Crowdsourced Algorithms in Data Management

DASFAA 2014 Tutorial

Dongwon Lee, Ph.D.

(Thanks to Jongwuk Lee, Ph.D.)

Penn State University, USA
dongwon@psu.edu
Latest Version

- Latest version of this DASFAA14 tutorial is available at:

  http://bit.ly/1h6wHAS
TOC

- Crowdsourcing Basics
  - Definition
  - Apps
- Crowdsourced Algorithms in DB
  - Sort
  - Top-1
  - Top-k
  - Select
  - Count
  - Join
“Collective intelligence can be brought to bear on a wide variety of problems, and complexity is no bar… conditions that are necessary for the crowd to be wise: diversity, independence, and … decentralization”
“Crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. … The crucial prerequisite is the use of the open call format and the large network of potential laborers…”

http://www.wired.com/wired/archive/14.06/crowds.html
Daren Brabhan, 2013

“Crowdsourcing as an online, distributed problem-solving and production model that leverages the collective intelligence of online communities to serve specific organizational goals”
“Human computation is simply computation that is carried out by humans…

Crowdsourcing can be considered a method or a tool that human computation systems can use…”

By Edith Law & Luis von Ahn
Game with a Purpose: GWAP

- Luis von Ahn @ CMU
- Eg,
  - ESP Game → Google Image Labeler
  - Foldit
  - Duolingo: crowdsourced language translation
Eg, Francis Galton, 1906

Weight-judging competition:
1,197 (mean of 787 crowds) vs. 1,198 pounds (actual measurement)
Eg, StolenSidekick, 2006

- A woman lost a cellphone in a taxi
- A 16-year-old girl ended up having the phone
  - Refused to return the phone
- Evan Guttman, the woman’s friend, sets up a blog site about the incident
  - http://stolensidekick.blogspot.com/
  - http://www.evanwashere.com/StolenSidekick/
  - Attracted a growing amount of attention → the story appeared in Digg main page → NY Times and CNN coverage → Crowds pressure on police …
- NYPD arrested the girl and re-possessed the phone

http://www.nytimes.com/2006/06/21/nyregion/21sidekick.html?_r=0
Eg, “Who Wants to be a Millionaire”

Asking the audience usually works ➔ Audience members have diverse knowledge that can be coordinated to provide a correct answer in sum.
Eg, DARPA Challenge, 2009

- To locate 10 red balloons in arbitrary locations of US
- Winner gets $40K
- MIT team won the race with the strategy:
  - 2K per balloon to the first person, A, to send the correct coordinates
  - 1K to the person, B, who invited A
  - 0.5K to the person, C, who invited B, …
Eg, Threadless.com

- Sells t-shirts, designed/voted by crowds
- Artists whose designs are chosen get paid
Eg, reCAPTCHA

The Norwich line steamboat train, from New-London for Boston, this morning ran off the track seven miles north of New-London.

As of 2012

Captcha: 200M every day
ReCaptcha: 750M to date
Crowdfunding, started in 2009
Project creators choose a deadline and a minimum funding goal
- Creators only from US, UK, and Canada
Donors pledge money to support projects, in exchange of non-monetary values
- Eg, t-shirt, thank-u-note, dinner with creators
- Donors can be from anywhere
Eg, Pebble, smartwatch
- 68K people pledged 10M
Crowdsourcing landscape Beta v2

For definitions, analysis, free book chapters, and other crowdsourcing resources go to:
www.resultsfromcrowds.com

Excerpted from
Getting Results From Crowds
by Ross Dawson and Steve Byghall

Note: examples only; see website for full list of crowdsourcing services

http://www.resultsfromcrowds.com/features/crowdsourcing-landscape/
What is Crowdsourcing?

- Many definitions
- A few characteristics
  - Online and distributed
  - Open call & right incentive
  - Diversity and independence
  - Top-down & bottom-up

- Q: What are the **computational** issues in crowdsourcing?
  - **Micro-tasks** for large crowds
What is Computational Crowdsourcing?

- Focus on computational aspect of crowdsourcing
- Mainly use micro-tasks
- Algorithmic aspect
- Optimization problem with three parameters

When to use Computational Crowdsourcing?

- Machine can’t do the task well
- Large crowds can do it well
- Task can be split to many micro-tasks
Computational Crowdsourcing

- Requesters
  - People submit some tasks
  - Pay **rewards** to workers

- Marketplaces
  - Provide crowds with tasks

- Crowds
  - Workers perform tasks

<table>
<thead>
<tr>
<th>Find an outlier among three images</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /> <img src="image2.png" alt="Image 2" /> <img src="image3.png" alt="Image 3" /></td>
</tr>
</tbody>
</table>

*Submit tasks* ↓ *Collect answers*

*Find tasks* ↓ *Return answers*
Crowdsourcing Marketplaces

- Platforms for posting/performing (often micro) tasks

- Those who want to have tasks done via crowdsourcing → Requesters
  - Eg, companies, researchers

- Those who want to perform tasks for monetary profits → Workers
  - Eg, individuals for extra income
Crowdsourcing Platforms

- Notables ones:
  - Mechanical Turk (AMT)
  - CrowdFlower
  - CloudCrowd
  - Clickworker
  - SamaSource
AMT: mturk.com

Workers

Mechanical Turk is a marketplace for work. We give businesses and developers access to an on-demand, scalable workforce. 

We have over 200,000 HITs available. View them now.

