Human-Powered Database Operations: Part 1

Dongwon Lee
Penn State University, USA
dongwon@psu.edu

Slide available @ http://goo.gl/4pNUhB

SBBD 2014 Tutorial
Where Am I From?
Penn State University

- State College, PA
  - Out of nowhere, but close to everywhere

- West: 2.5 hours to Pittsburgh
- East: 4 hours to New York
- South: 3 hours to Washington DC
- North: 3 hours to Niagara Falls
Penn State i-School

- College of Information Sciences and Technology (IST)
  - http://ist.psu.edu/
- 40+ tenure-track faculty on diverse areas
  - CompSci & EE
  - MIS & LIS
  - Design
  - Law
  - Psychology
  - Medical Infomatics
### Other Tutorials on Crowdsourcing

<table>
<thead>
<tr>
<th>Year</th>
<th>Sub-field</th>
<th>Venue</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>Crowdsourcing</td>
<td>SDM</td>
<td>Crowdsourcing &amp; Human Computation</td>
</tr>
<tr>
<td>2013</td>
<td>Crowdsourcing</td>
<td>HCOMP</td>
<td>Incentives in Human Computation</td>
</tr>
<tr>
<td>2012</td>
<td>HCI</td>
<td>AAAI</td>
<td>Crowdsourcing using MTurk for HCI Research</td>
</tr>
<tr>
<td>2012</td>
<td>Crowdsourcing</td>
<td>SBP</td>
<td>Crowdsourcing, Human Computation, and Collective Intelligence</td>
</tr>
<tr>
<td>2012</td>
<td>IR</td>
<td>SIGIR</td>
<td>Human Computation and Crowdsourcing</td>
</tr>
<tr>
<td>2012</td>
<td>DB</td>
<td>SIGMOD</td>
<td>Designing a Scalable Crowdsourcing Platform</td>
</tr>
<tr>
<td>2011</td>
<td>Crowdsourcing</td>
<td>AAAI</td>
<td>Human Computation: Core Research Questions and State of the Art</td>
</tr>
<tr>
<td>2011</td>
<td>IR</td>
<td>CLEF</td>
<td>Crowdsourcing for IR Experimentation and Evaluation</td>
</tr>
<tr>
<td>2011</td>
<td>ML</td>
<td>ICML</td>
<td>Collective Intelligence and Machine Learning</td>
</tr>
<tr>
<td>2011</td>
<td>Social Science</td>
<td>EC</td>
<td>Conducting Behavioral Research using AMT</td>
</tr>
<tr>
<td>2011</td>
<td>Multimedia</td>
<td>MM</td>
<td>Frontiers in Multimedia Search</td>
</tr>
<tr>
<td>2011</td>
<td>DB</td>
<td>VLDB</td>
<td>Crowdsourcing Application and Platforms</td>
</tr>
<tr>
<td>2011</td>
<td>Crowdsourcing</td>
<td>WWW</td>
<td>Managing Crowdsourced Human Computation</td>
</tr>
<tr>
<td>2010</td>
<td>Vision</td>
<td>CVPR</td>
<td>Mechanical Turk for Computer Vision</td>
</tr>
<tr>
<td>2008</td>
<td>IR</td>
<td>CIKM</td>
<td>Crowdsourcing for Relevance Evaluation</td>
</tr>
</tbody>
</table>
The Focus of This Tutorial

- Part 1 on **basics** of crowdsourcing
- Part 2 on **DB operations** that exploit crowdsourcing

http://istc-bigdata.org/index.php/crowdsourcing-big-data/
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- Computational Crowdsourcing
  - Preliminaries
  - Transcription
  - Sorting
- Demo
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
Loosely coupled teams can create software polytechniques with interfaces, decoupling data from analysis to enable users at a distance.

The U.S. Coast Guard developed a system to aid search and rescue operations. An interesting potential research area for computer scientists.

New open-source tools and techniques could help with group computing and crowdsourced image acquisition, map volume image processing, and the modeling of ocean drift modeling, and analysis of open-water satellite imagery.
Eg, Threadless.com

- Sells t-shirts, designed/voted by crowds
- Artists whose designs are chosen get paid
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
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
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, …
Mobile Millennium is a research project that includes a pilot traffic-monitoring system that uses the GPS in cellular phones to gather traffic information, process it, and distribute it back to the phones in real time.
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, Who Wants to be a Millionaire?

