Efficient Selection of Pairwise Comparisons for Computing Top-heavy Rankings

Shenshen Liang
Technology and Information Management
University of California, Santa Cruz
Santa Cruz, California 95136
Email: sliang@soe.ucsc.edu

Luca de Alfaro
Computer Science
University of California, Santa Cruz
Santa Cruz, California 95136
Email: luca@ucsc.edu

***Abstract*—Crowdsourcing provides an efficient way to gather information from humans for solving large scale problems. Learning to rank via pairwise comparison is one of the most essential tasks in crowdsourcing, and it is widely used in applications, such as recommendation systems, online contests, player matching, and more. While much research has been done on how to aggregate the comparison data into an overall ranking, comparatively less research has been done on how to optimally select items for pairwise comparison. In this research, we consider ranking problems where the benefit for each item to be ranked in position n is a geometrically decreasing function of n. This geometric dependence between ranking and benefit is common online and on the web. We define the quality of a ranking as the total mis-allocated benefit, so that in learning a ranking, we are more sensitive to errors in the ordering of top items than errors in items ordered in the long tail. We propose and compare several active learning methods for selecting pairs for comparison. The methods actively search for the pairs to compare, present them to the crowd, and update the ranking according to the comparison outcomes. We show experimentally that the best-performing method selects pairs on the basis of the expected benefit mis-allocation between the items in the pair. As the size of the ranking problem grows, the computational cost of selecting the optimal pair for each comparison becomes prohibitive. We propose and show an efficient algorithm that selects items in batches while retains nearly optimal performance, at a cost per comparison that grows only logarithmically with the total number of items.**

Keywords*–*Top-heavy Ranking; Pairwise Comparison; Active Learning; Crowdsourcing.

I. Introduction

Crowdsourcing systems obtain small pieces of information from ordinary crowds and have been proven effective to get human-power information with relatively low cost [1]. They have been extensively used online, such as Wikipedia, Amazon Mechanical Turk, Quora and Stack Exchange Network.

Ranking is one of the key problems in crowdsourcing and is widely used in a variety of applications. Typically, it is performed on the basis of three types of input: binary relevance label, which categorizes data into two types such as relevant/irrelevant, true/false, etc.; graded relevance label, which classifies data into multiple ordinal levels; and pairwise preference, where a label is expressed as preference between two items rather than an absolute judgment. Each type of input is a tradeoff between potential information gain and difficulty of getting effective information. For example, for data with five star graded relevance labels, while it obtains finer-grained information than binary relevance labels, users are more prone to bias and errors. Pairwise preference, on the other hand, is a simple expression of preference between two items. Such data is relatively easy to obtain, it is less prone to errors because of its simplicity, and can be expanded to graded relevance label [2]. As a result, ranking via pairwise comparisons becomes an essential task in ranking problems.

Extensive research has been done on how to aggregate pairwise comparisons into accurate rankings, such as Mallows [3], nuclear norm minimization [4], Bradley-Terry [5], Glicko [6], TrueSkill [7] and so on [8][9]. However, it is still very challenging to efficiently obtain data with minimal computational cost. Therefore, we concentrate on studying active learning strategies for selecting pairs of items to be sent to the crowd for comparison, so that rankings will converge as quickly as possible to the correct rank. Specifically, in this paper we focus on rankings where an ordinal rank k is associated with a benefit proportional to its position. These type of geometric benefit distributions are typical in online settings, where higher rank commands geometrically higher visibility, and revenue [2][10][11].

In setting the problem, we assume that each item in a ranking has an intrinsic "quality", and we define a "top-heavy" version of ranking distance where the mis-placement of items is weighed according to $1/k^{\lambda}$ for rank k with $\lambda > 0$; this will be used to judge the speed of convergence of the proposed methods. Thus, errors in the head of the ranking will be weighed more heavily than errors in the tail. As ranking aggregation is not our primary focus, we perform ranking aggregation by applying the Glicko [6] and TrueSkill [7], which are well-established on-line aggregation algorithms.

We then formulate and compare two active learning strategies for selecting the next pairs of items to compare. In particular, we define the *loss* (or error) involved in each pair of items, and we consider and justify strategies that select pairs for comparison that have maximum loss, or that lead to maximum ranking change. We identify one such strategy, *maximum loss,* as the one that leads to overall best results, as demonstrated by our simulations.

