Crowdsourced Top-k Algorithms: An Experimental Evaluation

Xiaohang Zhang Guoliang Li Jianhua Feng
Department of Computer Science, Tsinghua National Laboratory for Information Science and Technology (TNList),Tsinghua University, Beijing, China
zhangxiaohang12@mails.tsinghua.edu.cn; {liguoliang, fengjh}@tsinghua.edu.cn

ABSTRACT

Crowdsourced top-k computation has attracted significant attention recently, thanks to emerging crowdsourcing platforms, e.g., Amazon Mechanical Turk and CrowdFlower. Crowdsourced top-k algorithms ask the crowd to compare the objects and infer the top-k objects based on the crowdsourced comparison results. The crowd may return incorrect answers, but traditional top-k algorithms cannot tolerate the errors from the crowd. To address this problem, the database and machine-learning communities have independently studied the crowdsourced top-k problem. The database community proposes the heuristic-based solutions while the machine-learning community proposes the learning-based methods (e.g., maximum likelihood estimation). However, these two types of techniques have not been compared systematically under the same experimental framework. Thus it is rather difficult for a practitioner to decide which algorithm should be adopted. Furthermore, the experimental evaluation of existing studies has several weaknesses. Some methods assume the crowd returns high-quality results and some algorithms are only tested on simulated experiments. To alleviate these limitations, in this paper we present a comprehensive comparison of crowdsourced top-k algorithms. Using various synthetic and real datasets, we evaluate each algorithm in terms of result quality and efficiency on real crowdsourcing platforms. We reveal the characteristics of different techniques and provide guidelines on selecting appropriate algorithms for various scenarios.

1. INTRODUCTION

Given a set of objects, the top-k problem aims to find the top-k best objects from the set. Due to its widespread applications, top-k algorithms have been widely adopted in many real-world systems, e.g., search engines and advertisement systems. If the objects can be compared by machines, e.g., numerical values, existing deterministic top-k algorithms can efficiently compute top-k objects, e.g., the heap-based algorithms [5]. However in many real-world applications, the objects are rather hard to compare for machines. For example, in image search, comparing the picture clarity is hard for machines. As another example, in natural language processing, comparing the understanding difficulty between different sentences is also rather hard for machines. Obviously these comparisons are easy for human. Thus, crowdsourced top-k computation has been widely studied, thanks to emerging crowdsourcing platforms, e.g., Amazon Mechanical Turk and CrowdFlower. Crowdsourced top-k algorithms ask the crowd to compare objects and infer the top-k objects based on the crowdsourced comparison results.

As the crowd may return incorrect results, a natural problem is to obtain high-quality top-k answers by aggregating the inaccurate comparison results from the crowd. The database community and machine-learning community have independently studied the crowdsourced top-k problem. The database community takes top-k as the basic operation in databases and proposes heuristic-based solutions [9,18] while the machine-learning community focuses on ranking documents/images and proposes learning-based methods (e.g., maximal likelihood estimation [4,16], matrix decomposition [11]). However, these two categories of techniques have not been compared systematically under the same experimental framework. Thus it is difficult for a practitioner to decide which method should be adopted. Furthermore, the experimental evaluation of existing studies has weaknesses. Some methods assume the crowd returns high-quality results while some algorithms are only tested on simulated experiments.

To address these problems, this paper presents a comprehensive comparison on crowdsourced top-k algorithms. Most of existing studies utilize a comparison-based method and employ a two-step strategy. The first step (called pair selection) selects b object pairs and crowdsources them. And the second step (called result inference) infers the top-k pairs based on the comparison results of the crowdsourced pairs. We make a comprehensive survey on both pair selection and top-k inference algorithms. We also discuss other crowdsourced top-k algorithms. We evaluate all of the algorithms on real crowdsourcing platforms.

To summarize, we make the following contributions. (1) We provide a comprehensive survey on more than twenty crowdsourced top-k algorithms. (2) We compare existing crowdsourced top-k algorithms from different research communities through extensive experiments with a variety of synthetic and real datasets on real crowdsourcing platforms. (3) We report comprehensive findings obtained from the experiments. We provide new insights on the strengths and weaknesses of existing algorithms that can guide practitioners to select appropriate algorithms for various scenarios.

Proceedings of the VLDB Endowment, Vol. 9, No. 8
Copyright 2016 VLDB Endowment 2150-8097/16/04.

2. PRELIMINARIES

2.1 Problem Formulation

Definition 1 (Crowdsourced top-k Problem). *Given an object set $\mathcal{O} = \{o_1, o_2, \cdots, o_n\}$, where the objects are comparable but hard to compare by machines, find a k-size object set $\mathcal{R} = \{o_1, o_2, \cdots, o_k\}$ where o_i is preferred to o_j (denoted by $o_i \succ o_j$) for $o_i \in \mathcal{R}$ and $o_j \in \mathcal{O} - \mathcal{R}$.*

Any comparison operation ($\succ$) can be used to ask the crowd to compare objects. For example, given six images, we want to identify top-3 clearest ones. Obviously it is hard to use machines to decide which image is clearer and we need to utilize the crowd to compare the images. The goal of devising crowdsourced algorithms is to identify the top-k objects as accurate as possible and we will discuss two metrics to quantify crowdsourced algorithms in Section 6.1.

2.2 Workflow

To utilize the crowd to find top-k objects, we need to generate crowdsourced tasks. There are two widely-used ways to generate crowdsourced tasks. The first is pairwise comparison, which selects two objects and asks the crowd to choose the preferred one. The second is rating, which selects multiple objects and asks the crowd to assign a rate for each object. The rating-based method has some weaknesses. First, the crowd prefers pairwise comparisons to ratings as the former is much easier. Second, it is rather hard for the crowd to assign an accurate rate and the objects in different rating groups may not be correctly compared. Thus the rating-based method usually has a lower accuracy than pairwise comparison [12,14]. Due to these reasons, most of existing works use pairwise comparisons [13]. We first focus on pairwise comparisons, and then discuss hybrid methods that combine ratings and pairwise comparisons [12,20].

3. PAIRWISE COMPARISONS

Given a set with n objects, if all pairs are crowdsourced, there are $\binom{n}{2}$ pairs. Due to the high monetary cost for the crowd, it is not acceptable to enumerate all pairs for large object sets. An alternative is to ask $\mathcal{B}$ pairs, where $\mathcal{B}$ is a given budget. To achieve high recall, an iterative method is widely adopted, where b pairs are crowdsourced in each round. Based on the comparison results of the crowdsourced pairs, it decides how to select b pairs in next round. Iteratively, it crowdsources $\mathcal{B}$ pairs in total and uses the comparison results on these pairs to infer the top-k objects.

For each selected pair, a microtask is generated, which asks the crowd to compare the pair. To reduce crowd errors, each microtask is assigned to multiple workers and the final result is aggregated based on the results of these workers, e.g., weighted majority vote. Thus for each pair (o_i, o_j), two numbers are reported from the crowd: M_{ij}, M_{ji}, where M_{ij} (M_{ji}) is the number of workers who prefer o_i (o_j) to o_j (o_i). An aggregated weight w_{ij} is computed based on M_{ij} and M_{ji}, e.g., $w_{ij} = \frac{M_{ij}}{M_{ij}+M_{ji}}$. We can use a graph to model the comparison results of the crowdsourced pairs.

Definition 2 (Graph Model). *Given a set of crowdsourced pairs, a directed graph is constructed where nodes are objects and edges are comparison results. For each pair (o_i, o_j), if the aggregated result is $o_i \succ o_j$ (e.g., $w_{ij} > \frac{1}{2}$), there is a directed edge from o_i to o_j with weight w_{ij}.*

Table 1: Notations Used In The Paper.

Notation	Description
o_i	an object in set $\mathcal{O}$
n	the number of objects in set $\mathcal{O}$
k	the number of reported objects
b	the budget of selected pairs in each round
$\mathcal{L}$	collected pairwise comparison results from the crowd
$o_i \succ o_j$	o_i is preferred to o_j
M_{ij}	the number of workers reporting $o_i \succ o_j$
M	voting matrix collected from the crowd
p_{ij}	the probability of $o_i \succ o_j$
$\hat{s}$	estimation matrix of the latent scores for all objects
s_i	$s_i = \hat{s}[i]$, estimated score for object o_i
δ_i	variance of score for object o_i
η_w	the estimated accuracy for worker w

$$\begin{array}{c|cccccc} & o_1 & o_2 & o_3 & o_4 & o_5 & o_6 \\ \hline o_1 & - & 2 & 3 & 3 & - & - \\ o_2 & 1 & - & - & 3 & 3 & - \\ o_3 & 0 & - & - & - & 3 & - \\ o_4 & 0 & 0 & - & - & 3 & - \\ o_5 & - & 0 & 0 & - & - & 2 \\ o_6 & - & - & - & - & 1 & - \end{array}$$

Figure 1: Pairwise Result.

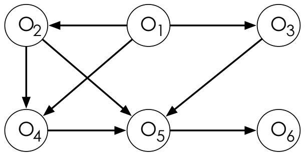


Figure 2: Graph Model.

Algorithm 1: Crowdsourced Top-k Algorithm

```
Input: Objects $\mathcal{O}$; Budget $\mathcal{B}$; Budget in each round $b$
Output: top-$k$ highest ranked objects $\mathcal{R}$
1 $\mathcal{U}$= All pairs in $\mathcal{O}$; // unanswered pairs
2 $\mathcal{E}$= $\phi$; // answered pairs
3 while $\mathcal{B} > 0$ do
4     $\mathcal{Q}$=PairSelection($\mathcal{O}$,$b$,$\mathcal{E}$,$\mathcal{U}$);
5     Publish questions $\mathcal{Q}$ to the crowd ;
6     Collect answers from the crowd ;
7     $\mathcal{E} = \mathcal{E} + \mathcal{Q}$; $\mathcal{U} = \mathcal{U} - \mathcal{Q}$; $\mathcal{B} = \mathcal{B} - b$ ;
8 $\mathcal{R}$=ResultInference($\mathcal{O}$,$\mathcal{E}$) ;
9 Return $\mathcal{R}$;
```

For example, consider the six objects in Figure 1, $o_1 \succ o_2 \succ o_3 \succ o_4 \succ o_5 \succ o_6$. Assume seven pairs are crowdsourced and each pair is answered by 3 workers (see Figure 1). $M_{12} = 2$ and $M_{21} = 1$ denote that two workers prefer o_1 to o_2 and one worker prefers o_2 to o_1. If $o_i \succ o_j$, we says o_i wins (o_j loses) or o_i is better than o_j (o_j is worse than o_i). Based on the crowdsourced answers, a graph is constructed in Figure 2. Table 1 summarizes the notations.