Millionaire
Audience
BIGGEST
FAIL
EVER!!!!!
Eg, Game-With-A-Purpose: GWAP

- Term coined by Luis von Ahn @ CMU

- Eg,
  - ESP Game → Google Image Labeler: image recognition
  - Foldit: protein folding
  - Duolingo: language translation
Crowdsourcing landscape

For definitions, analysis, free book chapters, and other crowdsourcing resources go to:

www.resultsfromcrowds.com

http://www.resultsfromcrowds.com/features/crowdsourcing-landscape/
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- Computational Crowdsourcing
  - Preliminaries
  - Transcription
  - Sorting
- Demo
"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
“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
“Crowdsourcing as an online, distributed problem-solving and production model that leverages the collective intelligence of online communities to serve specific organizational goals… top-down and bottom-up…”
What is Crowdsourcing?

- Many definitions

- A few characteristics
  - *Outsourced* to *human* workers
  - *Online* and *distributed*
  - Open call & right *incentive*
  - *Diversity* and *independence*
  - Top-down & bottom-up
What is **Computational Crowdsourcing**?

- Focus on computational aspect of crowdsourcing
  - Algorithmic aspect
  - Non-linear optimization problem

- Mainly use micro-tasks

- When to use Computational Crowdsourcing?
  1. Machine cannot do the task well
  2. Large crowds can probably do it well
  3. Task can be split to many micro-tasks
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- Computational Crowdsourcing
  - Preliminaries
  - Transcription
  - Sorting
- Demo
Three Parties

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

- **Marketplaces**
  - Provide crowds with tasks

- **Crowds**
  - Workers perform tasks
Notable Marketplaces

- Mechanical Turk
- CrowdFlower
- CloudCrowd
- Clickworker
- SamaSource
AMT: mturk.com

Workers

Requesters
AMT: Workers vs. Requesters

- **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: HIT

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

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

Translation task
Micro- vs. Macro-task: Eg, oDesk

Workers

Rodrigo A.
Game Programmer (Desktop/Mobile/Web)
I'm a Computer Scientist who is always studying hard developing softwares and games with a big passion. I love apps using Java for the last 5 years and since ...
### AMT: HIT List

![HIT List Image]

#### Workers qualification

- 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
**AMT: HIT Example**

**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 a phone number.
- Please be sure to click the appropriate option if the site has a phone number.

---

**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.]
AMT: HIT Example

Store name, date, time, total, location on this receipt
Requester: Vishwanath Kumar
Qualifications Required: Total approved HITs is greater than 1000

Walmart Supercentre

Thank you for choosing your Langley Wal-Mart
20202 66 Ave
Langley, BC

ST# 3158 DP# 00003990 TE# 09 TR# 00647
GV CALENDAR 062891600247 $1.00 J
GV CALENDAR 062891500247 $1.00 J
KTX U TMP 18 00360001595 3.97 J
CHIPSLVNGA 008411411904 2.47 J
ASPIRIN RS 005650038595 $4.76 J
VENDOR COUPON 005648911429 $6.97 J
VENDOR COUPON 00563184734 $6.97 J
VENDOR COUPON 005693690708 $2.00 H
CHEMO BITE

Subtotal $20.14
GST 5% 1.26
PST 7% 0.84
Total 22.24
Debit Tender Charge Due $0.00

GST/HST 137466199 RT 0001
QST 1016551356 TQ 0001
Open-Source Marketplace S/W

Build with PYBOSSA
The only open source framework for making crowdsourcing projects


volontariat / voluntary

Engine and Framework for open source crowdsourcing platforms like Volontari.at
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- **Computational Crowdsourcing**
  - Preliminaries
  - Transcription
  - Sorting
- Demo
Three Computational Factors

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

- **Monetary cost**
  - Cost per question
  - # of questions (i.e., HITs)
  - # of workers

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

**Diagram:**
- Latency
- Cost
- Quality

**Questions:**
- How long do we wait for?
- How much is the quality of answers satisfied?
- How much $$ does we spend?
#1: Latency

- Some crowdsourcing tasks finish faster than others
  - Eg, easier, or more rewarding tasks are popular
- Dependency among tasks
#2: Cost

- Cost per question
- # of HITs

Remaining cost to pay: 
$0.03 \times 2075 = $62.25
#3: Quality of Answers

- Avoid spam workers
- Use workers with reputation

- Ask the **same** question to multiple workers to get consensus (e.g., majority voting)
- Assign **more number of** (better-skilled) workers to **more difficult** questions
Size of Comparison

- Diverse forms of questions in a HIT
- Different sizes of comparisons in a question

Which is better?