These pair selection strategies are computationally expensive, as they require at each round the computation of many alternatives in order to select the best. To address this problem, we propose and develop efficient batched versions of the pair selection strategies, which deliver essentially the same performance while exhibiting complexity that grows only logarithmically with the number of items, the dominant step being a sorting step.

Our contributions can be summarized as follows:

- We define a distance to measure "top-heavy" rankings.
- We propose two active learning strategies for selecting pairs effectively which reduce ranking loss rapidly.
- We propose an efficient batch algorithm with low computational cost.

After a review of related work in Section II, we define the problem precisely, and describe the Glicko and TrueSkill methods for ranking aggregation (Section III). In Section IV we present our active learning methods for selecting pairs, and in Section V we present and analyze the experimental results.

II. Related Work

Obtaining data from crowdsourcing is widely applied in many fields, such as advertising, ranking, knowledge sharing, elections, opinion collection, and so on [12][13]. Many applications collect boolean or grade-based feedback about individual items. For example, StackOverFlow provides a vote up and vote down mechanism and allows users stating whether a question is useful. Yelp asks users to grade merchants in a 1 to 5 star grade-based rating system.

Much research has been done on how to aggregate comparison outcomes into a ranking [14]–[19]. Generally, ranking aggregation methods can be categorized into three types: permutation-based, such as researches in Mallows [3] and CPS [15] models; matrix factorization, such as work in [4]; and score-based probabilistic method, such as Plackett-Luce [20][21], Bradley-Terry [5], Thurstone [22], etc. Permutation methods are generally computational expensive, while matrix factorization methods do not have sufficient probabilistic interpretations. As a result, we use score-based methods for ranking aggregation in this paper.

Score-based methods assign a score to each item, and use all scores to generate rankings. Among score-based methods learning from pairwise data, the Elo ranking method is perhaps the first Bayesian pairwise ranking algorithm [23], and it is widely used in ranking sports and estimating the underlying skills of players. A player's skill is assumed to follow a Gaussian distribution with two parameters as average skill level and players uncertainty. Glickman extended Bradley-Terry model and updated player skills based on designed length of period, assuming same Gaussian distribution of player skills [6]. Trueskill by Microsoft is another Bayesian ranking system with Gaussian distribution assumption [7]. It extendes a Thurstone-Mosteller model which adds a latent variable as player performance.

Ranking aggregation via pairwise comparisons aims at computing a ranking for items that can represent all the comparison outcomes with minimum data disagreement. The problem that concerns us in this paper is how to optimally select pairs for comparison, so that a "good" ranking can be obtained with as few comparisons as possible, and thus, as efficiently as possible.

Active learning is an effective way to improve efficiency and promote performance. It has been recognized that by properly selecting items, learning tasks achieve better accuracy and require less data for training [24]–[27]. There are mainly three types of active learning methods. The first one is uncertainty sampling, which targets at finding items that the system is most uncertain about [28][29]. Another one is minimizing expected loss, which focuses on searching for items that can reduce highest expected error [30]. Lastly, query by committee method looks for items that a set of learners (refers as committee) having largest disagreement with [31][32]. While extensive research has been performed on active learning, most of them are for binary or graded based problems [33]–[39].

Some research was done on active learning for ranking from pairwise comparisons. Donmez et al. applied their document selection algorithm to RankSVM and RankBoost [40]. Also using RankSVM, Yu proposed to add most ambiguous document pairs to training dataset [41]. Chen et al. proposed a framework to find reliable annotators and informative pairs jointly, which requires annotator quality information [42]. A maximum likelihood based algorithm was proposed by Guo et al. to locate the topmost item in a ranking [43]. Other research based on pairwise comparisons includes work done by Chu et al. [44], Ailon [45], Jamieson et al. [46] and so on. Notably, a majority of them focus on selecting annotators rather than items.

III. Top-Heavy Ranking

A ranking problem consists in sorting a set of items $S = \{s_1, s_2, \ldots, s_n\}$ according to their quality. Ranking problems are solved via crowdsourcing when assessing the quality of an item requires human judgement, as is the case, for instance, when assessing the quality or appeal of videos, images or merchandise. We consider here ranking problems that are *top-heavy* in their reward: being ranked in position k has value proportional to $1/k$.