The pairwise-comparison-based method contains two steps. The first is `result inference`, which infers the top-k answers based on the crowdsourced pairs. The second is `pair selection`, which selects next b pairs based on the current result. Algorithm 1 shows the pseudo code. It first selects b pairs (line 4). Then it publishes these b pairs to the crowd and collects the results (lines 5-6). Finally it infers the top-k results (line 8). We will discuss the details of result inference and pair selection in Sections 4 and 5 respectively.

4. RESULT INFERENCE

In this section, we discuss how to infer the top-k results. We first discuss the heuristic-based algorithms in Section 4.1 and then introduce the machine-learning methods in Section 4.2. Next we discuss how to extend heap-based methods to infer the top-k results in Section 4.3. Lastly, we present the hybrid methods in Section 4.4.

4.1 Heuristic-Based Methods

Guo et al. prove that finding the maximal (top-1) object is NP-Hard by a reduction from Kemeny rankings [9]. Thus inferring the top-k objects is also NP-Hard and many heuristics are proposed to infer top-k objects.

4.1.1 Score-Based Algorithms

The score-based algorithms assign each object o_i with a score s_i and the k objects with the largest scores are selected as top-k answers. Next we discuss how to compute scores.

(1) Election Algorithms. The election algorithms that determine the winner of an election can be used to compute the score of an object. There are two famous election algorithms BordaCount [1] and Copeland [17].

BordaCount [1]. The score of o_i is its out-degree $d^+(o_i)$ (the number of wins compared with its out-neighbors), i.e.,

$$s_i = d^+(o_i). \tag{1}$$

In our example, s_1 is 3 as o_1 has 3 out-neighbors. The scores of $\langle o_1, o_2, o_3, o_4, o_5, o_6 \rangle$ are $\langle 3, 2, 1, 1, 1, 0 \rangle$.

Copeland [17]. The score of object o_i is its out-degree $d^+(o_i)$ minus its in-degree $d^-(o_i)$ (the number of wins minus the number of losses), i.e.,

$$s_i = d^+(o_i) - d^-(o_i). \tag{2}$$

In our example, s_2 is 1 as o_2 has 2 out-neighbors and 1 in-neighbor. The scores of $\langle o_1, o_2, o_3, o_4, o_5, o_6 \rangle$ are $\langle 3, 1, 0, -1, -2, -1 \rangle$. The top-2 answers are o_1 and o_2.

(2) Max Algorithms. Guo et al. [9] proposed several graph-based algorithms to compute the maximal object, which can be used to assign a score for each object.

Local [9]. The two-election methods only consider the neighbors of each object and cannot capture more information. Local algorithm [9] is proposed to solve the max problem by considering 2-hop neighbors and it can naturally be transformed to address the top-k problem. Obviously if an object has more 2-hop out-neighbors (i.e., its out-neighbors' out-neighbors), the object will beat more objects (based on transitivity), and thus the object has a larger score. Similarly, if an object has more 2-hop in-neighbors (i.e., its in-neighbors' in-neighbors), the object will be beaten by more objects, and thus the object has a lower score. Accordingly, the overall score of an object is computed as below.

$$s_i = M_{i*} - M_{*i} + \sum_{o_j \in \mathcal{D}^+(o_i)} M_{j*} - \sum_{o_j \in \mathcal{D}^-(o_i)} M_{*j}, \tag{3}$$

where $M_{i*} = \sum_j M_{ij}$, $M_{*i} = \sum_j M_{ji}$, and $\mathcal{D}^+(o_i)$ is the out-neighbor set of o_i and $\mathcal{D}^-(o_i)$ is the in-neighbor set of o_i. In our example, the score of o_2 is $(7-2)+(3+2)-1 = 9$ and the score of o_5 is $(2-10)+1-(2+3+6) = -18$. The scores of $\langle o_1, o_2, o_3, o_4, o_5, o_6 \rangle$ are $\langle 20, 9, 1, -4, -18, -11 \rangle$.

Indegree [9]. It computes the score based on the bayesian model. The probability of $o_i \succ o_j$ given M_{ij} and M_{ji} is

$$\begin{aligned} P(o_i \succ o_j | M_{ij}, M_{ji}) &= \frac{P(M_{ij}, M_{ji} | o_i \succ o_j) P(o_i \succ o_j)}{P(M_{ij}, M_{ji})} \\ &= \frac{P(M_{ij}, M_{ji} | o_i \succ o_j)}{P(M_{ij}, M_{ji} | o_i \succ o_j) + P(M_{ij}, M_{ji} | o_j \succ o_i)}, \end{aligned} \tag{4}$$

$$\begin{aligned} P(M_{ij}, M_{ji} | o_i \succ o_j) &= \binom{M_{ij} + M_{ji}}{M_{ij}} p^{M_{ij}} (1-p)^{M_{ji}}; \\ P(M_{ij}, M_{ji} | o_j \succ o_i) &= \binom{M_{ij} + M_{ji}}{M_{ji}} p^{M_{ji}} (1-p)^{M_{ij}}. \end{aligned} \tag{5}$$

p is the estimated worker accuracy which is a fixed value. Besides, it assumes $P(o_i \succ o_j) = P(o_i \prec o_j)$. Then it computes the score of object o_i as below.

$$s_i = \sum_{j \neq i} P(o_i \succ o_j | M_{ij}, M_{ji}). \tag{6}$$

Note that if o_i and o_j are not compared, $P(o_i \succ o_j | M_{ij}, M_{ji}) = 0.5$. In our example, if we set p to 0.55 [9], the scores for $\langle o_1, o_2, o_3, o_4, o_5, o_6 \rangle$ are $\langle 2.84, 2.74, 2.5, 2.35, 2.11, 2.45 \rangle$.

Top-k Algorithms		Details			
		Area	Input	Output	Parameters
Heuristic (Local)	BordaCount*	DB	Graph	Score	N/A
	Copeland*	DB	Graph	Score	N/A
	Local*	DB	Graph	Score	N/A
	Indegree*	DB	Graph	Score	N/A
	ELO*	ML	Comparison	Score	$H = 32$
	BRE*	ML	Comparison	Score	N/A
	URE*	ML	Comparison	Score	N/A
	SSCO	ML	Comparison	Score	$q = 0.1$
	SSSE	ML	Comparison	Score	$q = 0.1$
Heuristic (Global)	Iterative	DB	Graph	Subset	N/A
	PathRank	ML	Graph	Subset	N/A
	AdaptiveReduce	ML	Comparison	Subset	$p = 0.9$
	MPageRank*	DB	Graph	Score	$c = 1$
	RankCentrality*	ML	Graph	Score	N/A
	SSRW*	ML	Graph	Score	$q = 0.1$
ML	TrueSkill*	ML	Comparison	Score	$\varepsilon = 0$
	CrowdBT*	ML	Comparison	Score	$\lambda = 0.5$
	CrowdGauss*	ML	Comparison	Score	N/A
	HodgeRank*	ML	Comparison	Score	N/A
Extension	TwoStageHeap	DB	Comparison	Subset	$X = \frac{0.1n}{k^2}$
Rate + Compare	Combine*	ML	Rate+Compare	Score	N/A
	Hybrid*	DB	Rate+Compare	Score	$\delta = 0.1$
	HybridMPR*	DB	Rate+Graph	Score	$\beta = 0.6$

Table 2: Details for top-k Inference Algorithms (Methods with $*$ can also be used for sorting.)

Modified PageRank(MPageRank) [9]. It extends the original PageRank by considering the crowdsourced comparisons and computes the score of each object as below.

$$s_i = pr_k[i] = \frac{1-c}{n} + c \sum_{i \neq j} \frac{M_{ij}}{M_{*j}} pr_{k-1}[j], \tag{7}$$

where $pr_k[i]$ is the score of o_i in the *k-th* iteration (initially $pr_1[i] = \frac{1}{n}$) and c is a damping factor and set to 1 by default). The scores are $\langle 0.5, 0.5, 5e^{-11}, 5e^{-11}, 1.67e^{-11}, 1.67e^{-11} \rangle$.

(3) Ranking Algorithms. There are some ranking-based algorithms that focus on ranking documents/images.

RankCentrality [15]. The algorithm adopts an iterative method based on random walk. Its core component is to construct an $n \times n$ transition matrix P, in which

$$P_{ij} = \begin{cases} \frac{1}{d_{max}} w_{ji} & \text{if } i \neq j \\ 1 - \frac{1}{d_{max}} \sum_{k \neq i} w_{ki} & \text{if } i = j \end{cases}, \tag{8}$$

where w_{ji} is the weight of edge from o_j to o_i and d_{max} is the maximum in-degree of a node (i.e., $d_{max} = \max_{o_i} d^-(o_i)$). Then it initializes a $1 \times n$ matrix $\hat{s}$ and computes $\hat{s} \times P \times P \times \cdots$ until convergence. The score of o_i is computed as

$$s_i = \hat{s}[i] \text{ where } \hat{s} = \lim_{t \to \infty} \hat{s} \times P^t. \tag{9}$$

In our example, the scores for $\langle o_1, o_2, o_3, o_4, o_5, o_6 \rangle$ are $\langle 2.26, 1.68, 0.93, 0.63, 0.38, 0.91 \rangle$.

Balanced Rank Estimation(BRE)/Unbalanced Rank Estimation(URE) [19]. The score is computed based on the probability theory. The balanced rank estimation (BRE) considers both incoming and outgoing edges. To compute the score s_j, it computes the relative difference of the number of objects proceeding and succeeding o_j:

$$s_j = \frac{\sum_{i \neq j} b_{ij}(2w_{ij} - 1)}{2\alpha n} \propto \sum_{i \neq j} b_{ij}(2w_{ij} - 1). \tag{10}$$

where α is the selection rate (i.e., $\alpha\binom{n}{2}$ pairs will be compared); $w_{ij} = 1$ if $o_i \succ o_j$; $w_{ij} = 0$ otherwise; $b_{ij} = 1$ if o_i and o_j are compared by the crowd; $b_{ij} = 0$ otherwise.

The unbalanced rank estimation (URE) computes the score of o_i only based on its incoming edges. To compute the score s_j, it computes the number of objects proceeding o_j:

$$s_j = \frac{1}{\alpha n}\sum_{i \neq j} b_{ij} w_{ij}^n \propto \sum_{i \neq j} b_{ij} w_{ij}^n, \tag{11}$$

In our example, if we set the value of α to 1, then the scores are $\langle -0.25, -0.08, 0, 0.08, 0.17, 0.08\rangle$ for BRE. And the scores are $\langle 0, 0.17, 0.17, 0.33, 0.5, 0.17\rangle$ for URE.