Make Money by working on HITs

HITs - Human Intelligence Tasks - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work

Find an interesting task

Work

Earn money

Find HITs Now

Requesters

Get Results from Mechanical Turk Workers

As a Mechanical Turk Requester you:

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Fund your account

Load your tasks

Get results

Get Started
AMT (cont)

- **Workers**
  - Register w. credit account (only US workers can register as of 2013)
  - Bid to do tasks for earning money

- **Requesters**
  - First deposit money to account
  - Post tasks
    - Task can specify a qualification for workers
  - Gather results
  - Pay to workers if results are satisfactory
AMT (cont)

- Tasks
  - Called HIT (Human Intelligence Task)
  - Micro-task

- Eg
  - Data cleaning
  - Tagging / labeling
  - Sentiment analysis
  - Categorization
  - Surveying
  - Photo moderation
  - Transcription

Translation task
# AMT: HIT List

![HIT List](image)

## Workers qualification

<table>
<thead>
<tr>
<th>Requester</th>
<th>HIT Expired Date</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>rohitDd</td>
<td>Oct 25, 2013</td>
<td>$0.00</td>
</tr>
<tr>
<td>John</td>
<td>Oct 10, 2013</td>
<td>$0.06</td>
</tr>
<tr>
<td>CrowCrow</td>
<td>Oct 8, 2013</td>
<td>$0.07</td>
</tr>
<tr>
<td>CrowCrow</td>
<td>Oct 9, 2013</td>
<td></td>
</tr>
</tbody>
</table>

### Qualifications Required:
- Location is not VN
- Location is not TR
- Location is not RO
- Location is not PK
- Location is not PH
- Location is not IN
- Location is not ID
- Location is not HK
- HIT approval rate (%) is greater than 96
Can You Find the Provided Phone Number or Street Address on this Website?

Overview
In this task, you’ll be provided a web page for a business, including its name, address, and phone number. Your goal is to answer a few questions about the business on the web page.

IMPORTANT: Sometimes the business will have multiple locations, and you will have to search the website for the specific business that we provide in order to verify the website.

Step by step instructions:
• Click the link to go to the provided site.
• First, please tell us whether or not the name of the business on the provided website is a close or identical match to the name of the business shown at the top of the page.
• Next, please tell us whether the provided business has
• Please be sure to click the appropriate option if the site is not correct.

Wrinkles Day Spa
Phone: +61893455333
Street: Shop 5a Stirling Central Shopping Centre, 478 Wanneroo Rd
City: Westminster
State: WA
Postalcode (Zip): 6061
Country Code: AU

Click here to visit the website.

Is the name of the business on the web page similar or identical to 'Wrinkles Day Spa'?
○ Yes: the name of the business is similar to Wrinkles Day Spa
○ Yes: the name of the business is nearly identical to Wrinkles Day Spa
○ No: the name is very different from Wrinkles Day Spa
○ For the first option, the street number does not need to match, just the street, Shop 5a Stirling Central Shopping Centre, 478 Wanneroo Rd.
Eg, Text Transcription [Miller-13]

- **Problem**: one person can’t do a good transcription
- **Key idea**: iterative improvement by many workers

Greg Little *et al.* “Exploring iterative and parallel human computation processes.” HCOMP 2010
Eg, Text Transcription [Miller-13]

Please improve the transcription of this handwriting.
People will vote whether to approve your changes.

You (?) (?) work. (?) (?) work (not) time. I (?) (?) a few grammatical mistakes. Overall your writing style is a bit too phoney. You (?) have good points, but they got lost amidst the writing. (signature)

improvement $0.05
Eg, Text Transcription [Miller-13]

You (misspelled) (several) (words) (work). (?) (?) (?) work next (time). I also notice a few grammatical mistakes. Overall your writing style is a bit too (phony). You do (?) have good (points), but they got lost amidst the (writing). (signature)

You (?) (?) (?) work. (?) (?) (?) work not (time). I (?) (?) a few grammatical mistakes. Overall your writing style is a bit too (phony). You do (?) have good (points), but they got lost amidst the (writing). (signature)

3 votes @ $0.01
“You (misspelled) (several) (words). Please spellcheck your work next time. I also notice a few grammatical mistakes. Overall your writing style is a bit too phoney. You do make some good (points), but they got lost amidst the (writing). (signature)"

According to our ground truth, the highlighted words should be “flowery”, “get”, “verbiage” and “B-” respectively.

After 9 iterations
I had intended to hit the nail, but I’m not a very good aim it seems, and I ended up hitting my thumb. This is a common occurrence, I know, but it doesn’t make me feel any less ridiculous having done it myself. My new strategy will involve lightly tapping the nail while holding it, until it is embedded into the wood enough that the wood itself is holding it straight, and then I’ll remove my hand and pound carefully away. We’ll see how this goes.

Another example: blurry text

After 8 iterations
Eg, Computer Vision [Li-HotDB12]

- How similar is the artistic style?

*Human and Machine Detection of Stylistic Similarity in Art. Adriana Kovashka and Matthew Lease. CrowdConf 2010*
Crowdsourcing Basics
- Definition
- Apps

Crowdsourced Algorithms in DB
- Sort
- Top-1
- Top-\(k\)
- Select
- Count
- Join
Crowdsourcing DB Projects

- CDAS @ NUS
- CrowdDB @ UC Berkeley & ETH Zurich
- MoDaS @ Tel Aviv U.
- Qurk @ MIT
- sCOOP @ Stanford & UCSC
Eg, CrowdDB System

- Crowd-enabled databases
  - Hybrid human/machine databases
  - Building a database engine that can dynamically **crowdsource** certain operations

```
CREATE TABLE Department ( 
  university STRING, 
  name STRING, 
  url CROWD STRING, 
)
```

[Franklin-SIGMOD11]
Preliminaries: 3 Factors

- **Latency (or execution time)**
  - Worker pool size
  - Job attractiveness

- **Monetary cost**
  - # of questions
  - # of workers
  - Cost per question

- **Quality of answers**
  - Worker maliciousness
  - Worker skills
  - Task difficulty

How long do we wait for?

Cost

Latency

Quality

How much $$$ does we spend?

How much is the quality of answers satisfied?
Preliminaries: Size of Comparison

- Diverse forms of questions in a HIT
- Different sizes of comparisons in a question
  - Eg, Binary question
    - $s = 2$
  - Eg, $N$-ary question
    - $s = N$

Which is better?

Which is the best?
Preliminaries: Batch

- Repetitions of questions within a HIT
- Eg, two $n$-ary questions (batch factor $b=2$)
Preliminaries: Response ($r$)

- # of human responses seeked for a HIT

Which is better?

- $W_1$
- $W_2$
- $W_3$

$r = 3$
Preliminaries: Round (= Step)

- Algorithms are executed in rounds
- # of rounds ≈ latency

Which is better?