These top-heavy problems find application whenever the ranking is used to display the items to users. In such cases, a higher rank commands a larger amount of user engagement, which can be measured in item views, page visits, user votes and so forth, according to the nature of the items being ranked. As user attention is valuable (and can be monetized), we assume that the *value* of being ranked in position k is proportional to $1/k^\lambda$, for some $\lambda > 0$. We call this a λ-top heavy ranking. This is equivalent to assuming that user attention follows a Zipf distribution, an assumption that has been validated on the Web [47].

A. Ranking Quality

To measure the quality of a ranking, we introduce a measure of *distance* between top-heavy rankings. Our distance will give more weight to differences among top positions than to differences among positions in the tail of rankings. This reflects the intuition that errors in top rankings matter more than errors in the tails of rankings, as there is much more value in the top than in the tail. For instance, in a sport competition where athlete sponsorship is proportional to the inverse of the rank, it would obviously be worse to get the order wrong between the first and second positions than between the 101st and the 102nd.

Precisely, for our set of items $S = \{s_1, s_2, \ldots, s_n\}$, consider two rankings r and r' so that $r(i)$ is the position (the ordinal) of item s_i according to ranking r, for $1 \leq i \leq n$, and similarly for r'. We define the distance $d(r, r')$ between r

and r' by:

$$d(r, r') = \sum_{i=1}^{n} \left| \frac{1}{r(i)^\lambda} - \frac{1}{r'(i)^\lambda} \right| , \qquad (1)$$

where λ is the coefficient of the λ-top heavy ranking. Equation (1) can be understood as follows. If r is the correct ranking, and r' is another ranking, then $|\frac{1}{r(i)^\lambda} - \frac{1}{r'(i)^\lambda}|$ is the amount of value that item i receives in error, either in positive or negative. Thus, the quantity (1) represents the total value mis-allocation of ranking r', measured with respect to ranking r.

In particular, if r^* is the correct ranking, we denote by

$$\mathcal{L}(r) = d(r, r^*) \qquad (2)$$

the *loss* of r, measured as its distance from optimality.

B. Learning Top-Heavy Rankings

Our goal consists in developing algorithms for learning top-heavy rankings via crowdsourcing. The algorithms we develop follow the following scheme:

1) We start with a random ranking.
2) At each round:
    a) We select two items, and we ask a user to compare them.
    b) We use the result of the comparison to update the rankings.

We rely on binary comparisons because they are the most elementary of comparisons, and they require less cognitive load on the user than multi-way comparisons. The goal of the above process is to converge as quickly as possible to the optimal ranking according to distance (1), that is, to reduce the loss of the rank as quickly as possible. As our distance is top-weighed, this means identifying the top items early.

In this paper, we focus on step 2a: the selection of the items to be compared. Once two items are compared, there are several classical methods for updating a ranking according to the comparison outcome; we describe two such alternative methods, Glicko and Trueskill, in the following. Our focus here is on how to choose the elements whose comparison will reduce ranking loss in the fastest possible way, and with low computational cost. Intuitively, choosing the elements to compare entails estimating which elements might be incorrectly ranked, keeping into account that errors at the top matter more than errors in the tail of the ranking. As the choice of the pairs to be compared uses information from the ranking update step, we first describe the ranking update step, and subsequently our proposed methods for item selection.

C. Ranking Update Methods

We describe here two ranking update methods: Glicko [6], and TrueSkill [7].

1) Glicko: The Glicko [6] method for ranking update models each item as having a score that has a Gaussian distribution. Thus, for each item s_i, Glicko stores the median μ_i and the standard deviation σ_i of the score. The model further assumes that if two items s_i, s_j have scores X_i and X_j (sampled from their respective distributions), then the probability that a user prefers s_i to s_j is proportional to

$$\frac{e^{\kappa(X_i - X_j)}}{1 + e^{\kappa(X_i - X_j)}} ,$$

where $\kappa > 0$ is an arbitrary scaling constant. This is known as the Bradley-Terry model of match outcomes [5]. In [6], the constant is $\kappa = (\log 10)/400$, and was chosen to scale the resulting scores so that they would approximate the scores of the Elo ranking for chess players [23]. The value of the constant is immaterial to the ranking being produced (it is simply a scaling for the scores), and we choose $\kappa = 1$ in our implementation.