ELO [6]. The ELO method is a chess ranking system, which can be applied to address the pairwise top-k problem. The basic idea is that, if object o_i with higher ranking beats another lower one o_j, only a few points will be added to s_i; on the contrary, if o_j wins, a lot of points will be added to s_j. Formally, each object o_i is randomly assigned a score s_i initially. When o_i is compared with o_j, the two scores s_i and s_j will be updated as below.

$$\begin{aligned} s_i &= s_i + H(C_i - \frac{1}{1 + 10^{(s_j - s_i)/400}}); \\ s_j &= s_j + H(1 - C_i - \frac{1}{1 + 10^{(s_i - s_j)/400}}), \end{aligned} \tag{12}$$

where H is a tuning parameter (set to 32 by default). $C_i = 1$ if $o_i \succ o_j$; $C_i = 0$ otherwise. The scores are $\langle 97.3, 65.3, 50.7, 36.2, 23.6, 32.8\rangle$ if the initial score is 50.

4.1.2 Iterative Reduce Top-k Algorithms

They iteratively eliminate lowly ranked objects that have small possibilities in the top-k results, until k objects left.

Iterative [9]. Guo et al. [9] improve the max algorithm and propose the iterative algorithm to improve the quality. It first utilizes the score-based methods to compute the scores of each object and then removes the half of the objects with small scores. Next it re-computes the scores on the survived objects and repeats the iterations until k objects left.

Eriksson et al. [7] propose two iterative reduce algorithms which are devised to rank documents and images.

PathRank [7]. The main idea of PathRank is to perform a "reverse" depth first search (DFS) for each node, which traverses the graph by visiting the in-neighbors of each node. If it finds a path with length larger than k, it eliminates the object as k objects have already been better than the object. However this method is not robust [7], because if not all the comparisons are observed, due to missing edges, objects ranked far from the top-k answers could potentially have no more than k-paths observed, and non-top-k objects can also be erroneously returned as an estimated top-k result.

AdaptiveReduce [7]. Initially there are n objects. It first randomly selects $(16(p - \frac{1}{2})^{-2} + 32)\log n$ objects as the chosen set X_c, where p is the worker accuracy. Then it wants to select an informative set and utilizes the set to eliminate objects with small possibilities in the top-k answers. To this end, it computes the voting count v_i for each object o_i in the chosen set, i.e., $v_i = \sum_{j=1}^{n} M_{ij}$. Next, it computes a subset of X_c as the informative set X_v, with voting count between $\frac{n}{4}$ and $\frac{3n}{4}$, i.e., $X_v = \{x \in X_c : \frac{n}{4} \leq v_x \leq \frac{3n}{4}\}$.

Based on X_v, it computes the wining count t_i of each object o_i, i.e., $t_i = \sum_{x \in X_v} M_{ix}$. Then objects with wining counts larger than $\frac{|X_v|}{2}$ will be kept, because they win many objects in the informative set. Then, it repeats the above steps using the survived objects until finding top-k objects.

Fekete et al. [3] propose bound-based methods for ranking in recommendation systems and image search, which build a stochastic matrix Q where $Q_{ij} = \frac{M_{ij}}{M_{ij}+M_{ji}}$, and devise several algorithms to identify top-k results from the matrix. **Sampling Strategy with Copeland's Ranking(SSCO) [3]** selects the top-k rows with most entries above 0.5 as the top-k results. **Sampling Strategy with Sum of Expectations (SSSE) [3]** selects the top-k rows with the largest average value as the top-k results. **Sampling Strategy based on Random Walk (SSRW) [3]** first transfers the matrix Q to the stochastic matrix S where $S_{ij} = \frac{Q_{ij}}{\sum_l Q_{li}}$ and computes the principal eigenvectors (that belong to the eigenvalue 1). Then it identifies the top-k rows with the largest eigenvalues as the top-k answers.

4.2 Machine-Learning Methods

These methods assume that each object has a latent score which follows a certain distribution. Then they utilize machine-learning techniques to estimate the scores. Lastly, they select k objects with the largest scores as the top-k results.

4.2.1 Maximum Likelihood Estimation

CrowdBT with Bradley-Terry Model [2]. This method uses Bradley-Terry (BT) model to estimate the latent score [2]. In the BT model, the probability of $o_i \succ o_j$ is assumed as $\frac{e^{s_i}}{e^{s_i}+e^{s_j}}$. Then based on the crowdsourced comparisons, it computes the latent scores by maximizing

$$\sum_{o_i \succ o_j \in \mathcal{L}} \log(\frac{e^{s_i}}{e^{s_i} + e^{s_j}}), \tag{13}$$

where $\mathcal{L}$ is a set of crowdsourced comparison results.

However, the BT model does not consider the crowd quality. As different workers have different quality, it is very important to tolerate errors in estimating the scores. To address this problem, Chen et al. [4] propose the CrowdBT model, which assumes that each worker w has a quality η_w. That is, if worker w returns $o_i \succ o_j$, the probability of $o_i \succ o_j$ is η_w and the probability of $o_j \succ o_i$ is $1 - \eta_w$. In addition, if the pairwise comparison graph is not strongly connected, Equation 13 may not get an accurate estimation for the objects. To solve this problem, Chen et al. [4] add a virtual object to the comparison graph with comparisons to other objects. The score for the virtual object is represented by s_0. Then it computes the scores by maximizing

$$\begin{aligned} &\sum_{w} \sum_{o_i \succ o_j \in \mathcal{L}_w} \log\big(\eta_w \frac{e^{s_i}}{e^{s_i} + e^{s_j}} + (1 - \eta_w)\frac{e^{s_j}}{e^{s_i} + e^{s_j}}\big) + \\ &\qquad \lambda \sum_{i=1}^{n} \big(\log(\frac{e^{s_0}}{e^{s_0} + e^{s_i}}) + \log(\frac{e^{s_i}}{e^{s_0} + e^{s_i}})\big), \end{aligned} \tag{14}$$

where $\mathcal{L}_w$ is the set of comparison results by worker w.

CrowdGauss with Gaussian Model [16]. Pfeiffer et al. [16] propose CrowdGauss that uses the gaussian model to estimate the score. It assumes that the score follows the gaussian distribution, where the score is the mean of the distribution. The probability of $o_i \succ o_j$, i.e., $P(o_i \succ o_j)$, can be computed by the cumulative distribution function (Φ) of the two standard gaussian distributions, i.e.,

$$P(o_i \succ o_j) = \Phi(s_i - s_j). \tag{15}$$

Then CrowdGauss computes the scores by maximizing

$$\sum_{o_i \succ o_j \in \mathcal{L}} M_{ij} \cdot \log(\Phi(s_i - s_j)). \tag{16}$$

4.2.2 Matrix Decomposition Based Method

HodgeRank [11]. In order to estimate a global order for n objects, HodgeRank [11] utilizes a matrix decomposition based technique. It first computes a preference matrix Y based on the crowd's answers, where

$$Y_{ij} = \begin{cases} ln(\frac{M_{ji}}{M_{ij}}) & \text{if } i \neq j \textit{ and } M_{ij} \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Then, it constructs another matrix X where $X_{ij} = s_j - s_i$. Lastly, it computes the score by minimizing

$$\sum_{o_i \succ o_j \in \mathcal{L}} (X_{ij} - Y_{ij})^2 = \sum_{o_i \succ o_j \in \mathcal{L}} (s_j - s_i - Y_{ij})^2. \tag{18}$$

4.2.3 Others

TrueSkill [10]. TrueSkill improves ELO by reducing the number of iterations as ELO needs to repeat many times to convergence. Different from ELO, the score of each object o_i is represented by a Gaussian distribution $N(s_i, \delta_i)$, where s_i is the estimated score for o_i and δ_i is the deviation of s_i. Let $\beta = \frac{\sum_i \delta_i}{n}$ denote the average deviation for all objects. $\mathcal{U}$ and $\mathcal{V}$ are two monotonically decreasing functions, which are used to update s_i and δ_i. $\mathcal{U}(\frac{s_i - s_j}{c}, \frac{\varepsilon}{c})$ is the reward score added to o_i or the punishment score subtracted from o_j, where ε is a predefined parameter. $\mathcal{V}(\frac{s_i - s_j}{c}, \frac{\varepsilon}{c})$ indicates the reduction of the deviation. For each comparison result $o_i \succ o_j$, it updates the scores and deviations as below.

$$\begin{gathered} s_i \leftarrow s_i + \frac{\delta_i^2}{c} \cdot \mathcal{U}(\frac{s_i - s_j}{c}, \frac{\varepsilon}{c}); \quad s_j \leftarrow s_j - \frac{\delta_j^2}{c} \cdot \mathcal{U}(\frac{s_i - s_j}{c}, \frac{\varepsilon}{c}) \\ \delta_i^2 \leftarrow \delta_i^2 \cdot [1 - \frac{\delta_i^2}{c^2} \cdot \mathcal{V}(\frac{s_i - s_j}{c}, \frac{\varepsilon}{c})]; \delta_j^2 \leftarrow \delta_j^2 \cdot [1 - \frac{\delta_j^2}{c^2} \cdot \mathcal{V}(\frac{s_i - s_j}{c}, \frac{\varepsilon}{c})], \end{gathered} \tag{19}$$

where $\beta = \frac{\sum_i \delta_i}{n}$, $c = \sqrt{2\beta^2 + \delta_i^2 + \delta_j^2}$, ε is a predefined parameter, and s_i, δ_i are learned from the crowdsourced pairs.

4.3 Extensions of Heap-Based Methods

TwoStageHeap [5]. A two-stage heap algorithm [5] is proposed to extend heap-based algorithms to support crowdsourced top-k computation in databases. In the first phase, the objects are divided into $\frac{n}{X}$ buckets (where $X = \frac{xn}{k^2}$) such that the probability of two top-k objects appearing in the same bucket is at most x (set to 0.1 by default). In each bucket, a tournament based max algorithm [5] is conducted to select the best object in the bucket. Each pair on top levels of the tournament is compared multiple times and each pair on low levels of the tournament is compared only once. The second phase utilizes a heap-based method [8] to identify the top-k results from these best objects. To tolerate errors, when constructing and re-heapifying the heap, each pair is compared by multiple workers and the algorithm uses the majority voting to obtain a combined preference. After popping an object from the heap, the algorithm asks next pairs following the re-heapifying order.

4.4 Combining Rating And Comparison

There are two algorithms [12,20] that combine rating and comparison. The methods in [12] are initially designed to find the maximal object, and they can be extended to address the top-k problem. They utilize the rating and comparison results to learn the score for each object. For rating, they pre-define τ categories and each category χ_c has a range $(\gamma_{c-1}, \gamma_c]$, where $\gamma_0 < \gamma_1 < \cdots < \gamma_\tau$. If the score of object o_i falls in $(\gamma_{c-1}, \gamma_c]$, o_i is in the category χ_c.