Round #1

Parallel Execution

Round #2

Sequential Execution
Sort Operation

- Rank $N$ items using crowdsourcing with respect to the constraint $C$
- Eg, $C$ as “Representative,” “Dangerous,” “Beautiful”

```sql
SELECT * 
FROM ImageTable AS I 
WHERE I.Date > 2014 AND I.loc = “NY” 
ORDER BY CrowdOp(“Representative”) 
```
Naïve Sort

- Assuming pair-wise comparison of 2 items
  - Eg, “Which of two images is better?”
- Cycle: A > B, B > C, and C > A
- If no cycle occurs
  - Naïve all pair-wise comparisons takes \( \binom{N}{2} \) comparisons
  - Optimal # of comparison is \( O(N \log N) \)
- If cycle exists
  - More comparisons are required
Sort [Marcus-VLDB11]

- Proposed 3 crowdsourced sort algorithms

- **#1: Comparison-based Sort**
  - Workers rank $S$ items ($S \subseteq \mathcal{N}$) per HIT
  - Each HIT yields $\binom{S}{2}$ pair-wise comparisons
  - Build a DAG using all pair-wise comparisons from all workers
    - If $i > j$, then add an edge from $i$ to $j$
  - Break a cycle in the DAG: “head-to-head”
    - Eg, If $i > j$ occurs 3 times and $i < j$ occurs 2 times, keep only $i > j$
  - Perform a topological sort in the DAG
Sort [Marcus-VLDB11]

There are 2 groups of squares. We want to order the squares in each group from smallest to largest.

- Each group is surrounded by a dotted line. Only compare the squares within a group.
- Within each group, assign a number from 1 to 7 to each square, so that:
  - 1 represents the smallest square, and 7 represents the largest.
  - We do not care about the specific value of each square, only the relative order of the squares.
  - Some groups may have less than 7 squares. That is OK: use less than 7 numbers, and make sure they are ordered according to size.
  - If two squares in a group are the same size, you should assign them the same number.
Sort [Marcus-VLDB11]

- N=5, S=3
Sort [Marcus-VLDB11]

- N=5, S=3

A

B

C

D

E

W1

W2

\[ \text{\textbf{A}} \]

\[ \text{\textbf{B}} \]

\[ \text{\textbf{C}} \]

\[ \text{\textbf{D}} \]

\[ \text{\textbf{E}} \]

\[ \text{\textbf{A}} \]

\[ \text{\textbf{B}} \]

\[ \text{\textbf{C}} \]

\[ \text{\textbf{D}} \]

\[ \text{\textbf{E}} \]
Sort [Marcus-VLDB11]

- N=5, S=3

A
W1
B
W2
C
D
W3
E

Diagram:

A → B
A → C
A → D
B → E
C → D
D → E

Weight:

1 1 1
1 1 1
1 2 1
1 1 1
Sort [Marcus-VLDB11]

- N=5, S=3

```
A  W1  >  W2  >  W3  >  W4
A  B  C  D  E
```

![Graph](image)
Sort [Marcus-VLDB11]

- N=5, S=3

Topological Sort

A > B > C > E > D
#2: Rating-based Sort

- $W$ workers rate each item along a numerical scale
- Compute the mean of $W$ ratings of each item
- Sort all items using their means
- Requires $W*N$ HITs: $O(N)$

<table>
<thead>
<tr>
<th>Worker</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>4</td>
</tr>
<tr>
<td>W2</td>
<td>3</td>
</tr>
<tr>
<td>W3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Worker</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>1</td>
</tr>
<tr>
<td>W2</td>
<td>2</td>
</tr>
<tr>
<td>W3</td>
<td>1</td>
</tr>
</tbody>
</table>

Mean rating:

- 1.3
- 3.6
- 8.2
Sort [Marcus-VLDB11]

There are 2 squares below. We want to rate squares by their size.

- For each square, assign it a number from 1 (smallest) to 7 (largest) indicating its size.
- For perspective, here is a small number of other randomly picked squares:

![Diagram of squares with size ratings]

Submit
Sort [Marcus-VLDB11]

- #3: Hybrid Sort
  - First, do rating-based sort $\rightarrow$ sorted list $L$
  - Second, do comparison-based sort on $S$ ($S \subset L$)
  - How to select the size of $S$
    - Random
    - Confidence-based
    - Sliding window
Sort [Marcus-VLDB11]

- Rank correlation btw. Comparison vs. rating
- Worker agreement

Bar chart showing Kappa/Tau values for Q1 to Q5.
Sort [Marcus-VLDB11]
Top-1 Operation

- Find the top-1, either MAX or MIN, among $N$ items w.r.t. “something”

- Objective
  - Avoid sorting all $N$ items to find top-1
Top-1 Operation

- Examples
  - [Venetis-WWW12] introduces the bubble max and tournament-based max in a parameterized framework
  - [Guo-SIGMOD12] studies how to find max using pair-wise questions in the tournament-like setting and how to improve accuracy by asking more questions
Max [Venetis-WWW12]

- Introduced two Max algorithms
  - Bubble Max
  - Tournament Max
- Parameterized framework
  - \( s_i \): size of sets compared at the \( i \)-th round
  - \( r_i \): # of human responses at the \( i \)-th round

Which is better?

- \( s_i = 2 \)
- \( r_i = 3 \)

Which is the best?