Which is the best?

Accuracy

Cost

Latency

Binary question

N-ary question
Size of Batch

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

- # of human responses sought for a HIT

$r = 1$

Which is better?

$\begin{align*}
\text{W1} &\quad 0 \\
\text{W1} &\quad 0
\end{align*}$

$r = 3$

Which is better?

$\begin{align*}
\text{W1} &\quad 0 \\
\text{W2} &\quad 0 \\
\text{W3} &\quad 0
\end{align*}$

Cost, Latency

Smaller $r$

Accuracy

Larger $r$
Round (= Step)

- Algorithms are executed in rounds
- # of rounds $\approx$ latency

Which is better?

Parallel Execution

Round #1

Sequential Execution

Round #2

Which is better?
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- **Computational Crowdsourcing**
  - Preliminaries
  - **Transcription**
  - Sorting
- Demo
Eg, Text Transcription [Miller-13]

- **Problem**: one person cannot 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]

You (time) have good points, but they got lost amidst the writing. Overall your writing style is a bit too phoney. You do (work) not (not work) work (good time). I a few grammatical mistakes. Please improve the transcription of this handwriting. People will vote whether to approve your changes.

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

Please select the better transcription for this handwriting.
Differences are highlighted in yellow.

You (misspelled) (several) (words) (work). (?) (?) (?) work next (time). I also notice a few grammatical mistakes. Overall your writing style is a bit too (phoney). 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.

After 8 iterations with thousands of crowds
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- **Computational Crowdsourcing**
  - Preliminaries
  - Transcription
  - **Sorting**
- Demo
Human-Powered Sort

- Rank $N$ items using crowdsourcing with respect to the constraint $C$
- Often $C$ is subjective, fuzzy, ambiguous, and/or difficult-for-machines-to-compute
- Eg,
  - Which image is the most “representative” one of Brazil?
  - Which animal is the most “dangerous”?
  - Which actress is the most “beautiful”?
Human-Powered Sort

```
SELECT * FROM SoccerPlayers AS P
WHERE P.WorldCupYear = '2014'
ORDER BY CrowdOp('most-valuable')
```
Naïve Sort

- Eg, “Which of two players is better?”
- Naïve all pair-wise comparisons takes $\binom{N}{2}$ comparisons
  - Optimal # of comparison is $O(N \log N)$
Naïve Sort

- Conflicting opinions may occur
  - Cycle: A > B, B > C, and C > A

- If no cycle occurs
  - Naïve all pair-wise comparisons takes \( \binom{N}{2} \) comparisons

- If cycle exists
  - More comparisons are required
Sort [Marcus-VLDB11]

- N=5, S=3

A

B

C

D

E

W1

W2

W3

W4

A

B

C

D

E

1 1

1 2

1

1

1 1

1

1 1

1

1

1

1

1 1

1
Sort [Marcus-VLDB11]

- N=5, S=3

**Sorted Result**

A → B → C → E → D

Topological Sort

DAG
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- Computational Crowdsourcing
  - Preliminaries
  - Transcription
  - Sorting
- Demo
Demo: Human-Powered Sorting

- From your smartphone or laptop, access the following URL or QR code:

http://goo.gl/3tw7b5
Part 1 Conclusion

- **Crowdsourcing ≈ Human Computation**
- **Academia**: novel paradigm to solve the challenging problems in Computer Science
- **Industry**: novel entrepreneurial opportunities
  - Eg, Brazil-version Mechanical Turk?

This slide is available at

http://goo.gl/4pNUhB
Reference

- [Franklin-SIGMOD11] CrowdDB: answering queries with crowdsourcing, Michael J. Franklin et al, SIGMOD 2011
- [Howe-08] Crowdsourcing, Jeff Howe, 2008
- [Li-HotDB12] Crowdsourcing: Challenges and Opportunities, Guoliang Li, HotDB 2012
- [Marcus-VLDB11] Human-powered Sorts and Joins, Adam Marcus et al., VLDB 2011
- [Shirky-08] Here Comes Everybody, Clay Shirky, 2008
Human-Powered Database Operations: Part 2