Combine [20]. It first selects some rating and comparison tasks, and then infers the scores based on these results. The score for each object o_i is modeled by $s_i + \varepsilon_i$, where $\varepsilon_i \sim N(0, \delta^2)$, which is utilized to model crowd errors.

For rating, o_i is in the category χ_c with probability of

$$P(\chi_c|s_i) = Pr(\gamma_{\chi_c - 1} < s_i + \varepsilon_i \leq \gamma_{\chi_c}) = \Phi(\frac{\gamma_{\chi_c} - s_i}{\delta}) - \Phi(\frac{\gamma_{\chi_c - 1} - s_i}{\delta}) \tag{20}$$

where Φ represents the Cumulative Density Function(CDF) of standard Gaussian distribution. Then it constructs a voting matrix V based on the voting result, where V_{ij} is the number of times that o_i is observed in the j-category.

It computes the probability of observing V given s by:

$$\begin{aligned} P(V|s) &= \prod_{i=1}^{n} Pr(V_{i1}, V_{i2}, \cdots, V_{i\tau}|s_i) \\ &= c_1 \prod_{i=1}^{n} \prod_{c=1}^{\tau} (\Phi(\frac{\gamma_c - s_i}{\delta}) - \Phi(\frac{\gamma_{c-1} - s_i}{\delta}))^{V_{i,c}} \end{aligned} \tag{21}$$

where c_1 is a constant.

As ε_i has a normal distribution, the pairwise preference can be modeled as,

$$P(o_i \succ o_j) = Pr(s_i + \varepsilon_i > s_j + \varepsilon_j) = \Phi(\frac{s_i - s_j}{\sqrt{2}\delta}) \tag{22}$$

For comparison, it constructs the comparison matrix M and computes the probability of observing M by:

$$P(M|s) = \prod_{i<j\in 1,\cdots n} Pr(M_{ij}, M_{ji}|s_i, s_j) = c_2 \prod_{i \neq j} \Phi(\frac{s_i - s_j}{\sqrt{2}\delta})^{M_{ij}} \tag{23}$$

where c_2 is a constant.

Based on $Pr(V|s)$ and $Pr(M|s)$, it computes $Pr(s|V, M)$ based on the bayesian theory and estimates the score by

$$\hat{s} = \arg\max_s Pr(s|V, M) \tag{24}$$

Monte Carlo methods can be used to compute Equation 24, but the computation cost is rather high. [20] approximates $Pr(s|V, M)$ by $Pr(s|V, M, \hat{\gamma})$, where $\hat{\gamma} = \arg\max_\gamma Pr(V, M|\gamma)$. Utilizing the Laplace approximation, an analytical form can be formulated for $Pr(V, M|\gamma)$, and the maximum likelihood estimation of γ can be computed by gradient based optimization methods. After obtaining γ, s can be estimated by maximizing a posteriori probability (MAP) estimation.

Hybrid [12]. It first crowdsources all rating tasks and then selects some candidate objects with higher ratings. Then it chooses some pairs from the candidates as comparison tasks. The score of each object o_i is modeled as a normal variable $N(s_i, \delta^2)$. Given the rating results E_R and comparison results E_C, it estimates the score by

$$\hat{s} = \arg\max_s P(E_R, E_C|s) \tag{25}$$

If the questions are answered independently, the above function can be decomposed to

$$\begin{aligned} P(E_R, E_C|s) &= P(E_R|s)P(E_C|s) \\ P(E_R|s) &= \prod_{i=1}^{n} \prod_{j=1}^{|R_i^*|} P(R_i^j|s_i) \\ P(E_C|s) &= \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \prod_{t=1}^{|M_{ij}+M_{ji}|} P(C_{ij}^t|s_i, s_j) \end{aligned} \tag{26}$$

where $|R_i^*|$ is the number of ratings for object o_i, R_i^j is the assigned category in the j-th rating (by assuming each category has equal width and $\gamma_0 = 0, \gamma_\tau = 1$). Thus, the probability that object o_i lies in category χ_c is computed as

$$P(R_i^j = \chi_c | s_i) = \begin{cases} F_{s_i,\delta^2}(\frac{c}{\tau}) & c = 1 \\ F_{s_i,\delta^2}(\frac{c}{\tau}) - F_{s_i,\delta^2}(\frac{c-1}{\tau}) & 1 < c < \tau \\ 1 - F_{s_i,\delta^2}(\frac{c-1}{\tau}) & c = \tau \end{cases} \tag{27}$$

where $F_{s_i,\delta^2}(x)$ is the CDF of s_i and δ.

Let C_{ij}^t denote the comparison result in the t-th comparison for pair (o_i, o_j). $C_{ij}^t = 0$ if o_j is superior to o_i, and $P(C_{ij} = 0) + P(C_{ij} = 1) = 1$. Since o_i and o_j both follow the normal distribution, $o_i - o_j \sim N(s_i - s_j, 2\delta^2)$,

$$P(C_{i,j}^t = 0 | s_i, s_j) = F_{s_i - s_j, 2\delta^2}(0). \tag{28}$$

HybridMPR. However, the maximum likelihood estimation is rather expensive, thus a modified PageRank approximation [12] is proposed to estimate the score for each object. It defines the average rating score r_i for each object o_i as,

$$r_i = \begin{cases} dv & \text{if } |R_i^*| = 0 \\ \frac{\sum_j R_i^j}{\tau |R_i^*|} - \frac{1}{2\tau} & otherwise \end{cases} \tag{29}$$

where dv is a default value (e.g., 0.5) for the case of no rating on o_i. Then the modified PageRank is defined as below,

$$\rho_i = \beta \sum_{j: w_j \neq 0} \frac{w_{j,i}}{w_j} \rho_j + (1 - \beta) \frac{r_i}{\sum_j r_j} \tag{30}$$

where β is the fraction of comparison questions to the total questions, $w_j = \sum_i w_{j,i}$, and $w_{j,i}$ is defined as

$$w_{j,i} = \begin{cases} 1 & \text{if } M_{ij} + M_{ji} = 0 \ and \ r_i \geq r_j \\ 0 & \text{if } M_{ij} + M_{ji} = 0 \ and \ r_i < r_j \\ \frac{M_{ij}}{M_{ij} + M_{ji}} & otherwise \end{cases} \tag{31}$$

5. PAIR SELECTION

This section discusses different pair selection methods. The random selection method that randomly selects next b pairs is not effective [19], because different comparison pairs have different importance to infer the results. For example, in Figure 2, suppose we select pairs (o_1, o_2), (o_2, o_5), (o_1, o_5). As $o_1 \succ o_2$ and $o_2 \succ o_5$, (o_1, o_5) does not need to be asked as its result can be deduced. Similarly, (o_1, o_6) and (o_2, o_6) do not need to be crowdsourced. Thus it is important to select high-quality pairs. We first introduce the heuristic-based methods in Section 5.1 and then discuss the bound-based methods in Section 5.2. Lastly, we discuss the active-learning methods in Section 5.3.

5.1 Heuristic-Based Methods

Guo et al. [9] prove that selecting the pairs to maximize the probability of obtaining the top-k results (given the comparison results of arbitrary crowdsourced pairs) is NP-Hard and propose four heuristics, which are designed for selecting the max (top-1) result. The proposed algorithms first compute a score s_i for object o_i (see Section 4.1). Suppose the sorted objects based on the scores are $o_1, o_2, \cdots, o_n$. Then, the algorithms select the next b pairs as follows.

Max. It selects b pairs: (o_1, o_2), (o_1, o_3), $\cdots$, (o_1, o_{b+1}).

Group. It groups the *i-th* object with the $(i+1)$-*th* object and the selected pairs are (o_1, o_2), (o_3, o_4), $\cdots$, (o_{2b-1}, o_{2b}).

Greedy. It selects the pairs based on $s_i \times s_j$ in descending order, and selects b pairs with the largest value.

Complete. It has two phases. The first phase finds the maximal number of objects and makes a pairwise comparison on them. Thus the first phase finds top-x objects with the highest scores, where x is the largest integer satisfying $\frac{x*(x-1)}{2} \leq b$. The first phase compares every two objects among the first x objects. If the budget is not exhausted in the first phase (i.e., $b > \frac{x*(x-1)}{2}$), then the second phase compares the first $b - \frac{x*(x-1)}{2}$ objects with the $(x+1)$-th object, i.e., $(o_1, o_{x+1}), (o_2, o_{x+1}), \cdots, (o_{b-\frac{x*(x-1)}{2}}, o_{x+1})$.

5.2 Bound-Based Methods

SSCO and SSSE estimate a bound for each pair and utilize the bound to select next pairs [3]. Formally, SSCO and SSSE first compute a confidence interval $[l_{ij}, u_{ij}]$, where l_{ij} (u_{ij}) is the lower (upper) bound of the probability of $o_i \succ o_j$, which can be computed as below.

$$u_{ij} = \frac{M_{ij}}{M_{ij} + M_{ji}} + c_{ij}; \quad l_{ij} = \frac{M_{ij}}{M_{ij} + M_{ji}} - c_{ij}, \tag{32}$$

where $c_{ij}=\sqrt{\frac{1}{2(M_{ij}+M_{ji})} \log \frac{2n^2 n_{max}}{q}}$, n_{max} is a predefined maximal number of crowdsourced tasks for each pair and q is a confidence factor that top-k objects can be found with probability $1-q$. Based on the confidence interval, they select a set S and discard a set D. Since the relationships between pairs in $S \cup D$ have been effectively captured, these pairs do not need to be compared. They propose two algorithms to select pairs that are not in $S \cup D$.