- \( s_i = 3 \)
- \( r_i = 2 \)
Max [Venetis-WWW12]

- Bubble Max Case #1

- $N = 5$
- Rounds = 3
- # of questions = $r_1 + r_2 + r_3 = 11$

$s_1 = 2$
$r_1 = 3$

$s_2 = 3$
$r_2 = 3$

$s_3 = 2$
$r_3 = 5$
Max [Venetis-WWW12]

- Bubble Max Case #2

  - $N = 5$
  - Rounds = 2
  - # of questions = $r_1 + r_2 = 8$

\[ s_1 = 4 \quad r_1 = 3 \]
\[ s_2 = 2 \quad r_2 = 5 \]
Max [Venetis-WWW12]

- Tournament Max
  - $N = 5$
  - Rounds = 3
  - # of questions
    $$= r_1 + r_2 + r_3 + r_4 = 10$$
  $s_1 = 2$
  $r_1 = 1$
  $s_2 = 2$
  $r_2 = 1$
  $s_3 = 2$
  $r_3 = 3$
  $s_4 = 2$
  $r_4 = 5$
Max [Venetis-WWW12]

- How to find optimal parameters?: $s_i$ and $r_i$
- Tuning Strategies (using Hill Climbing)
  - Constant $s_i$ and $r_i$
  - Constant $s_i$ and varying $r_i$
  - Varying $s_i$ and $r_i$
Max [Venetis-WWW12]

- Bubble Max
  - Worst case: with $s_i=2$, $O(N)$ comparisons needed

- Tournament Max
  - Worst case: with $s_i=2$, $O(N)$ comparisons needed

- Bubble Max is a special case of Tournament Max
Max [Venetis-WWW12]
Max [Venetis-WWW12]
Max [Venetis-WWW12]

![Tournament Graph](image)

- **Budget**
  - $B = 1500$
  - $B = 4000$
  - $B = 5500$

**Steps**
- **Step 1**: Low performance
- **Step 2**: Moderate performance
- **Step 3**: High performance
- **Step 4**: Peak performance

**Performance Index**
- $r_i$
Top-\(k\) Operation

- Find top-\(k\) items among \(N\) items w.r.t. “something”

- Top-\(k\) list vs. top-\(k\) set

- Objective
  - Avoid sorting all \(N\) items to find top-\(k\)
Top-\(k\) Operation

- Examples
  - [Davidson-ICDT13] investigates the variable user error model in solving top-\(k\) list problem
  - [Polychronopoulos-WebDB13] proposes tournament-based top-\(k\) set solution
Top-$k$ Operation

- Naïve solution is to “sort” $N$ items and pick top-$k$ items
- Eg, $N=5$, $k=2$, “Find two best Bali images?”
  - Ask $\binom{5}{2} = 10$ pair-wise questions to get a total order
  - Pick top-2 images
Top-\(k\): Tournament Solution (\(k = 2\))

- **Phase 1:** **Building a tournament tree**
  - For each comparison, only winners are promoted to the next round
Top-$k$: Tournament Solution ($k = 2$)

- **Phase 1:** Building a tournament tree
  - For each comparison, only winners are promoted to the next round
Top-$k$: Tournament Solution ($k = 2$)

- Phase 1: **Building a tournament tree**
  - For each comparison, only winners are promoted to the next round

Round 1
Round 2
Round 3

Total, 4 questions with 3 rounds
Phase 2: **Updating a tournament tree**

- **Iteratively** asking pair-wise questions from the bottom level
Top-$k$: Tournament Solution ($k = 2$)

- **Phase 2:** Updating a tournament tree
  - **Iteratively** asking pair-wise questions from the bottom level

Round 4
Top-\(k\): Tournament Solution (\(k = 2\))

- Phase 2: **Updating a tournament tree**
  - **Iteratively** asking pair-wise questions from the bottom level

Total, 6 questions
With 5 rounds
Top-k: Tournament Solution

- This is a top-k list algorithm
- Analysis

<table>
<thead>
<tr>
<th></th>
<th>( k = 1 )</th>
<th>( k \geq 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td># of questions</td>
<td>( O(n) )</td>
<td>( O(n + k \lceil \log_2 n \rceil) )</td>
</tr>
<tr>
<td># of rounds</td>
<td>( O(\lceil \log_2 n \rceil) )</td>
<td>( O(k \lceil \log_2 n \rceil) )</td>
</tr>
</tbody>
</table>

- If there is no constraint for the number of rounds, this tournament sort yields the optimal result
Top-\(k\) [Polychronopoulous-WebDB13]

- **Top-\(k\) set** algorithm
  - Top-\(k\) items are “better” than remaining items
  - Capture NO ranking among top-\(k\) items
  - Tournament-based approach
  - Can become a Top-\(k\) **list** algorithm
    - Eg, Top-\(k\) set algorithm, followed by [Marcus-VLDB11] to sort \(k\) items
Algorithm

- Input: \( N \) items, integer \( k \) and \( s \) (ie, \( s > k \))
- Output: top-\( k \) items
- Procedure:
  - \( O \leftarrow \) \( N \) items
  - While \( |O| > k \)
    - Partition \( O \) into disjoint subsets of size \( s \)
    - Identify top-\( k \) items in each subset of size \( s \): \( s\text{-}rank(s) \)
    - Merge all top-\( k \) items into \( O \)
  - Return \( O \)

Effective only when \( s \) and \( k \) are small

- Eg, \( s\text{-}rank(20) \) with \( k=10 \) won’t work well
Top-\(k\) [Polychronopoulous-WebDB13]

- Eg, \(N=10, s=4, k=2\)
Top-$k$ [Polychronopoulous-WebDB13]

- $s$-rank($s$)

\[\text{// workers rank } s \text{ items and aggregate}\]

- Input: $s$ items, integer $k$ (ie, $s > k$), $w$ workers
- Output: top-$k$ items among $s$ items
- Procedure:
  - For each of $w$ workers
    - Rank $s$ items ≈ comparison-based sort [Marcus-VLDB11]
  - Merge $w$ rankings of $s$ items into a single ranking
    - Use median-rank aggregation [Dwork-WWW01]
  - Return top-$k$ item from the merged ranking of $s$ items
Top-$k$ [Polychronopoulous-WebDB13]

- Eg, s-rank(): $s=4$, $k=2$, $w=3$

<table>
<thead>
<tr>
<th>W1</th>
<th>W2</th>
<th>W3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Median Ranks</th>
<th>4</th>
<th>2</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>

Top-2
Top-$k$ [Polychronopoulous-WebDB13]

- How to set $\#$ of workers, $w$, per s-rank()
  - Basic
    - same $\#$ to all s-rank()
  - Adaptive
    - 3-level assignments: low, medium, high
    - if rankings from workers disagree, allowing more workers would improve the accuracy
      - Needs more rounds for improvement
Top-$k$ [Polychronopoulous-WebDB13]