Dongwon Lee
Penn State University, USA
dongwon@psu.edu

Slide available @ http://goo.gl/UEUEBh

SBBD 2014 Tutorial
Part 1: Crowdsourcing Basics

- Examples
- Definitions
- Marketplaces
- Computational Crowdsourcing
  - Preliminaries
  - Transcription
  - Sorting
- Demo
Part 2: Crowdsourced Algo. in DB

- Preliminaries
- Sort
- Select
- Count
- Top-1
- Top-\(k\)
- Join
New Challenges

- Open-world assumption (OWA)
- Non-deterministic algorithmic behavior
- Trade-off among cost, latency, and accuracy
Crowdsourcing DB Projects

- CDAS @ NUS
- CrowdDB @ UC Berkeley & ETH Zurich
- MoDaS @ Tel Aviv U.
- Qurk @ MIT
- sCOOP @ Stanford & UCSC
Part 2: Crowdsourced Algo. in DB

- Preliminaries
- **Sort**
- Select
- Count
- Top-1
- Top-\(k\)
- Join
Sort Operation

- Rank $N$ items using crowdsourcing w.r.t some criteria
- 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
- If cycle exists
  - More comparisons are required
Sort [Marcus-VLDB11]

- Proposed 3 crowdsourced sort algorithms
- **#1: Comparison-based Sort**
  - Workers rank $S$ items ($S \subset N$) per HIT
  - Each HIT yields $\binom{S}{2}$ pair-wise comparisons
  - Build a directed graph using all pair-wise comparisons from all workers
    - If $i > j$, then add an edge from $i$ to $j$
  - Break a cycle in the graph: “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
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

A > B > C > D > E

Diagram:

- Node A connected to B, D, E with weights 1.
- Node B connected to C with weight 1 and to E with weight 2.
- Node C connected to D with weight 1.
- Node D connected to E with weight 1.

Edges:

- A to B: 1
- A to D: 1
- A to E: 1
- B to C: 1
- B to E: 2
- C to D: 1
- D to E: 1
Sort [Marcus-VLDB11]

- N=5, S=3

A - B - C - D - E

DAG

Topological Sort

Sorted Result

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: $O(N)$
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:

![Squares Diagram]
Sort [Marcus-VLDB11]

- #3: Hybrid Sort
  - First, do rating-based sort → sorted list $L$
  - Second, do comparison-based sort on $S \subseteq L$

- How to select the size of $S$
  - Random
  - Confidence-based
  - Sliding window
Sort [Marcus-VLDB11]

The chart illustrates the rank correlation between comparison and rating, as well as worker agreement. The bars represent the Kappa/Tau values for different quadrants (Q1 to Q5). The chart shows that Kappa values are consistently higher than Tau values, indicating better agreement in the comparison vs. rating metric.
Sort [Marcus-VLDB11]
Part II: Crowdsourced Algo. in DB

- Preliminaries
- Sort
- **Select**
- Count
- Top-1
- Top-\( k \)
- Join
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 search an image relevant to a query
  - [Parameswaran-SIGMOD12] develops human-powered filtering algorithms
  - [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 [Yan-MobiSys10]

- Improving mobile image search using crowdsourcing
Select [Yan-MobiSys10]

- Ensuring accuracy with majority voting
- Given accuracy, optimize cost and latency
- **Deadline** as latency in mobile phones
Select [Yan-MobiSys10]

- Goal: For a query image $Q$, find the first relevant image with \textbf{min cost} before the \textbf{deadline}
Select [Yan-MobiSys10]

- Parallel crowdsourced validation
Select [Yan-MobiSys10]

- Sequential crowdsourced validation
Select [Yan-MobiSys10]

- CrowdSearch: using early prediction on the delay and outcome to start the validation of next candidate early
Select [Yan-MobiSys10]
Select [Parameswaran-SIGMOD12]

- Novel grid-based visualization

Same person?

Yes  ○  No  ○
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 $|\#\text{YES} - \#\text{NO}| > X$, or at most $Y$ questions → **Chopped off triangle**
Select [Parameswaran-SIGMOD12]

- What is the best strategy? Find strategy with minimum overall expected cost s.t.
  1. **Overall expected error** is less than threshold
  2. **# of questions** per item never exceeds $m$
Part 2: Crowdsourced Algo. in DB

- Preliminaries
- Sort
- Select
- Count
- Top-1
- Top-\(k\)
- Join
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
Q: “How many teens are participating in the Hong Kong demonstration?”
Count Operation

- Using Face++, guess the age of a person

10 - 56
20 - 30
15 - 29

http://www.faceplusplus.com/demo-detect/
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 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
Part 2: Crowdsourced Algo. in DB