Sampling Strategy with Copeland's Ranking(SSCO) [3]. Based on the confidence interval $[l_{ij}, u_{ij}]$, it computes a bound $L_i = |\{o_j | l_{ij} > 1/2\}|$ of object o_i, which is the number of objects with high probability of $o_i \succ o_j$, and a bound $U_i = |\{o_j | u_{ij} < 1/2\}|$ of object o_i, which is the number of objects with high probability of $o_j \succ o_i$. Given two objects o_i and o_j, if $L_i > n - U_j$, o_i should be preferred to o_j. If $|\{j | L_i > n - U_j\}| > n - k$ (i.e., the number of objects beaten by o_i is larger than $n - k$), o_i will have high probability in the top-k results and is added into set S, i.e.,

$$S = \{o_i : |\{j | L_i > n - U_j\}| > n - k\}. \tag{33}$$

Similarly, if $U_i > n - L_j$, o_j should be preferred to o_i. If $|\{j | U_i > n - L_j\}| > k$ (i.e., the number of objects preferred to o_i is larger than k), o_i has small probability in the top-k results and is added into the set D, i.e.,

$$D = \{o_i : |\{j | U_i > n - L_j\}| > k\}. \tag{34}$$

Sampling Strategy with Sum of Expectations(SSSE) [3]. This method utilizes the sum of expectations to compute the two bounds, and $L_i = \frac{1}{n-1} \sum_{j \neq i} l_{ij}$ is the expectation of lower bound of o_i and $U_i = \frac{1}{n-1} \sum_{j \neq i} u_{ij}$ is the expectation of upper bound of o_i. If $L_i > U_j$, o_i should be preferred to o_j. If $|\{j | L_i > U_j\}| > n - k$ (i.e., the number of objects beaten by o_i is larger than $n - k$), o_i will have high probability in the top-k results and is added into set S, i.e.,

$$S = \{o_i : |\{j | L_i > U_j\}| > n - k\}. \tag{35}$$

Similarly, if $U_i < L_j$, o_j should be preferred to o_i. If $|\{j | U_i < L_j\}| > k$ (i.e., the number of objects preferred to

o_i is larger than k), o_i will have small probability in the top-k results and is added into the set D, i.e.,

$$D = \{o_i : |\{j|U_i < L_j\}| > k\}. \tag{36}$$

Sampling Strategy based on Random Walk(SSRW) [3]. SSRW uses a random walk based method to select next pairs. It defines a matrix $\bar{Y}$, where each of its element $\bar{y}_{i,j}(\bar{y}_{i,j} = \frac{M_{i,j}}{M_{i,j}+M_{j,i}})$ is the estimated probability that o_i is preferred to o_j. The real probabilities for pairwise comparisons are represented by matrix Y, and the accurate value for matrix Y is unknown. To utilize random walk to address the top-k problem, the method constructs a stochastic matrix $\bar{S}$ for $\bar{Y}$. Each element in $\bar{S}$ is computed as $\bar{s}_{i,j} = \frac{\bar{y}_{i,j}}{\sum_l \bar{y}_{l,i}}$. Besides, the stochastic matrix for Y is defined as S and $s_{i,j} = \frac{y_{i,j}}{\sum_l y_{l,i}}$. Assuming the principal eigenvectors for $\bar{S}$ and S are $\bar{v}$ and v. The method makes a statement that

$$\parallel v - \bar{v} \parallel_{max} \leq \parallel S - \bar{S} \parallel_1 \parallel \bar{A}^{\#} \parallel_{max} \tag{37}$$

where $\bar{A}^{\#} = (I - \bar{S} + 1\bar{v}^T)^{-1} - 1\bar{v}^T$. And $\parallel \bar{A}^{\#} \parallel_{max}$ is converged to $\parallel A^{\#} \parallel_{max}$. As $\parallel A^{\#} \parallel_{max}$ is bounded, $\parallel S - \bar{S} \parallel_1 \leqslant \frac{n}{3} \arg\max_{i,j} c_{i,j} \sum_l \bar{y}_{l,i}$. To minimize Equation 37, a sampling strategy is used to minimize $\parallel S - \bar{S} \parallel_1$, and it selects pairs $(i, j) = \arg\max_{i,j} c_{i,j} \sum_l \bar{y}_{l,i}$, where $c_{ij} = \sqrt{\frac{1}{2(M_{ij}+M_{ji})} \log \frac{2n^2 n_{max}}{q}}$.

5.3 Active-Learning Methods

Active-learning methods select next pairs by maximizing the information gain based on the estimated latent scores.

5.3.1 Active Learning

CrowdGauss [16]. The scores can be modeled by a multivariate Gaussian distribution $N(\hat{s}, \mathcal{C})$, where $\hat{s}$ is a $1 \times n$ matrix indicating the score for all the objects (initialized by random values), and $\mathcal{C}$ is the covariance matrix of $\hat{s}$ ($\mathcal{C}_{ij}$ is the value at i-th row and j-th column). In each round of pair selection, the expected information gain g_{ij} for each pair (o_i, o_j) is computed. For pair (o_i, o_j), $N(\hat{s}^{ij}, \mathcal{C}^{ij})$ is the reevaluation distribution by assuming $o_i \succ o_j$, and p_{ij} is the probability of $o_i \succ o_j$ estimated in the previous iteration. The expected information gain is computed as:

$$g_{ij} = p_{ij} \cdot \mathtt{KL}\big(N(\hat{s}^{ij}, \mathcal{C}^{ij}), N(\hat{s}, \mathcal{C})\big) + p_{ji} \cdot \mathtt{KL}\big(N(\hat{s}^{ji}, \mathcal{C}^{ji}), N(\hat{s}, \mathcal{C})\big), \tag{38}$$

where $p_{ij} = \Phi\left(\frac{s_i - s_j}{1+\mathcal{C}_{ii}+\mathcal{C}_{jj}-2\mathcal{C}_{ij}}\right)$ and `KL` is the Kullback-Leibler divergence. At each iteration, it selects the pair with the largest expected information gain and updates $\hat{s}$ and $\mathcal{C}$.

CrowdBT [4]. The above method does not consider the worker quality. Chen et al. [4] propose an active-learning method by taking into account the worker quality. The score of object o_i is modeled by a Gaussian distribution $N(s_i, \delta_i)$ and the quality of worker w is modeled by a Beta distribution $\mathtt{Beta}(\alpha_w, \beta_w)$. Each time it computes the probability of worker w on $o_i \succ o_j$, denoted by $P(o_i \succ_w o_j)$, as below.

$$P(o_i \succ_w o_j) = \eta_w \frac{e^{s_i}}{e^{s_i} + e^{s_j}} + (1 - \eta_w) \frac{e^{s_j}}{e^{s_i} + e^{s_j}}. \tag{39}$$

It uses a parameter γ to tune the comparison result and worker quality. It assigns (o_i, o_j) to worker w by maximizing the probability of w reporting a correct result on (o_i, o_j).

5.3.2 Combining Rating and Pairwise Comparisons

Combine [20]. Instead of just utilizing pairwise comparisons, Ye et al. [20] propose an active-learning strategy by combining rating and comparison together. Each time with a budget b, it selects a subset of rating-based questions and some comparison-based questions to maximize the expected information gain Z^*. The rating-based method asks the crowd to assign an object to a rate, and the rating-based information gain is computed as below.

$$gr(i) = \mathtt{E}\Big(\sum_{r=1}^{M} p_{ir} \log(\frac{p_{ir}}{p(x_i = r)})\Big) \tag{40}$$

where p_{ir} is the probability that the crowd assigns o_i with rate r. $p(x_i{=}r)$ is the prior probability for o_i with rate r.

The comparison based information gain is computed as

$$gc(i, j) = \mathtt{E}\big(p_{ij} \log(\frac{p_{ij}}{p(x_{ij} = 1)}) + q_{ij} \log(\frac{q_{ij}}{p(x_{ij} = 0)})\big), \tag{41}$$

where p_{ij} is the probability of $o_i \succ o_j$ based on crowdsourced comparisons, $q_{ij} = 1 - p_{ij}$, $p(x_{ij} = 1)(p(x_{ij} = 0))$ is the prior probability of $o_i \succ o_j$ ($o_j \succ o_i$). Assuming the cost for rating (comparison) is C_r (C_c), which are set to 1, it selects tasks with the maximal information gain.

6. EXPERIMENTAL STUDY

We evaluated all inference and selection methods. All the source code and real datasets were available at `http://dbgroup.cs.tsinghua.edu.cn/ligl/crowdtopk/`.

6.1 Experimental Setting

6.1.1 Real Experiments

Existing works selected datasets based on the following characters: difficulty (easy or hard for workers), task types (images or texts), objective/subjective. We utilized these features to select datasets and Table 3 showed the details.

`PeopleAge`[1]. It included 50 human photos with ages from 50 to 100. We asked the crowd to judge which one was younger.

`PeopleNum` [12]. It contained 39 images taken in a mall. Each image contained multiple people, where the number of people varied from 13 to 53. We asked the crowd to judge which one contained more people.

`EventTime`. It contained 100 history events, e.g., "Germany invades Poland starting World War II" and "Napoleon defeated at Waterloo". Each event happened in different years. The crowd was asked to judge which event happened earlier.

`ImageClarity`. It contained 100 images with different clarities. The crowd was asked to judge which image was clearer.

The datasets had different levels of difficulties. `ImageClarity` was the easiest, and `EventTime` was the hardest. `PeopleAge` and `PeopleNum` had medium difficulty. `EventTime` used textual tasks while `ImageClarity`, `PeopleAge` and `PeopleNum` used image tasks. `PeopleAge` was subjective because different workers might have different judgements on people age while `EventTime`, `PeopleNum` and `ImageClarity` were objective as the tasks had clear answers.

We conducted the real experiments on a real crowdsourcing platform, CrowdFlower(www.crowdflower.com). The price of each microtask was 0.05$. To obtain high-quality workers, we used 5 qualification tests on each dataset.

6.1.2 Simulation Experiments

To further evaluate different methods on larger datasets, we generated a dataset with 1000 objects with scores from 1 to 1000 and there were totally 499,500 pairs. (Note that if we used 10000 objects, there would be 49,995,000 pairs and most of algorithms cannot support such large dataset.) We varied the worker accuracy in {60%, 70%, 80%, 90%}.

[1] http://www.edouardjanssens.com/art/1-to-100-years/men/

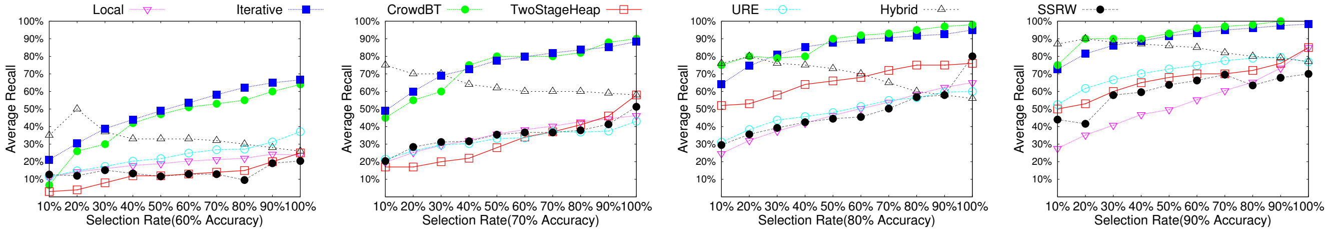


Figure 3: Recall on Simulation for Selected Inference Methods: Varying Selection Rate (1000 Objects, $k = 10$).