With these choices, once a comparison is done, the Glicko model parameters are updated as follows. Denote with s_{ij} the outcome of comparison between item s_i and item s_j:

$$s_{ij} = \begin{cases} 1 & \text{if } i \text{ wins } j \\ 0 & \text{if } j \text{ wins } i \end{cases}$$

Let also for $i = 1, 2$,

$$g(\sigma_i^2) = \frac{1}{\sqrt{1 + 3q^2\sigma^2/\pi^2}}$$

The update formulas for the mean and standard deviations are:

$$\mu_i' = \mu_i + \frac{q}{\frac{1}{\sigma_i^2} + \frac{1}{\delta_i^2}} g(\sigma_j^2)(s_{ij} - z_j)$$

$$\sigma_i'^2 = \left(\frac{1}{\sigma_i^2} + \frac{1}{\delta_i^2} \right)^{-1}$$

where

$$z_j = \frac{1}{1 + e^{-g(\sigma_j^2)(\mu - \mu_j)}}$$

$$\delta_i^2 = \left[q^2 \left(g(\sigma_j^2) \right)^2 z_j (1 - z_j) \right]^{-1} .$$

The above update formulas are obtained from [6] as the special case in which time decay of the scores does not occur. Glicko models time decay of scores, so that as players remain inactive, their score median decreases, and their score standard deviation increases, modeling increased uncertainty about their abilities. Such time dependence is appropriate in modeling tennis and chess scores, but not in modeling the quality of items in crowdsourcing batches of short temporal duration.

2) TrueSkill: The TrueSkill rating system [7] also assumes a Gaussian belief on online game players' skills. It estimates skills by constructing a factor graph, connecting players that have had a match together, and using approximate message passing. Trueskill was developed for players belonging to teams; we present here a simplified version without teams (or, more precisely, where all teams have one player only), which corresponds to the problem at hand.

The skill of player s_i, denoted as X_i, is again assumed to be Gaussian-distributed with mean μ_i and standard deviation σ_i. The performance of player i, denoted by Y_i, is also assumed to be Gaussian distributed, with mean value equal to X_i and standard deviation β, with β constant for all players. Thus, X_i models the intrinsic quality of item s_i, whereas Y_i models the actual performance of player s_i in a specific match. Translated into our setting, Y_i models how the quality of an item is perceived by the worker performing the comparison. A tie between s_i and s_j occurs when $|Y_i - Y_j| \leq \varepsilon$ for a chosen ε, while s_i is preferred to s_j when $Y_i - Y_j > \varepsilon$.

If denote $\boldsymbol{Z}$ as whole set of game outcomes, the skills of all players as $\boldsymbol{X}$, the performances of all players as $\boldsymbol{Y}$, and all differences between the performances of two players as $\boldsymbol{D}$, the general Bayesian inference problem is:

$$\begin{aligned} p(\boldsymbol{X}|\boldsymbol{Z}) &\propto \iint p(\boldsymbol{Z}, \boldsymbol{D}, \boldsymbol{Y}, \boldsymbol{X})\, d\boldsymbol{D}\, d\boldsymbol{Y} \\ &= \iint p(\boldsymbol{Z}|\boldsymbol{D}) p(\boldsymbol{D}|\boldsymbol{Y}) p(\boldsymbol{Y}|\boldsymbol{X}) p(X)\, d\boldsymbol{D}\, d\boldsymbol{Y} \end{aligned}$$

where the joint density of the entire system is presented as a product of distributions. So, the problem consists in computing a marginal probability. To solve it, Trueskill implements an approximate message passing method between nodes that correspond to the parameters of the problem. The message passing iterations stop when convergence is reached, yielding the player's skills, which correspond for us to the item qualities.

We apply TrueSkill after each comparison, updating the scores of the two items that took part in the comparison itself.

IV. Active Learning for Pair Selection

Our active learning strategies aim at selecting the pair whose comparison will reduce ranking loss most effectively. In designing pair selection strategies, we focus on strategies that select items which might be incorrectly ranked, as comparing such items is likely to be more beneficial. We propose two strategies: maximum loss and maximum ranking changes.