- Preliminaries
- Sort
- Select
- Count
- Top-1
- Top-\(k\)
- Join
Top-1 Operation

- Find the top-1, either MAX or MIN, among $N$ items w.r.t. some criteria

- 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 = $r_1 + r_2 + r_3 = 11$

\[
\begin{align*}
  s_1 &= 2 \\
  r_1 &= 3 \\
  s_2 &= 3 \\
  r_2 &= 3 \\
  s_3 &= 2 \\
  r_3 &= 5
\end{align*}
\]
Max [Venetis-WWW12]

- Bubble Max Case #2

- \( N = 5 \)
- \( \text{Rounds} = 2 \)
- \( \# \text{ of questions} = r_1 + r_2 = 8 \)

\( s_1 = 4 \)
\( r_1 = 3 \)

\( s_2 = 2 \)
\( r_2 = 5 \)
Max [Venetis-WWW12]

- **Tournament Max**
  - $s_1 = 2$
  - $r_1 = 1$
  - $s_2 = 2$
  - $r_2 = 1$
  - $s_3 = 2$
  - $r_3 = 3$
  - $s_4 = 2$
  - $r_4 = 5$

- $N = 5$
- Rounds = 3
- # of questions = $r_1 + r_2 + r_3 + r_4 = 10$
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]
Part 2: Crowdsourced Algo. in DB

- Preliminaries
- Sort
- Select
- Count
- Top-1
- Top-\(k\)
- Join
Top-\(k\) Operation

- Find top-\(k\) items among \(N\) items w.r.t. some criteria

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

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

- Examples
  - \textbf{[Davidson-ICDT13]} investigates the variable user error model in solving top-\textit{k} list problem
  - \textbf{[Polychronopoulous-WebDB13]} proposes tournament-based top-\textit{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

Round 1

Round 2

Round 3

Total, 4 questions with 3 rounds
Top-\textit{k}: Tournament Solution ($k = 2$)

- Phase 2: \textbf{Updating a tournament tree}
- \textbf{Iteratively} asking pair-wise questions from the bottom level

Round 1

Round 2

Round 3
Top-$k$: Tournament Solution ($k = 2$)

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

Round 4

Round 5

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 based top-$k$ scheme 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
Top-\(k\) [Polychronopoulous-WebDB13]

- **Algorithm**
  - **Input:** \(N\) items, integer \(k\) and \(s\) (ie, \(s > k\))
  - **Output:** top-\(k\) set
  - **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 \(s\): \(s\)-rank(\(s\))
      - Merge all top-\(k\) items into \(O\)
    - **Return** \(O\)

- **More effective when** \(s\) and \(k\) are **small**
  - Eg, \(s\)-rank(20) with \(k=10\) may give poor accuracy
Top-\(k\) [Polychronopoulous-WebDB13]

- Eg, \(N=10, s=4, k=2\)

Top-2 items

\[\text{s-rank()}\]
Top-\(k\) [Polychronopoulous-WebDB13]

- \(s\)-rank(\(s\))
  
  // workers rank \(s\) 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(): $s=4$, $k=2$, $w=3$

```
| W1  | 4  | 1  | 2  | 3  |
| W2  | 4  | 2  | 1  | 3  |
| W3  | 3  | 2  | 3  | 4  |
```

Median Ranks

```
|            | 4  | 2  | 2  | 3  |
| Top-2      |    |    |    |    |
```
Top-\(k\) [Polychronopoulos-WebDB13]

- Comparison to Sort [Marcus-VLDB11]
Top-\(k\) [Polychronopoulos-WebDB13]

- Comparison to Max [Venetis-WWW12]
Part 2: Crowdsourced Algo. in DB

- Preliminaries
- Sort
- Select
- Count
- Top-1
- Top-$k$
- Join
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
  - $R \text{ JOIN}_p S$, where $p=sim(R.A, S.A) > t$
  - $sim()$ 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
Join [Marcus-VLDB11]

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

#2 Naïve Batching Join

Batch factor $b = 2$
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 again.
- If none of the celebrities match, check the I did not find any pairs option.
- There may be multiple matches per page.

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

Submit

#3 Smart Batching Join
Join [Marcus-VLDB11]

MV: Majority Voting
QA: Quality Adjustment
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, \( (10,000 \times 10,000) / (20 \times 20) = 250,000 \) HITs → Still too many!

