-Star	5-Star
Price	50$	55$	60$	100$	110$	120$	200$	220$	240$

- Fails for certain datasets like photoDB. (All older people are not the same person)

Algorithm for Correlated Clustering





Algorithm 3 Algorithm for clustering in the full correlation case (given $\epsilon, \delta > 0$)

– List all elements in L in an arbitrary order.
– Initialize $\texttt{link}(y) = null$ for each element y.
– Set `repeat_loop` = true.
while `repeat_loop` is true **do**
  – Let $s = |L|$.
  – Initially, the entire L forms a single interval.
  while $|L| > s/2$ **do** {/**The total number of elements in L is not halved**/ }
    if each interval has exactly one element **then**
      – `repeat_loop` = false
    else
      /* Divide each interval in half to form two smaller intervals*/
      for each interval B with two or more elements **do**
        – Find the median of the elements in B.
        – Partition the elements in B in two halves comparing with the median by value questions.
        – Each of these two halves forms a new interval, say B_1 and B_2.
        for both B_i, $i \in \{1, 2\}$ **do**
          – Check if B_i has at least two types: The first element y in B_i is compared with each of the other elements z in B_i to check if there is a z such that $\texttt{type}(y) \neq \texttt{type}(z)$.
          – If B_i has at least two types, B_i is called an *active interval*. Do nothing.
          – If B_i is not active (all elements have the same type), choose an arbitrary element y from B_i. For the other elements z in the interval, set $\texttt{link}(z) = y$. Delete all elements in B_i from L except y.
        end for
      end for
    end if
  end while
end while
return all elements y with their link $\texttt{link}(y)$.

Number of questions asked

- Finding the median and partition for jth interval : $O\left(n_j\right)$ value questions
- Comparing first element of interval with other elements using type questions: $O\left(n_j\right)$

$$\sum_{i=1}^{b} \boldsymbol{O}\left(\boldsymbol{n_j}\right) = \boldsymbol{O}(\boldsymbol{s})$$

- Inner while loop to find active intervals : $O\ (\log J)$
- Outer while loop for list size n: $\mathrm{Q(n)} = \mathrm{Q}\left(\frac{n}{2}\right) + O\ (n\ \log J)$

$$= O\ (n\ \log J)$$

Handling erroneous answers

- Answer is correct with probability $\frac{1}{2}+e$
- Each type or value question to be performed $O\left(\frac{1}{e^2}\left(\log\frac{n}{\delta}\right)\right)$
- Hence complexity is $O\ (n\ \log J) * O\left(\log\frac{n}{\delta}\right)$

Max/Top-k each cluster

- Can be achieved by combining previous results
- Small modification to clustering algorithm
- Ask type question to compare elements
- Additionally ask value question
- Just retain the element with larger value

SELECT most-recent(photo)

FROM photoDB

WHERE singlePerson(photo)

GROUP BY Person(photo)

Conclusion

- Discussed max/top-k and clustering problems
- Proposed efficient algorithms to reduce number of type and value questions and reduce cost
- Proposed the variable error model which asks fewer questions than the constant error model
- Studied that fewer questions are needed when there is a correlation between type and value

Future work

- Interesting to have a ‘value-based’ variable error model
- Reducing probability of errors when a pre-defined budget on number of comparisons is given
- Minimize number of rounds of interaction with crowd

Thank you!