- Basic vs. Adaptive

200 items, $k=5$, $s=10$, 40% error rate, default spam

![Graph showing average error vs. budget for different methods: basic, adapt. & spec., adapt. & spec. & redist., adaptive](image)
Top-$k$ [Polychronopoulous-WebDB13]

- Comparison to Sort [Marcus-VLDB11]
Top-$k$ [Polychronopoulous-WebDB13]

- Comparison to Max [Venetis-WWW12]
**Top-$k$ [Polychronopoulous-WebDB13]**

- AMT result

![Graph](image)

160 shapes, $k=5$, $s=10$, budget 600 HITs

- basic
- adaptive
- adaptive & speculative

Difficulty levels:
- Easy
- Medium
- Hard

Average error vs. Difficulty.
Select Operation

- Given $N$ items, select $k$ items that satisfy a predicate $P$
- $\approx$ Filter, Find, Screen, Search
Select Operation

Examples

- [Yan-MobiSys10] uses crowds to find an image relevant to a query
- [Parameswaran-SIGMOD12] develops filters
- [Franklin-ICDE13] efficiently enumerates items satisfying conditions via crowdsourcing
- [Sarma-ICDE14] finds a bounded number of items satisfying predicates using the optimal solution by the skyline of cost and time
Select [Parameswaran-SIGMOD12]

- Novel grid-based visualization
Common strategies

- Always ask X questions, return most likely answer → Triangular strategy

- If X YES return “Pass”, Y NO return “Fail”, else keep asking → Rectangular strategy

- Ask until |#YES - #NO| > X, or at most Y questions → Chopped off triangle
What is the best strategy? Find strategy with minimum overall expected cost

1. Overall expected error is less than threshold
2. # of questions per item never exceeds m
Count Operation

- Given $N$ items, estimate a fraction of items $M$ that satisfy a predicate $P$

- Selectivity estimation in DB $\rightarrow$ crowd-powered query optimizers
- Evaluating queries with GROUP BY + COUNT/AVG/SUM operators
- Eg, “Find photos of females with red hairs”
  - Selectivity(“female”) $\approx 50\%$
  - Selectivity(“red hair”) $\approx 2\%$
  - Better to process predicate(“red hair”) first
Hypothesis: Humans can estimate the frequency of objects’ properties in a batch without having to explicitly label each item.

Two approaches

#1: Label Count
- Sampling theory based
- Have workers label samples explicitly

#2: Batch Count
- Have workers estimate the frequency in a batch
Count [Marcus-VLDB13]

- Label Count (via sampling)

There are 2 people below. Please identify the gender of each.

What is the gender of this person?
- male  female

What is the gender of this person?
- male  female
Count [Marcus-VLDB13]

- **Batch Count**

There are 10 people below. Please provide rough estimates for how many of the people have various properties.

About how many of the 10 people are **male**?

4

About how many of the 10 people are **female**?
Count [Marcus-VLDB13]

- Findings on accuracy
  - Images: Batch count > Label count
  - Texts: Batch count < Label count

- Further Contributions
  - Detecting spammers
  - Avoiding coordinated attacks
Join Operation

- Identify matching records or entities within or across tables
  - \( \approx \) similarity join, entity resolution (ER), record linkage, de-duplication, ...
  - Beyond the exact matching

- [Chaudhuri-ICDE06] similarity join
  - \( R \JOIN_p S, \text{where } p = \text{sim}(R.A, S.A) > t \)
  - \( \text{sim()} \) can be implemented as UDFs in SQL
  - Often, the evaluation is expensive
    - DB applies UDF-based join predicate after Cartesian product of R and S
Join Operation

Examples

- [Marcus-VLDB11] proposes 3 types of joins
- [Wang-VLDB12] generates near-optimal cluster-based HIT design to reduce join cost
- [Wang-SIGMOD13] reduces join cost further by exploiting transitivity among items
- [Whang-VLDB13] selects right questions to ask to crowds to improve join accuracy
- [Gokhale-SIGMOD14] proposes the hands-off crowdsourcing for join workflow
Join [Marcus-VLDB11]

- To join tables $R$ and $S$
- **#1: Simple Join**
  - Pair-wise comparison HIT
  - $|R||S|$ HITs needed
- **#2: Naïve Batching Join**
  - Repetition of #1 with a batch factor $b$
  - $|R||S|/b$ HITs needed
- **#3: Smart Batching Join**
  - Show $r$ and $s$ images from $R$ and $S$
  - Workers pair them up
  - $|R||S|/rs$ HITs needed
Join [Marcus-VLDB11]

Is the same celebrity in the image on the left and the image on the right?

#1 Simple Join

Yes  No
Join [Marcus-VLDB11]

Is the same celebrity in the image on the left and the image on the right?

- Yes
- No

Batch factor $b = 2$

#2 Naïve Batching Join
Join [Marcus-VLDB11]

Find pairs of images with the same celebrity

- To select pairs, click on an image on the left and an image on the right. Selected pairs will appear in the Matched Celebrities list on the left.
- To magnify a picture, hover your pointer above it.
- To unselect a selected pair, click on the pair on the left.
- If none of the celebrities match, check the I did not find any pairs check box.
- There may be multiple matches per page.

r images from R

s images from S

Matched Celebrities
To remove a pair added in error, click on the pair in the list below.

#3 Smart Batching Join
Join [Marcus-VLDB11]
Join [Marcus-VLDB11]

Last 50% of wait time is spent completing the last 5% of tasks
Join [Wang-VLDB12]

- [Marcus-VLDB11] proposed two batch joins
  - More efficient smart batch join still generates \( \frac{|R||S|}{rs} \) # of HITs
  - Eg, \( \frac{(10,000 \times 10,000)}{(20 \times 20)} = 250,000 \) HITs → Still too many!