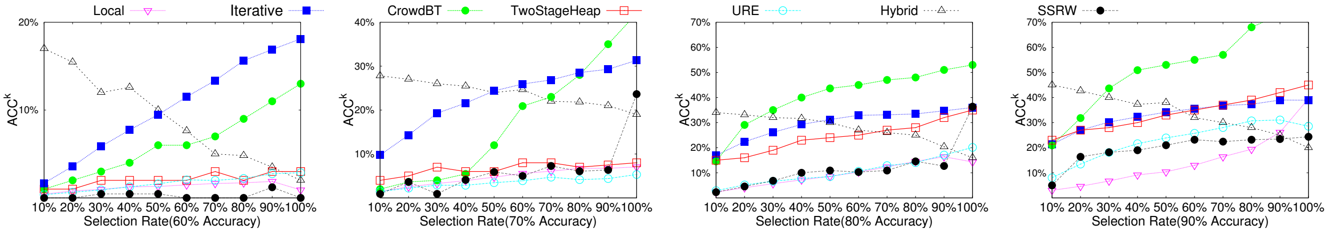


Figure 4: ACC^k on Simulation for Selected Inference Methods: Varying Selection Rate (1000 Objects, $k = 10$).

Table 3: Worker Accuracy on Real Datasets (c-Accuracy: Comparison; r-Accuracy: Rating).

Datasets	#Objects	#Pairs	c-Accuracy	#Ratings	r-Accuracy
PeopleAge	50	1225	85.2%	50	100%
PeopleNum	39	741	88.4%	39	71.8%
EventTime	100	4950	76.6%	100	89.4%
ImageClarity	100	4950	98.6%	100	97%

6.1.3 Setting

Microtasks. For comparison-based methods, we generated comparison tasks to compare each object pair. For hybrid methods that combined rating and comparison, we used two types of tasks: comparison and rating. The rating task asked the worker to select a category for each object. We used five categories for each dataset. For example, the five categories on the PeopleAge dataset were '50-60', '60-70', '70-80', '80-90', and '90-100'. Each task was assigned to 3 workers. For comparison, we combined the comparison results of the 3 workers using majority vote and computed voted comparison accuracy, denoted by c-accuracy. For rating, we used the majority vote to combine the result and computed voted rating accuracy, denoted by r-accuracy.

Quality Metrics. We used two metrics to evaluate the quality. Let A^k and T^k respectively denote the top-k results by an algorithms and the real top-k results. (1) Recall. It was computed by $\frac{|A^k \cap T^k|}{k}$. (2) ACC^k. It considered the orders of the returned results and was computed as

$$ACC^k = \frac{\sum_{o_i, o_j \in T^k \cap A^k} \mathbb{I}(a_i < a_j \wedge t_i < t_j)}{k(k+1)/2} \tag{42}$$

where t_i was the real ranking position for object o_i and a_i was the ranking position by the algorithm. $\mathbb{I}$ was a function, which returned 1 if the condition was true; 0 otherwise.

Setting. All algorithms were implemented by Python 2.7.6. The parameters were used the same as in the original paper as shown in Table 2. All experiments were conducted on a Macbook Pro with 2.3GHz Intel i7 CPU and 16GB RAM.

6.2 Evaluation of Inference Methods

The number of crowdsourced pairs could affect the quality of inference methods. We varied the selection rate which was the ratio of the number of selected pairs to the total number of pairs, to compare the inference algorithms.

6.2.1 Simulation – Varying Selection Rate

We first compared the inference methods by randomly selecting crowdsourced pairs, and then evaluated different selection strategies in Section 6.3. Here we showed the results of selected representative methods from each category and the results of all the methods were in the full version[2]. For the heuristic-based algorithms, we selected Local, URE, SSRW, and Iterative as they achieved higher performance. For the machine-learning methods, we selected CrowdBT which outperformed other methods. For hybrid methods, we selected Hybrid. We also compared with TwoStageHeap. Thus we showed the results of these seven algorithms. Figure 3 showed the recall and Figure 4 showed ACC^k.

We had the following observations. First, for low worker quality (e.g., the worker accuracy of 60%), all the algorithms had low recall. For example, in Figure 3(a), the recall for most of the algorithms were smaller than 60%, because for low worker accuracy, there were many incorrect comparisons and it was hard to correct many errors.

Second, for high worker quality (e.g., the worker quality larger than 70%), the machine-learning methods outperformed most of the heuristic-based methods, as the machine-learning methods utilized the global comparison results to infer the top-k answers while the heuristic-based approaches only used the local comparison results, e.g., in-degree and out-degree. If the comparison results on local pairs were incorrect, they would significantly affect the performance while the machine-learning methods tolerated local errors.

Third, among the heuristic-based methods, the iterative inference algorithms (e.g., Iterative, AdaptiveReduce) were better than the other heuristic inference algorithms(e.g., BordaCount, Copeland, Indegree, ELO), and even outperformed the machine-learning algorithms, because the iterative algorithms iteratively reduced objects that were unlikely in the top-k answers. In addition, Local was better than BordaCount and Copeland, as Local utilized 2-hop neighbors while BordaCount and Copeland only utilized the direct neighbors.

Fourth, TwoStageHeap was worse than others when the worker accuracy was low, while it had a good performance when the accuracy was high. The main reason was that for low worker quality, it might miss some top-k candidates in the first step due to the comparison errors from the crowd. URE had a better performance than BRE when the average worker accuracy was above 70%. URE was more robust than BRE as URE computed the number of losses for each object, rather than computing the relative difference of the number

[2] http://dbgroup.cs.tsinghua.edu.cn/ligl/crowdtopk/topk.pdf

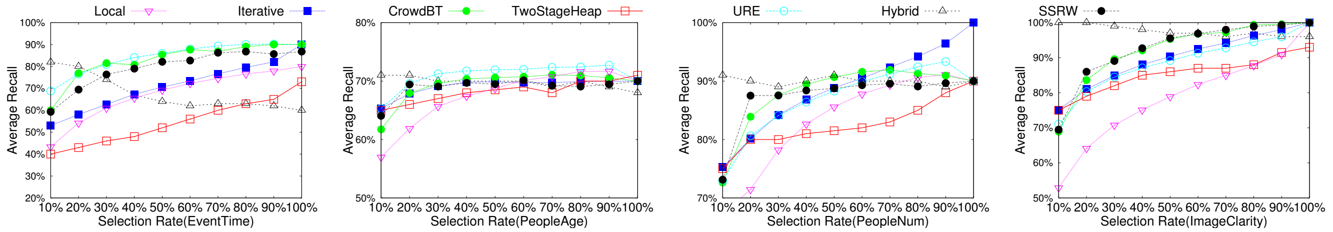


Figure 5: Recall on Real Datasets for Selected Inference Methods: Varying Selection Rate ($k = 10$).

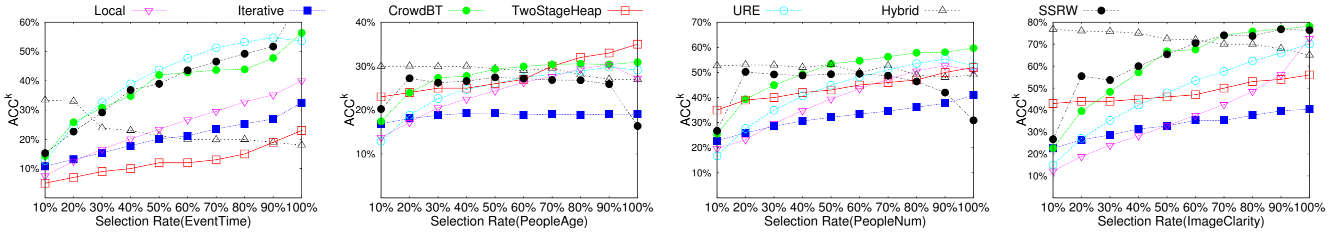


Figure 6: ACC^k on Real Datasets for Selected Inference Methods: Varying Selection Rate ($k = 10$).

of wins and losses for each specific object. As the number of pairwise comparisons was not sufficient, the losses for each object could bring more information than the relative difference of the wins and losses. Among the three bound-based algorithms, SSCO and SSSE had poor performance as they utilized local information to update the score matrix. SSRW outperformed SSCO and SSSE, as SSRW utilized random walk to capture global comparison results.

Fifth, for the machine-learning methods, CrowdBT outperformed other methods. CrowdBT had better performance than CrowdGauss, because CrowdBT not only considered the worker quality, but also utilized the graph structure to facilitate the inference. HodgeRank and SSRW had lower recall, because the matrix decomposition had deviations and led to errors to compute latent scores.

Sixth, AdaptiveReduce had lower recall than other methods for small selection rates but achieved comparable and even higher recall for large selection rates. This was because for small selection rates, AdaptiveReduce cannot get high-quality informative set (which required to find a subset with voting account between $\frac{n}{4}$ and $\frac{3n}{4}$, and there were no enough comparisons for small rates); for large selection rates, AdaptiveReduce identified a high-quality informative set based on lots of comparison results. PathRank had poor recall because it required that the graph was strongly connected and had no circles, which made it not robust.

Seventh, for the hybrid algorithms, Hybrid achieved higher performance for a low selection rate, because it greatly reduced the candidates with little cost and utilized global pairwise comparisons in the second stage to infer the top-k answers. Hybrid achieved a good performance for high worker accuracy, because its quality was greatly affected by the worker accuracy and if the worker accuracy was low, its first stage could not guarantee that all the top-k objects were selected into the candidates. Combine was worse than Hybrid, as Combine did not consider worker accuracy. HybridMPR was worse than Hybrid because its estimated scores were worse than the scores learned by Hybrid.

Eighth, with the increase of the selection rates, the comparison algorithms achieved higher recall, as more crowdsourced pairs were used to infer the final answers. However for the hybrid method, the quality first increased and then decreased, because for a larger selection rate, it involved more candidates and the second phase had to consider more pairs to infer the top-k results. The results were also consistent with those in the original paper [12].

Ninth, ACC^k was smaller than recall, as ACC^k considered the orders of returned results. The methods computing scores for each object were better than the adaptive-reduce methods (e.g., AdaptiveReduce, PathRank, TwoStageHeap), because the former used the score to infer an order for the objects while the latter could not get any order.

Tenth, with the increase of selection rates, the recall increased, but the improvement after 30% selection rates was not significant. In other words, 30% selection rates had similar recall with 100% selection rates. Thus if we had a monetary budget, we could select 30% pairs.

Although machine-learning methods had higher recall than some heuristic-based algorithms, the superiority was not significant and they were even worse than Iterative, as some heuristics also considered global information. The heuristics achieved as high recall as the machine-learning methods if they effectively utilized global comparison results.