A. Maximum Loss

The maximum loss ranking strategy selects for comparison at each step the pair of items that have the largest expected mis-allocation of reward. To make this expected value mis-allocation precise, we define the expected loss of a pair of items as the product of the probability of the two items having incorrect relative orders by the amount of error resulting from this situation. If we denote the item with higher rank as s_i and the one with lower rank as s_j, the probability of a pair having incorrect relative orders is essentially the probability of item s_j having a larger sampled value than that of item s_i from their distributions respectively. The amount of value mis-allocation resulting from incorrect relative orders in a top-heavy ranking is $\left|\frac{1}{r(i)} - \frac{1}{r(j)}\right|$. So, our strategy of selecting maximum expected loss selects the items i, j given by:

$$\arg\max_{(i,j)} \left\{ Prob.(\tilde{s}_i < \tilde{s}_j) * \left| \frac{1}{r(i)} - \frac{1}{r(j)} \right| \right\} \tag{3}$$

where $\tilde{s}_i, \tilde{s}_j$ are sampled values from distributions of item s_i, s_j, $Prob.(\tilde{s}_i < \tilde{s}_j)$ is the probability of s_i, s_j having incorrect relative orders, and $r(i)$, $r(j)$ are the ranking positions of the items. By the properties of Gaussian distributions, the probability of $r(i)$, $r(j)$ having incorrect relative orders can be calculated as:

$$Prob.(\tilde{s}_i < \tilde{s}_j) = Prob.(\tilde{s}_i - \tilde{s}_j < 0) = \Phi\left(\frac{-|\mu_i - \mu_j|}{\sqrt{\sigma_i^2 + \sigma_j^2}} \right)$$

where μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of the distributions of s_i, s_i respectively.

B. Maximum Ranking Change

The maximum ranking change strategy selects the items whose comparison is going to have the greatest impact on the current ranking. Intuitively, if two items with incorrect relative orders change their rankings after a comparison, it implies that a big problem exists potentially and the previous ranking is unstable or unreliable. In consideration of this implication, we propose a strategy selecting pairs that will get the largest expected ranking change after comparison, for items having incorrect relative rankings.

The expected ranking change for items having incorrect relative order is the product of the probability of two items having incorrect relative orders and the expected amount of change to rankings after the pair comparison. With the same notations as first strategy, and denoting the expected amount of change for items having incorrect relative order as $g(s_i \prec s_j)$, ideally we would like to select a pair of items i, j as follows:

$$\arg\max_{(i,j)} \{ Prob.(\tilde{s}_i < \tilde{s}_j) * g(s_i \prec s_j) \} \tag{4}$$

where $Prob.(\tilde{s}_i < \tilde{s}_j)$ is computed as before. The expected amount of change for items having incorrect relative order can be calculated by:

$$g(s_i \prec s_j) = \left| \frac{1}{r(i)} - \frac{1}{r(i)^{s_i \prec s_j}} \right| + \left| \frac{1}{r(j)} - \frac{1}{r(j)^{s_i \prec s_j}} \right|$$

where $r(i)^{s_i \prec s_j}$ and $r(j)^{s_i \prec s_j}$ are the updated rankings of item s_i and s_i, if item s_j wins s_i in comparison.

The problem with the selection (4) is that it requires computing the outcome of all possible pair comparisons. This is very expensive computationally: in order to get the future ranking positions of the items in a pair, the algorithm has to perform quality updates for both items, and sort all items for a new ranking. To address this problem, we analyze the equation and propose an approximate version.

Equation (4) can also be expressed as:

$$\begin{aligned} &Prob.(\tilde{s}_i < \tilde{s}_j) * g(s_i \prec s_j) \\ &= Prob.(\tilde{s}_i < \tilde{s}_j) * \left(\frac{1}{r(i)} - \frac{1}{r(i)^{s_i \prec s_j}} - \frac{1}{r(j)} + \frac{1}{r(j)^{s_i \prec s_j}} \right) \end{aligned} \tag{5}$$

Equation (5) holds because with a comparison of s_j winning s_i, the ranking update algorithms will update the ranking so that $r(j)^{s_i \prec s_j}$ ranks higher than or equivalent to $r(j)$, while $r(i)^{s_i \prec s_j}$ ranks lower than or equivalent to $r(i)$. This result is consistent with the intuition that one item shall get a lower ranking if loses, while the other shall rank higher if wins.