- [Wang-VLDB12] contributes CrowdER:
  1. A hybrid human-machine join
     - #1 machine-join prunes obvious non-matches
     - #2 human-join 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>
<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]
Join [Wang-VLDB12]

- HIT Generation Problem
  - Input: pairs of records $P$, # of records in HIT $k$
  - Output: minimum # of HITs s.t.
    1. All HITs have at most $k$ records
    2. 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 conn with max indegree (# of edges to scc) and min outdegree (# of edges to non-scc) if tie
    - move $r_{\text{new}}$ from conn to scc
    - update conn using new scc
  - add scc into SCC
Join [Wang-VLDB12]

(a) Initialize scc = \{r_4\}

(b) conn = \{r_3, r_5, r_6, r_7\}
   Add r_6 into scc

(c) conn = \{r_3, r_5, r_7\}
   Add r_5 into scc

(d) conn = \{r_3, r_7\}
   Add r_3 into scc

(e) Output scc

(f) Output other scc
Join [Wang-VLDB12]

(a) Initialize scc=\{r_4\}
(b) conn = \{r_3, r_5, r_6, r_7\}
    Add r_6 into scc
(c) conn = \{r_3, r_5, r_7\}
    Add r_5 into scc
(d) conn = \{r_3, r_7\}
    Add r_3 into scc
(e) Output scc
(f) Output other scc
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

http://blogs.oc.edu/ece/transitivity/
Join [Wang-SIGMOD13]

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

  $iPad\ 2^{nd}\ Gen = iPad\ Two$
  
  $iPad\ Two = iPad\ 2$

  $iPad\ 2^{nd}\ Gen = iPad\ Two$

- Negative transitive relation
  - If a = b, b ≠ c, then a ≠ c

  $iPad\ 2^{nd}\ Gen = iPad\ Two$
  
  $iPad\ Two ≠ iPad\ 3$

  $iPad\ 2^{nd}\ Gen ≠ iPad\ 3$
Join [Wang-SIGMOD13]

- 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) → match
(o₅, o₇) → non-match
(o₁, o₇) → ?
Join [Wang-SIGMOD13]

- Given a pair \((o_i, o_j)\), to check the transitivity
  - Enumerate path from \(o_i\) to \(o_j\) \(\rightarrow\) 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 1: ID vs. Object

<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 2: ID vs. Object Pairs & Likelihood

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

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

- Theorem: Optimal labeling order

  \[ w = \langle p_1, \ldots, p_{i-1}, p_i, p_{i+1}, \ldots, p_n \rangle \]
  \[ w' = \langle p_1, \ldots, p_{i-1}, p_{i+1}, p_i, \ldots, p_n \rangle \]

  - 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**
  - $w = \langle p_1, p_2, ..., p_n \rangle$
  - $C(w) = \# \text{ of crowdsourced pairs required for } w$
  
  $$E[C(w)] = \sum_{i=1}^{n} P(p_i = \text{crowdsourced})$$

- $P(p_i = \text{crowdsourced})$
  - Enumerate all possible labels of $\langle p_1, p_2, ..., p_{i-1} \rangle$, 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>
<tr>
<td>$w_1 = \langle p_1, p_2, p_3 \rangle$</td>
<td>2.05</td>
</tr>
<tr>
<td>$w_2 = \langle p_1, p_3, p_2 \rangle$</td>
<td>2.09</td>
</tr>
<tr>
<td>$w_3 = \langle p_2, p_3, p_1 \rangle$</td>
<td>2.45</td>
</tr>
<tr>
<td>$w_4 = \langle p_2, p_1, p_3 \rangle$</td>
<td>2.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Theorem:** Expected optimal labeling order

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

![Graph representation](image)

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

<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>
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
Machine vs. Human

- Human-Powered Crowdsourcing $\rightarrow$ “Human-in-the-loop” Crowdsourcing
  - Should use machine to process majority of big data
  - Should use human to process a small fraction of challenging cases in big data
- How to **split** tasks and **combine** results for machines and human **automatically** is an open issue

http://www.theoddblog.us/2014/02/21/damienwaltershumanloop/
Conclusion

- New opportunities
  - Open-world assumption
  - Non-deterministic algorithmic behavior
  - Trade-off among cost, latency, and accuracy
- Crowdsourcing for Big Data?

This slide is available at

http://goo.gl/UEUEBh
Reference

- [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] Crowdpowered 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