- [Wang-VLDB12] contributes CrowdER:
  1. A hybrid human-machine join
     - #1 machine-ER prunes obvious non-matches
     - #2 human-ER examines likely matching cases
       - Eg, candidate pairs with high similarity scores
  2. Algorithm to generate min # of HITs for step #2
Join [Wang-VLDB12]

- Hybrid idea: generate candidate pairs using existing similarity measures (e.g., Jaccard)

### Table: Product Data

<table>
<thead>
<tr>
<th>ID</th>
<th>Product Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>iPad Two 16GB WiFi White</td>
<td>$490</td>
</tr>
<tr>
<td>r2</td>
<td>iPad 2nd generation 16GB WiFi White</td>
<td>$469</td>
</tr>
<tr>
<td>r3</td>
<td>iPhone 4th generation White 16GB</td>
<td>$545</td>
</tr>
<tr>
<td>r4</td>
<td>Apple iPhone 4 16GB White</td>
<td>$520</td>
</tr>
<tr>
<td>r5</td>
<td>Apple iPhone 3rd generation Black 16GB</td>
<td>$375</td>
</tr>
<tr>
<td>r6</td>
<td>iPhone 4 32GB White</td>
<td>$599</td>
</tr>
<tr>
<td>r7</td>
<td>Apple iPad2 16GB WiFi White</td>
<td>$499</td>
</tr>
<tr>
<td>r8</td>
<td>Apple iPod shuffle 2GB Blue</td>
<td>$49</td>
</tr>
<tr>
<td>r9</td>
<td>Apple iPod shuffle USB Cable</td>
<td>$19</td>
</tr>
</tbody>
</table>

Main Issue: HIT Generation Problem
Join [Wang-VLDB12]

Pair-based HIT Generation
= Naïve Batching in [Marcus-VLDB11]

Cluster-based HIT Generation
= Smart Batching in [Marcus-VLDB11]

---

** Decide Whether Two Products Are the Same (Show Instructions)**

<table>
<thead>
<tr>
<th>Product Pair #1</th>
<th>Product Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPad Two</td>
<td>iPad Two 16GB WiFi White</td>
<td>$490</td>
</tr>
<tr>
<td>iPad</td>
<td>iPad 2nd generation 16GB WiFi White</td>
<td>$469</td>
</tr>
</tbody>
</table>

Your Choice (Required)
- They are the same product
- They are different products

Reasons for Your Choice (Optional)

---

** Find Duplicate Products In the Table. (Show Instructions)**

Tips: you can (1) SORT the table by clicking headers;
(2) MOVE a row by dragging and dropping it

<table>
<thead>
<tr>
<th>Label</th>
<th>Product Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>iPad 2nd generation 16GB WiFi White</td>
<td>$469</td>
</tr>
<tr>
<td>1</td>
<td>iPad Two 16GB WiFi White</td>
<td>$490</td>
</tr>
<tr>
<td>2</td>
<td>iPhone 4 16GB White</td>
<td>$520</td>
</tr>
<tr>
<td>2</td>
<td>iPhone 4th generation White 16GB</td>
<td>$545</td>
</tr>
</tbody>
</table>

Reasons for Your Answers (Optional)

---

Submit (1 left)
Join [Wang-VLDB12]

- HIT Generation Problem
  - Input: pairs of records $P$, # of records in HIT $k$
  - Output: minimum # of HITs s.t.
    - All HITs have at most $k$ records
    - Each pair $(p_i, p_j) \in P$ must be in at least one HIT

1. Pair-based HIT Generation
   - Trivial: $P/k$ # of HITs s.t. each HIT contains $k$ pairs in $P$

2. Cluster-based HIT Generation
   - NP-hard problem $\rightarrow$ approximation solution
Join [Wang-VLDB12]

This is the minimal # of cluster-based HITs satisfying previous two conditions.
Join [Wang-VLDB12]

- Two-tiered Greedy Algorithm
  - Build a graph $G$ from pairs of records in $P$
  - $CC \leftarrow$ connected components in $G$
    - LCC: large CC with more than $k$ nodes
    - SCC: small CC with no more than $k$ nodes
  - Step 1: **Partition** LCC into SCCs
  - Step 2: **Pack** SCCs into HITs with $k$ nodes
    - Integer programming based
Join [Wang-VLDB12]

- Eg, Generate cluster-based HITs ($k = 4$)
  1. Partition the LCC into 3 SCCs
     - $\{r_1, r_2, r_3, r_7\}$, $\{r_3, r_4, r_5, r_6\}$, $\{r_4, r_7\}$
  2. Pack SCCs into HITs
     - A single HIT per $\{r_1, r_2, r_3, r_7\}$ and $\{r_3, r_4, r_5, r_6\}$
     - Pack $\{r_4, r_7\}$ and $\{r_8, r_9\}$ into a HIT
Join [Wang-VLDB12]

- **Step 1: Partition**
  - **Input:** LCC, $k$  
  - **Output:** SCCs
  - $r_{\text{max}} \leftarrow$ node in LCC with the max degree
  - $\text{scc} \leftarrow \{r_{\text{max}}\}$
  - $\text{conn} \leftarrow$ nodes in LCC directly connected to $r_{\text{max}}$
  - while $|\text{scc}| < k$ and $|\text{conn}| > 0$
    - $r_{\text{new}} \leftarrow$ node in $\text{conn}$ with max indegree (# of edges to $\text{scc}$) and min outdegree (# of edges to non-$\text{scc}$) if tie
    - move $r_{\text{new}}$ from $\text{conn}$ to $\text{scc}$
    - update $\text{conn}$ using new $\text{scc}$
  - add $\text{scc}$ into SCC
Join [Wang-VLDB12]

(a) Initialize \( \text{scc} = \{r_4\} \)
(b) \( \text{conn} = \{r_3, r_5, r_6, r_7\} \)
    Add \( r_6 \) into \( \text{scc} \)
(c) \( \text{conn} = \{r_3, r_5, r_7\} \)
    Add \( r_5 \) into \( \text{scc} \)
(d) \( \text{conn} = \{r_3, r_7\} \)
    Add \( r_3 \) into \( \text{scc} \)
(e) Output \( \text{scc} \)
(f) Output other \( \text{scc} \)
Join [Wang-VLDB12]
Join [Wang-VLDB12]
Join [Wang-SIGMOD13]