6.2.2 Real Experiments – Varying Selection Rate

We compared different algorithms on real datasets by varying selection rates. Figures 5 and 6 illustrated the results for the selected algorithms. We had the following observations.

First, the observations on the simulation experiments were still true on the real datasets. On the `EventTime` dataset, CrowdBT, CrowdGauss, TrueSkill had similar performance, which outperformed other methods. Among all heuristic-based methods, URE and SSRW had the highest recall, as URE could tolerate more noise and SSRW could utilize global information based on random walk. Second, on the `PeopleAge` dataset, URE was slightly superior to the other methods, because if the noise mainly came from the dataset (as workers had diverse judgements on people's ages), the simple heuristics could have a competitive performance with the machine-learning methods. Third, on the `ImageClarity` dataset, the algorithms based on random walk and maximum likelihood estimation had much better recall than the heuristic-based methods. For AdaptiveReduce, when the selection rate was low, it got low recall; however with a high selection rate, its performance was significantly improved, because if more pairwise results were collected, AdaptiveReduce could utilize more comparison results to do effective pruning. Fourth, on the `PeopleNum` and `ImageClarity` datasets, as the workers' accuracy was high, all the algorithms could get higher recall.

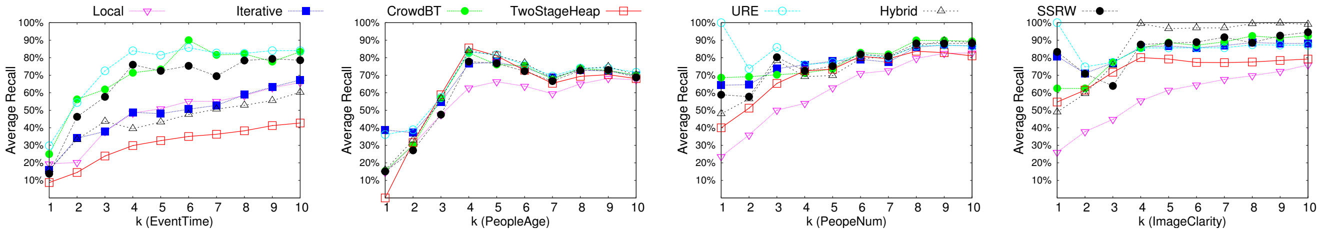


Figure 7: Recall on Real Datasets for Selected Inference Methods: Varying k.

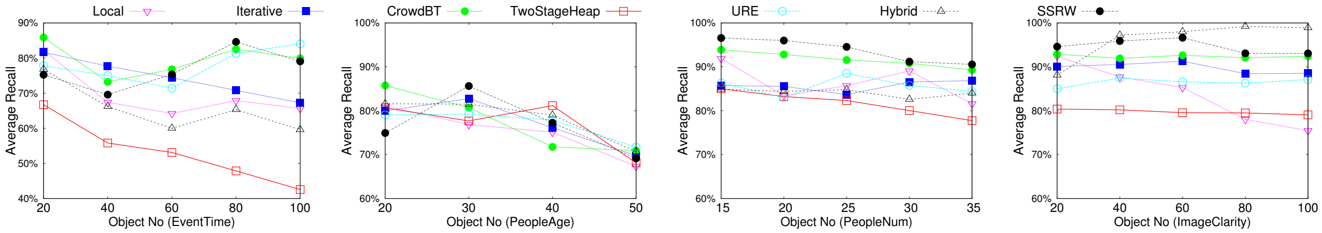


Figure 8: Recall on Real Datasets for Selected Inference Methods: Varying Object No ($k = 10$).

For example, on the ImageClarity dataset, the recall could reach 90%, even for small rates. Fifth, the hybrid method achieved higher quality for small selection rates, as it used rating to eliminate lowly ranked objects with little cost and utilized the pairwise comparisons to compare highly ranked objects. With the increase of the selection rate, more pairs would be selected into the second stage, and more noise were brought by these pairs. Sixth, for subjective tasks, e.g., PeopleAge, we found that the quality could not be improved even with a very high selection rate, because workers had diverse judgements on images with close ages. For objective tasks, the quality was consistently improved with the increase of selection rates.

6.2.3 *Real Datasets – Varying k*

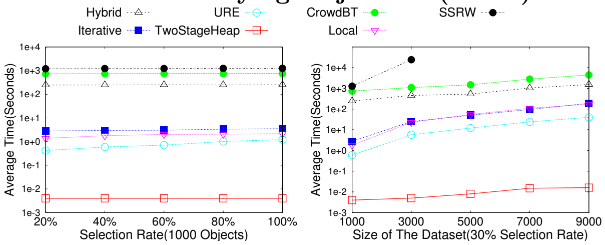
We varied the number of reported results, k. Figure 7 showed the results. We found that a large k had higher recall than a small k, as the crowd usually returned inaccurate results for hard comparison pairs (with close scores) and correct results for easy comparison pairs (with significantly different scores). The ratio of incorrect pairs to k decreased as k increased. Thus a small k was harder than a large k.

6.2.4 *Real Datasets – Varying Object No*

We varied the number of objects. Figure 8 showed the results. We found that with the increase of the object number, the recall of some algorithms decreased, because they had to consider more pairs and it become harder to get top-k results from a larger dataset (than a smaller dataset). This was consistent with the case of decreasing k.

6.2.5 *Efficiency And Scalability*

We evaluated the efficiency of different algorithms. We only reported the time used to infer the top-k answers and the time spent by workers was not included. We generated 1000 objects as discussed in Section 6.1.2. Figure 9 showed the results. First, the heuristics had higher efficiency and scalability than the machine-learning algorithms, because the latter involved huge computations of utilizing the global comparison results to infer latent scores. Second, the local inference heuristics had better efficiency and scalability than the global inference heuristics. Third, with the increase of selection rates, the time increased as they considered more pairs. The machine-learning methods slightly increased as they considered all pairs for every selection rate.



(a) Efficiency (b) Scalability

Figure 9: Efficiency/Scalability for Inference.

6.2.6 *Takeaways: Inference Algorithm Suggestions*

We had the following suggestions that can guide practitioners to select appropriate inference algorithms.

(1) If the user preferred high quality with small selection rates, the Hybrid method was recommended; If the user preferred a higher quality with larger selection rates, we recommended the machine-learning algorithms, e.g., CrowdBT.

(2) If the user preferred high quality with high efficiency, we recommended iterative inference heuristics, e.g., Iterative.

(3) If efficiency was crucial, we recommended TwoStageHeap.

(4) We recommended 30% selection rate.

6.3 Evaluation on Selection Methods

6.3.1 *Real Datasets – Varying Selection Rate*

We first varied the selection rates. For each selection rate, we utilized different methods to select pairs and adopted different inference algorithms to infer the top-k answers based on comparison results on these selected pairs. Figures 10-13 showed the results for using CrowdBT and Iterative as inference methods. We had the following observations.

First, the active-learning methods had higher quality than random selection on all the datasets, because they judiciously selected the pairs with high possibility in the top-k results, which were better than the randomly chosen ones. Second, the four heuristics were even worse than the random method, as these four methods were specifically devised to select the top-1 object and cannot effectively find the top-k objects. Even for large selection rates, they still could not achieve high quality, as they selected many duplicated pairs. Third, SSCO, SSSE and SSRW were slightly better than Random, because they could use the estimated scores to remove some unnecessary pairs but they could not estimate an accurate bound with limited number of comparisons. Fourth, Combine achieved high quality as it utilized

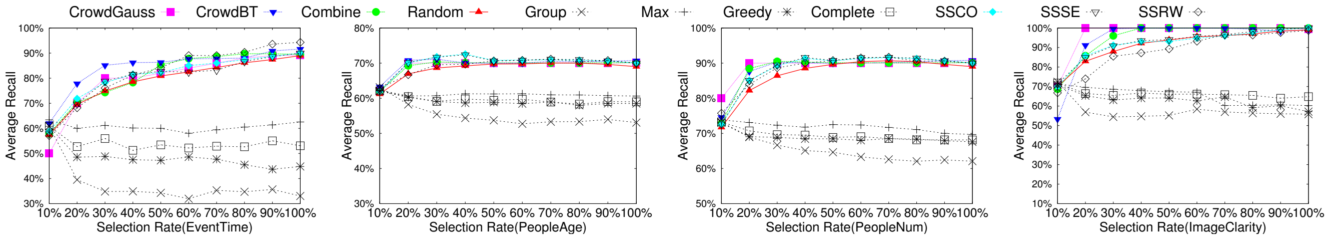


Figure 10: Recall of Selection Method (Inference Method: CrowdBT).

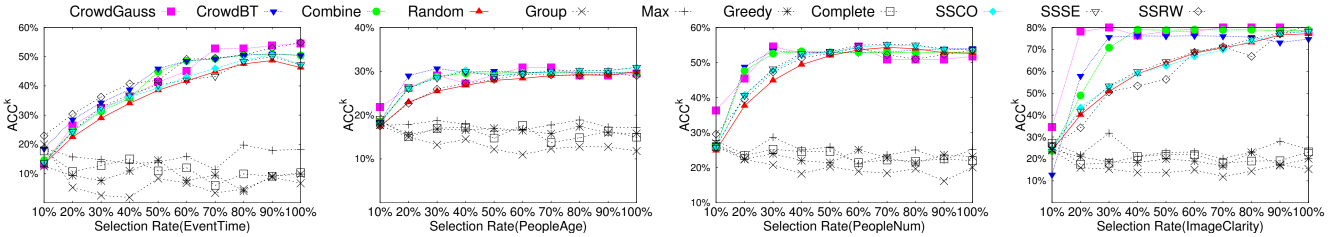


Figure 11: ACC^k of Selection Methods (Inference Method: CrowdBT).

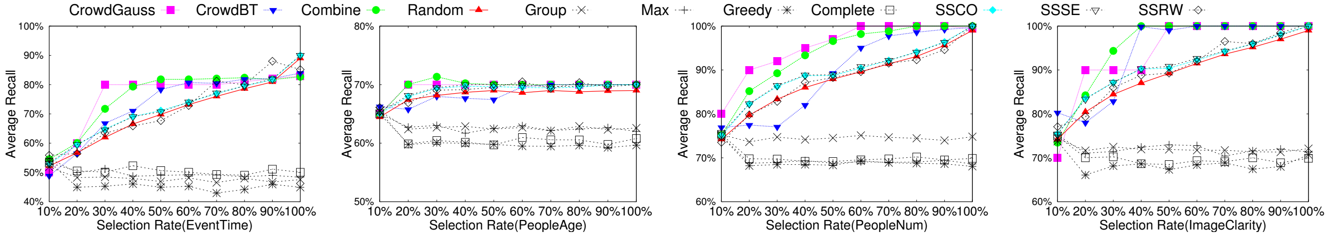


Figure 12: Recall of Selection Methods (Inference Method: Iterative).