Assuming the ranking changes for both items are in same scale of $\alpha > 0$, i.e.,

$$\begin{cases} \frac{1}{r(i)^{s_i \prec s_j}} = \frac{1}{r(i)} * (1 + \alpha) \\ \frac{1}{r(j)^{s_i \prec s_j}} = \frac{1}{r(j)} * (1 - \alpha) \end{cases}$$

then (5) can be further expressed as:

$$\begin{aligned} &Prob.(\tilde{s}_i < \tilde{s}_j) * \left(\frac{1}{r(i)} - \frac{1}{r(i)^{s_i \prec s_j}} - \frac{1}{r(j)} + \frac{1}{r(j)^{s_i \prec s_j}} \right) \\ &\quad = Prob.(\tilde{s}_i < \tilde{s}_j) * \left(\frac{1}{r(i)} \cdot \frac{\alpha}{(1+\alpha)} + \frac{1}{r(j)} \cdot \frac{\alpha}{(1-\alpha)} \right) \end{aligned} \tag{6}$$

By first order Tyler Series, while $\alpha \to 0$, $\frac{\alpha}{(1+\alpha)} \to \alpha$, $\frac{\alpha}{(1-\alpha)} \to \alpha$. Assuming $\alpha \to 0$, (6) can be approximated by:

$$Prob.(s_i \prec s_j) * \left(\frac{1}{r(i)} + \frac{1}{r(j)}\right) \cdot \alpha \tag{7}$$
$$\propto Prob.(s_i \prec s_j) * \left(\frac{1}{r(i)} + \frac{1}{r(j)}\right)$$

We assume α approaches 0 because we believe that in a ranking system with sufficient comparisons, one single comparison shall not change any item's ranking dramatically. As an example, a tennis player will not get a huge ranking drop just because he loses one game.

In light of the above approximations, the maximum ranking change strategy selects the items i, j as follows:

$$\arg\max_{(i,j)} \left\{ Prob.(s_i \prec s_j) * \left(\frac{1}{r(i)} + \frac{1}{r(j)}\right) \right\} . \tag{8}$$

Thus, the maximum ranking change pair selection strategy only relies on the current positions of items, and the computational complexity is significantly reduced.

C. Stochastic Pair Drawing

Deterministically selecting for comparison the pair with the highest value, as done in (3) and (8), carries the risk of trapping the ranking in a local optimum. To deal with this problem, we propose to use a *randomized* version of our pair selection strategies. In the randomized version, we select each pair with probability proportional to the arguments of (3) and (8), respectively. The algorithms thus still focus on the most promising pairs for comparison, but their randomized nature makes them more robust. In experiments we performed, the randomized version of the selection algorithms always outperformed the deterministic ones, so that for this paper we opted for presenting the results only for the stochastic versions, for the sake of conciseness.

D. Batch Algorithm for Efficient Selection

In many online applications, the volume of items in a ranking is huge. With such size, it is very computational expensive, or even impossible to evaluate all candidate pairs. Also, updating the ranking after every user comparison sequentially is impractical and will impose vast burden on the system. As a result, in practice, instead of sequential algorithms, batch active learning methods are widely used. Therefore, we design a batch algorithm to reduce the computational cost and make our algorithms more efficient.

In the above pair selection algorithms, we observe that items close in rank are more likely to be selected, because they have a higher probability of being incorrectly ranked. We make use of this observation in our batch algorithm to narrow the candidate pair space. For any item s_i with position $r(i)$, when selecting its candidate pairing set, we do not consider all other items: rather, the batch algorithm only looks for a subset of items immediately below s_i and evaluates the corresponding pairs. In this way, we are able to cut down evaluation pairs dramatically. Since the ranking can be incorrect, we also include in the evaluation pairs items that are sampled randomly. This randomness improves the stability of the algorithm.

Once the batch algorithm determines candidate pairing sets for all items, it uses the same active strategies to calculate values for all pairs. Then, it selects a batch of pairs stochastically, where each pair is selected with probability proportional to its value. All selected pairs are presented to users concurrently for their comparisons. After the comparison outcomes are collected, the batch algorithm updates the ranking with all outcomes at once and a new batch will be selected. In this way, the expensive process of computing pair evaluations is performed only once for every batch, rather than once for every pair presented to users.