- Use the same hybrid machine-human framework as [Wang-VLDB12]
- Aim to reduce # of HITs further
- Exploit transitivity among records
Join [Wang-SIGMOD13]

- Positive transitive relation
  - If $a=b$, and $b=c$, then $a=c$

  \[
  \begin{align*}
  iPad\ 2^{nd}\ Gen &= iPad\ Two \\
  iPad\ Two &= iPad\ 2
  \end{align*}
  \]

- Negative transitive relation
  - If $a = b$, $b \neq c$, then $a \neq c$

  \[
  \begin{align*}
  iPad\ 2^{nd}\ Gen &= iPad\ Two \\
  iPad\ Two &\neq iPad\ 3
  \end{align*}
  \]
Three transitive relations

- If there exists a path from \( o \) to \( o' \) which only consists of matching pairs, then \((o, o')\) can be deduced as a matching pair.

- If there exists a path from \( o \) to \( o' \) which only contains a single non-matching pair, then \((o, o')\) can be deduced as a non-matching pair.

- If any path from \( o \) to \( o' \) contains more than one non-matching pairs, \((o, o')\) cannot be deduced.
Join [Wang-SIGMOD13]

(o₃, o₅) \(\rightarrow\) match

(o₅, o₇) \(\rightarrow\) non-match

(o₁, o₇) \(\rightarrow\) ?
Join [Wang-SIGMOD13]

- Given a pair \((o_i, o_j)\), to check the transitivity
  - Enumerate path from \(o_i\) to \(o_j\) → exponential !
  - Count # of non-matching pairs in each path

- Solution: Build a cluster graph
  - Merge matching pairs to a cluster
  - Add inter-cluster edge for non-matching pairs

\[
\begin{align*}
(o_5, o_6) & \rightarrow ? \\
(o_1, o_5) & \rightarrow ?
\end{align*}
\]
Join [Wang-SIGMOD13]

- **Problem Definition:**
  - Given a set of pairs that need to be labeled, **minimize the # of pairs** requested to crowd workers based on **transitive relations**

<table>
<thead>
<tr>
<th>ID</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>iPhone 2nd Gen</td>
</tr>
<tr>
<td>$o_2$</td>
<td>iPhone Two</td>
</tr>
<tr>
<td>$o_3$</td>
<td>iPhone 2</td>
</tr>
<tr>
<td>$o_4$</td>
<td>iPad Two</td>
</tr>
<tr>
<td>$o_5$</td>
<td>iPad 2</td>
</tr>
<tr>
<td>$o_6$</td>
<td>iPad 3rd Gen</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Object Pairs</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>($o_2$, $o_3$)</td>
<td>0.85</td>
</tr>
<tr>
<td>$p_2$</td>
<td>($o_1$, $o_2$)</td>
<td>0.75</td>
</tr>
<tr>
<td>$p_3$</td>
<td>($o_1$, $o_6$)</td>
<td>0.72</td>
</tr>
<tr>
<td>$p_4$</td>
<td>($o_1$, $o_3$)</td>
<td>0.65</td>
</tr>
<tr>
<td>$p_5$</td>
<td>($o_4$, $o_5$)</td>
<td>0.55</td>
</tr>
<tr>
<td>$p_6$</td>
<td>($o_4$, $o_6$)</td>
<td>0.48</td>
</tr>
<tr>
<td>$p_7$</td>
<td>($o_2$, $o_4$)</td>
<td>0.45</td>
</tr>
<tr>
<td>$p_8$</td>
<td>($o_5$, $o_6$)</td>
<td>0.42</td>
</tr>
</tbody>
</table>
Join [Wang-SIGMOD13]

- Labeling order matters!

\[ (o_1, o_2), (o_1, o_6), (o_2, o_6) \]

vs.

\[ (o_1, o_6), (o_2, o_6), (o_1, o_2) \]

→ Given a set of pairs to label, how to order them affects the # of pairs to deduce using the transitivity
Join [Wang-SIGMOD13]

- Optimal labeling order
  
  \[ w = <p_1, \ldots, p_{i-1}, p_i, p_{i+1}, \ldots, p_n> \]
  
  \[ w' = <p_1, \ldots, p_{i-1}, p_{i+1}, p_i, \ldots, p_n> \]

- If \( p_i \) is a matching pair and \( p_{i+1} \) is a non-matching pair, then \( C(w) \leq C(w') \)
  
  - \( C(w) \): # of crowdsourced pairs required for \( w \)

- That is, always better to first label a matching pair and then a non-matching pair

- In reality, optimal label order cannot be achieved
Join [Wang-SIGMOD13]

- **Expected optimal labeling order**
  - \( w = <p_1, p_2, ..., p_n> \)
  - \( C(w) = \# \text{ of crowdsourced pairs required for } w \)

\[
E[C(\omega)] = \sum_{i=1}^{n} \mathbb{P}(p_i = \text{crowdsourced})
\]

- \( P(p_i = \text{crowdsourced}) \)
  - Enumerate all possible labels of \( <p_1, p_2, ..., p_{i-1}> \), and for each possibility, derive whether \( p_i \) is crowdsourced or not
  - Sum of the probability of each possibility that whether \( p_i \) is crowdsourced
Join [Wang-SIGMOD13]

**Expected optimal labeling order**

- \( w_1 = \langle p_1, p_2, p_3 \rangle \)
- \( E[C(w_1)] = 1 + 1 + 0.05 = 2.05 \)
  - \( P_1: P(P_1 = \text{crowdsourced}) = 1 \)
  - \( P_2: P(P_2 = \text{crowdsourced}) = 1 \)
  - \( P_3: P(P_3 = \text{crowdsourced}) = P(\text{both } P_1 \text{ and } P_2 \text{ are non-matching}) = (1-0.9)(1-0.5) = 0.05 \)

<table>
<thead>
<tr>
<th>Probability of matching</th>
<th>Expected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1 ) 0.9</td>
<td></td>
</tr>
<tr>
<td>( P_2 ) 0.5</td>
<td></td>
</tr>
<tr>
<td>( P_3 ) 0.1</td>
<td></td>
</tr>
</tbody>
</table>