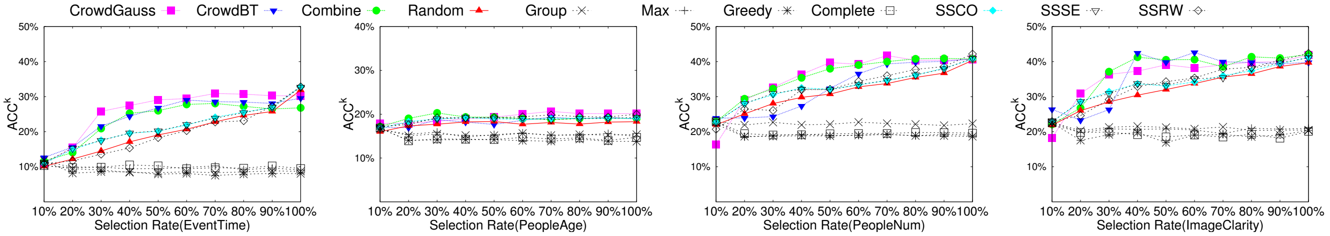


Figure 13: ACC^k of Selection Methods (Inference Method: Iterative).

the rating tasks to select high-quality rating or comparison tasks. Fifth, the recall of the selection methods became stable when the selection rates were larger than 30% on all the datasets, because the selection strategy was not important for large selection rates as they could utilize more pairs. Sixth, on the `PeopleNum` and `ImageClarity` datasets, the recall of all the methods were high even with a low selection rate due to the high worker accuracy on these datasets.

6.3.2 *Real Datasets – Varying k*

We varied the number of reported results, k. We used CrowdBT as the inference method. Figure 14 showed the results. With a larger k, the average recall was higher, because a large k made the problem easy as the comparisons on many distinctively different objects were easier than the comparisons on few similar objects. In addition, the active-learning methods and bound-based methods still outperformed Random, which was better than the heuristics.

6.3.3 *Real Datasets – Varying Object No*

We varied the number of objects and still used CrowdBT as the inference method. The selection rate was 30%. Figure 15 showed the results. With the increase of the object number, the average recall was slightly decreased, because more objects were involved, which made the problem harder. This was also similar to the case of increasing k.

6.3.4 *Efficiency and Scalability*

We evaluated the efficiency and scalability of the selection algorithms. In each iteration, we selected 10 pairs. Figure 16 showed the results. The four heuristics had much higher efficiency and scalability than other methods, as the heuristics only sorted the objects based on their scores and returned the pairs with the highest scores. The active-learning algorithms had the worst efficiency and scalability, because they had to evaluate all the unselected pairs. Specifically, CrowdGauss involved huge matrix computation in each iteration. CrowdBT enumerated all possible combinations of tasks and workers. Combine did not consider workers and identified the tasks with the largest information gain. Among the three bound-based methods, SSCO and SSSE had higher efficiency as they only found the rows with high bounds, while SSRW took longer time than SSCO and SSSE because SSRW evaluated all the unselected pairs.

6.3.5 *Takeaways: Selection Algorithm Suggestions*

We had the following suggestions that can guide practitioners to select appropriate selection algorithms. (1) If the user preferred high quality, the active-learning method CrowdBT was recommended. (2) If the user preferred high efficiency, we recommended SSCO/SSSE. (3) For large selection rates (>30%), the random method was acceptable.

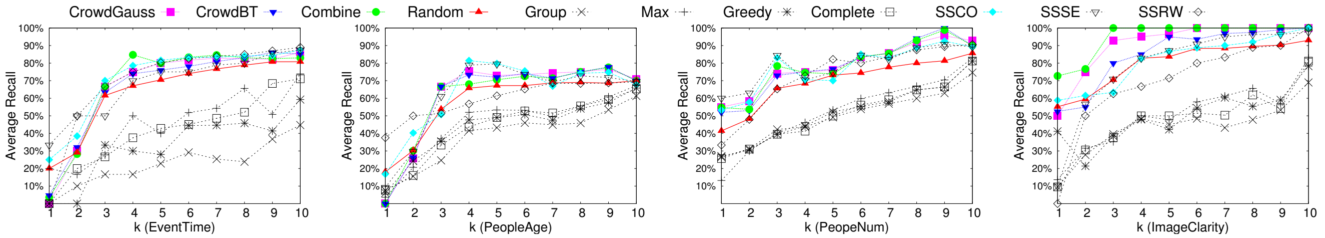


Figure 14: Recall on Real Datasets for Selection Methods: Varying k (Inference Method: CrowdBT).

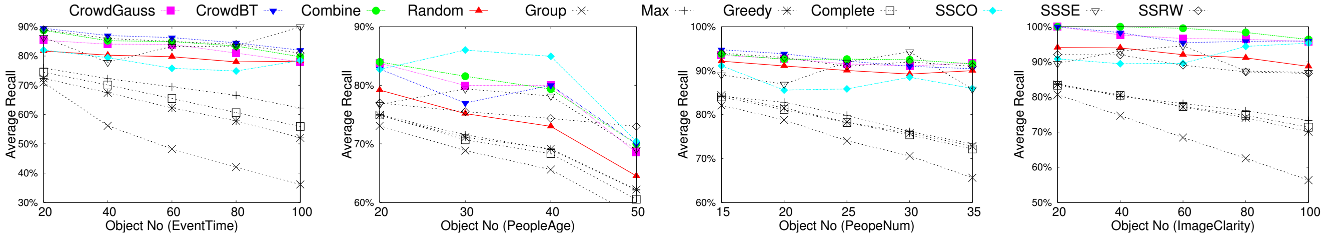
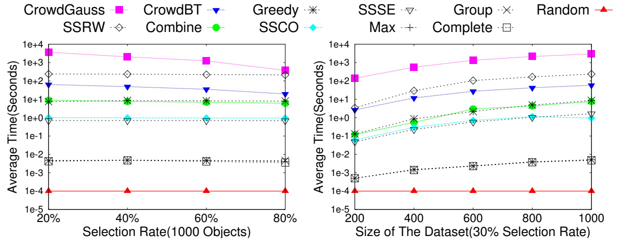


Figure 15: Recall on Real Datasets for Selection Methods: Varying Object No ($k = 10$, Inference: CrowdBT).



(a) Efficiency (b) Scalability

Figure 16: Efficiency/Scalability for Selection.

7. CONCLUSION

We provided a detailed survey on the crowdsourced top-k problem, discussed all of the comparison-based algorithms, rating-based algorithms and hybrid algorithms. Based on the experimental results, we also provided guidelines on selecting appropriate algorithms for various scenarios.

(1) The inference and selection algorithms were important to achieve high quality, especially for small selection rates.

(2) For result inference, the machine-learning methods achieved high quality. The global inference heuristics that utilized global comparison results achieved comparable and even higher quality than the machine-learning methods. The local inference heuristics had poor quality. However, the heuristics achieved higher efficiency and scalability. We had the following suggestions to select appropriate algorithms. (i) If the user preferred high efficiency, we recommended TwoStageHeap. (ii) If the user preferred high quality with low selection rates, we recommended Hybrid; If the user preferred a high quality with large selection rates, we recommended the machine-learning algorithms, e.g., CrowdBT; (iii) If the user preferred high quality with acceptable efficiency, we recommended the iterative inference heuristics, e.g., Iterative.

(3) For pair selection, the active-learning methods and bound-based methods achieved higher quality than heuristics. However the active-learning methods had lower efficiency and cannot meet the online requirement. We had the following suggestions. (i) We recommended 30% selection rate; (ii) If the user preferred high efficiency, we recommended SSCO/SSSE; (iii) If the user preferred quality, the active-learning method CrowdBT was recommended. (iv) For large selection rates, the random method was acceptable.

Acknowledgement: This work was supported by the 973 Program of China (2015CB358700), the NSF of China (61422205, 61472198), Tsinghua-Tencent Joint Laboratory, Huawei, Shenzhou, FDCT/116/2013/A3, MYRG105(Y1-L3)-FST13-GZ, 863 Program of China (2012AA012600), and Chinese Special Project of Science and Technology (2013zx01039-002-002).

8. REFERENCES

[1] R. M. Adelsman and A. B. Whinston. Sophisticated voting with information for two voting functions. *Journal of Economic Theory*, 15(1):145–159, 1977.

[2] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, pages 324–345, 1952.

[3] R. Busa-Fekete, B. Szorenyi, W. Cheng, P. Weng, and E. Hullermeier. Top-k selection based on adaptive sampling of noisy preferences. In *ICML*, pages 1094–1102, 2013.

[4] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *WSDM*, pages 193–202, 2013.

[5] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *ICDT*, pages 225–236, 2013.

[6] A. E. Elo. *The rating of chessplayers, past and present*, volume 3. Batsford London, 1978.

[7] B. Eriksson. Learning to top-k search using pairwise comparisons. In *AISTATS*, pages 265–273, 2013.

[8] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, 1994.

[9] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, pages 385–396, 2012.

[10] R. Herbrich, T. Minka, and T. Graepel. Trueskill: A bayesian skill rating system. In *NIPS*, pages 569–576, 2006.

[11] X. Jiang, L. Lim, Y. Yao, and Y. Ye. Statistical ranking and combinatorial hodge theory. *Math. Program.*, 127(1):203–244, 2011.

[12] A. R. Khan and H. Garcia-Molina. Hybrid strategies for finding the max with the crowd. Technical report, 2014.

[13] G. Li, J. Wang, Y. Zheng, and M. Franklin. Crowdsourced data management: A survey. *IEEE Trans. Knowl. Data Eng.*, 2016.

[14] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[15] S. Negahban, S. Oh, and D. Shah. Iterative ranking from pair-wise comparisons. In *NIPS*, pages 2483–2491, 2012.

[16] T. Pfeiffer, X. A. Gao, Y. Chen, A. Mao, and D. G. Rand. Adaptive polling for information aggregation. In *AAAI*, 2012.

[17] J.-C. Pomerol and S. Barba-Romero. *Multicriterion decision in management: principles and practice*, volume 25. Springer Science & Business Media, 2012.

[18] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *WWW*, pages 989–998, 2012.

[19] F. L. Wauthier, M. I. Jordan, and N. Jojic. Efficient ranking from pairwise comparisons. In *ICML*, pages 109–117, 2013.

[20] P. Ye and D. Doermann. Combining preference and absolute judgements in a crowd-sourced setting. In *Proc. of Intl. Conf. on Machine Learning*, pages 1–7, 2013.