Expected values:

- \( w_1 = \langle p_1, p_2, p_3 \rangle \) 2.05
- \( w_2 = \langle p_1, p_3, p_2 \rangle \) 2.09
- \( w_3 = \langle p_2, p_3, p_1 \rangle \) 2.45
- \( w_4 = \langle p_2, p_1, p_3 \rangle \) 2.05
- ...
Join [Wang-SIGMOD13]

- **Expected optimal labeling order**
  - Label the pairs in **the decreasing order of the probability** that they are a matching pair
  - Eg, $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8$

$$E[C(\omega)] = \sum_{i=1}^{n} \mathbb{P}(p_i = \text{crowdsourced})$$

<table>
<thead>
<tr>
<th>ID</th>
<th>Object Pairs</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>$(o_2, o_3)$</td>
<td>0.85</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$(o_1, o_2)$</td>
<td>0.75</td>
</tr>
<tr>
<td>$p_3$</td>
<td>$(o_1, o_6)$</td>
<td>0.72</td>
</tr>
<tr>
<td>$p_4$</td>
<td>$(o_1, o_3)$</td>
<td>0.65</td>
</tr>
<tr>
<td>$p_5$</td>
<td>$(o_4, o_5)$</td>
<td>0.55</td>
</tr>
<tr>
<td>$p_6$</td>
<td>$(o_4, o_6)$</td>
<td>0.48</td>
</tr>
<tr>
<td>$p_7$</td>
<td>$(o_2, o_4)$</td>
<td>0.45</td>
</tr>
<tr>
<td>$p_8$</td>
<td>$(o_5, o_6)$</td>
<td>0.42</td>
</tr>
</tbody>
</table>
Parallel labeling

- To reduce the rounds $\rightarrow$ smaller latency
- Eg, $w=<(o_1, o_2), (o_2, o_3), (o_3, o_4)>$
  - $(o_1, o_2)$ has to be crowdsourced
  - $(o_2, o_3)$ has to be crowdsourced whether the label of $(o_1, o_2)$ is matching or not
  - $(o_3, o_4)$ has to be crowdsourced no matter which labels $(o_1, o_2)$ or $(o_2, o_3)$ has
  - $\Rightarrow$ All three pairs can be crowdsourced concurrently
Join [Wang-SIGMOD13]

- **Parallel** labeling
  - \( w = \langle p_1, p_2, \ldots, p_h, \ldots, p_{i-1}, p_i, \ldots, p_n \rangle \)
  - \( p_i \) needs to be crowdsourced iff:
    - \( p_i \) cannot be deduced from \( \langle p_1, p_2, \ldots, p_{i-1} \rangle \)
    - \( \langle p_1, p_2, \ldots, p_{i-1} \rangle \) has more than one non-matching
  - If an intermediate pair \( p_h \) is unlabeled
    - Assume \( p_h \) as matching and check the transitivity
  - For each pair \( p_i \) (1 ≤ i ≤ n)
    - Output \( p_i \) as a crowdsourced pair if \( p_i \) cannot be deduced from \( \langle p_1, p_2, \ldots, p_{i-1} \rangle \) concurrently
    - Based on results, deduce pairs via transitivity
    - Iterate until all pairs are labeled
Join [Wang-SIGMOD13]

- Parallel labeling
- Iterative algorithm

The first iteration

(a) Crowdsourcing \{p_1, p_2, p_3, p_5, p_6\}

(b) Deducing \{p_4, p_8\}

The second iteration

(c) Crowdsourcing \{p_7\}

(d) Terminate
Join [Wang-SIGMOD13]

- Two data sets
  - Paper: 997 (author, title, venue, date, and pages)
  - Product: 1081 product (abt.com), 1092 product (buy.com)
Join [Wang-SIGMOD13]

- Transitivity

(a) Paper

(b) Product
Join [Wang-SIGMOD13]

- Labeling order

(a) Paper

(b) Product
Conclusion

- Sampled a few representative human-powered DB operations

- Exciting field with lots of opportunities
Reference

- [Cao-VLDB12] Whom to Ask? Jury Selection for Decision Making Tasks on Microblog Services, Caleb Chen Cao et al., VLDB 2012
- [Chaudhuri-ICDE06] A Primitive Operator for Similarity Join in Data Cleaning, Surajit Chaudhuri et al., ICDE 2006
- [Davidson-ICDT13] Using the crowd for top-k and group-by queries, Susan Davidson et al., ICDT 2013
- [Dwork-WWW01] Rank Aggregation Methods for the Web, Cynthia Dwork et al., WWW 2001
- [Franklin-SIGMOD11] CrowdDB: answering queries with crowdsourcing, Michael J. Franklin et al, SIGMOD 2011
- [Franklin-ICDE13] Crowdsourced enumeration queries, Michael J. Franklin et al, ICDE 2013
- [Guo-SIGMOD12] So who won?: dynamic max discovery with the crowd, Stephen Guo et al., SIGMOD 2012
- [Howe-08] Crowdsourcing, Jeff Howe, 2008
Reference

- [Li-HotDB12] *Crowdsourcing: Challenges and Opportunities*, Guoliang Li, HotDB 2012
- [Liu-VLDB12] *CDAS: A Crowdsourcing Data Analytics System*, Xuan Liu et al., VLDB 2012
- [Marcus-VLDB12] *Counting with the Crowd*, Adam Marcus et al., VLDB 2012
- [Sarma-ICDE14] Crowd-Powered Find Algorithms, Anish Das Sarma et al., ICDE 2014
- [Shirky-08] *Here Comes Everybody*, Clay Shirky, 2008
Reference

- [Surowiecki-04] *The Wisdom of Crowds*, James Surowiecki, 2004
- [Whang-VLDB13] *Question Selection for Crowd Entity Resolution*, Steven Whang et al., VLDB 2013
- [Yan-MobiSys10] *CrowdSearch: exploiting crowds for accurate real-time image search on mobile phones*, T. Yan et al., MobiSys 2010