V. Experiments

A. Experiment Settings

We conduct experiments with simulated data as it can provide precise "true" ranking for evaluation. We assume there are 100 items, each of them has an underlying score with a Gaussian distribution, and sample its mean and variance respectively. Specifically, we sample all items' means from one Gaussian distribution as opposed to any heavy-tailed distribution, because in this paper, we focus on the intrinsic "qualities" of items that represent their strength or greatness, such as competitiveness of sports players, skills of online gamers, ratings of merchandise, reviews of restaurants; and a ranking is generated based on items' qualities. In contrast, a heavy-tailed distribution is usually assumed in relevance ranking problems, which sort items by their degree of relevance.

All pair selection strategies start with the same non-informative score estimation. Every time an algorithm requests a user comparison, we simulate it by sampling from true distributions of both items. The item with a larger sampled value is considered the winner. We conduct 20 experiments for each strategy, with 2,000 pair selections per experiment. All algorithms are evaluated with respect to the loss (2). In the experiments, we choose λ values at $0.5, 1, 2$ respectively.

B. Algorithm Loss Comparison

Figure 1 shows the loss decrease resulting from the proposed algorithms, for different λ coefficients. Along the x-axis is the number of selected pairs and along the y-axis is the average loss per experiment. The error bars indicate 95% confidence intervals, and the random strategy is used as baseline. After each pairwise comparison, the item scores are updated via Glicko and TrueSkill respectively, and a new pair to compare is selected.

The results demonstrate that our two proposed algorithms perform significantly better than the baseline. With more pairwise comparisons, it is expected to see the ranking losses of all the algorithm decreasing, but the proposed algorithms converge to the true ranking and reduce loss much faster. The 95% confidence intervals between baseline and proposed algorithms rarely overlap, illustrating that the reduction in loss resulting from our algorithms is significant. Maximum loss and maximum ranking change algorithms get comparable results at the end of experiments, with maximum loss performing slightly better. We believe it is because the approximation used in maximum ranking change introduces some loss.

Relatively, the loss decrease is slower for algorithms evaluated with a λ coefficient of 2. We believe that it is due to the top-heavy reduction effect. With $\lambda > 1$, except for the very top of the ranking, the losses from mis-ranked elements become smaller, limiting the scope for loss decrease once the top-ranked items are correctly ranked. The divergence between algorithms and evaluation measure results in the performance decrease.

C. Sequential and Batch Version Comparison

We have also experimented with the batch version of our algoritms. We start with the same non-informative score estimation. We use 10% of total items as candidate pairing set for each item, with half of them sampled from the immediately lower ranked elements, and half of them sampled randomly. The batch size is set at 20, so the ranking will be updated after collecting 20 pairwise comparisons from users.

Sharing the same x-axis and y-axis as Figure 1, Figure 2 shows loss decrease of sequential and batch versions of each algorithm. It is evident that in spite of evaluating only 10% of all pairs, the batch versions achieve almost same performances as sequential versions. The results demonstrate that the batch versions of our algorithms are capable of getting comparable performance as the sequential versions, while dramatically reducing the computational cost. Figure 3 illustrates that the average run time per experiment of the batch versions are significantly smaller than that of the sequential versions. Collectively, Figures 2 and 3 prove that our batch algorithms are an effective solution to ranking problems with large dataset.

VI. Conclusion and Future Work

In this paper, we propose two active learning strategies for selecting pairwise comparisons for top-heaving rankings, where top ranking items are considered more critical than items lower in the rankings. To address the computational challenge arising from large data volume, an efficient batch algorithm is proposed and applied. Our experimental results show that both active learning methods are effective at reducing ranking loss; overall, the maximum loss method achieves slightly better performance. We also demonstrate how our batch algorithm can achieve comparable loss decrease results while significantly reducing computational costs.

We see several directions for future work. It is interesting to explore other approximation methods for maximum ranking change strategy. Furthermore, when a ranking only aims at selecting the top-k items, without caring about their relative order, different active learning strategies may yield superior results. Finally, other pairwise comparison aggregation methods

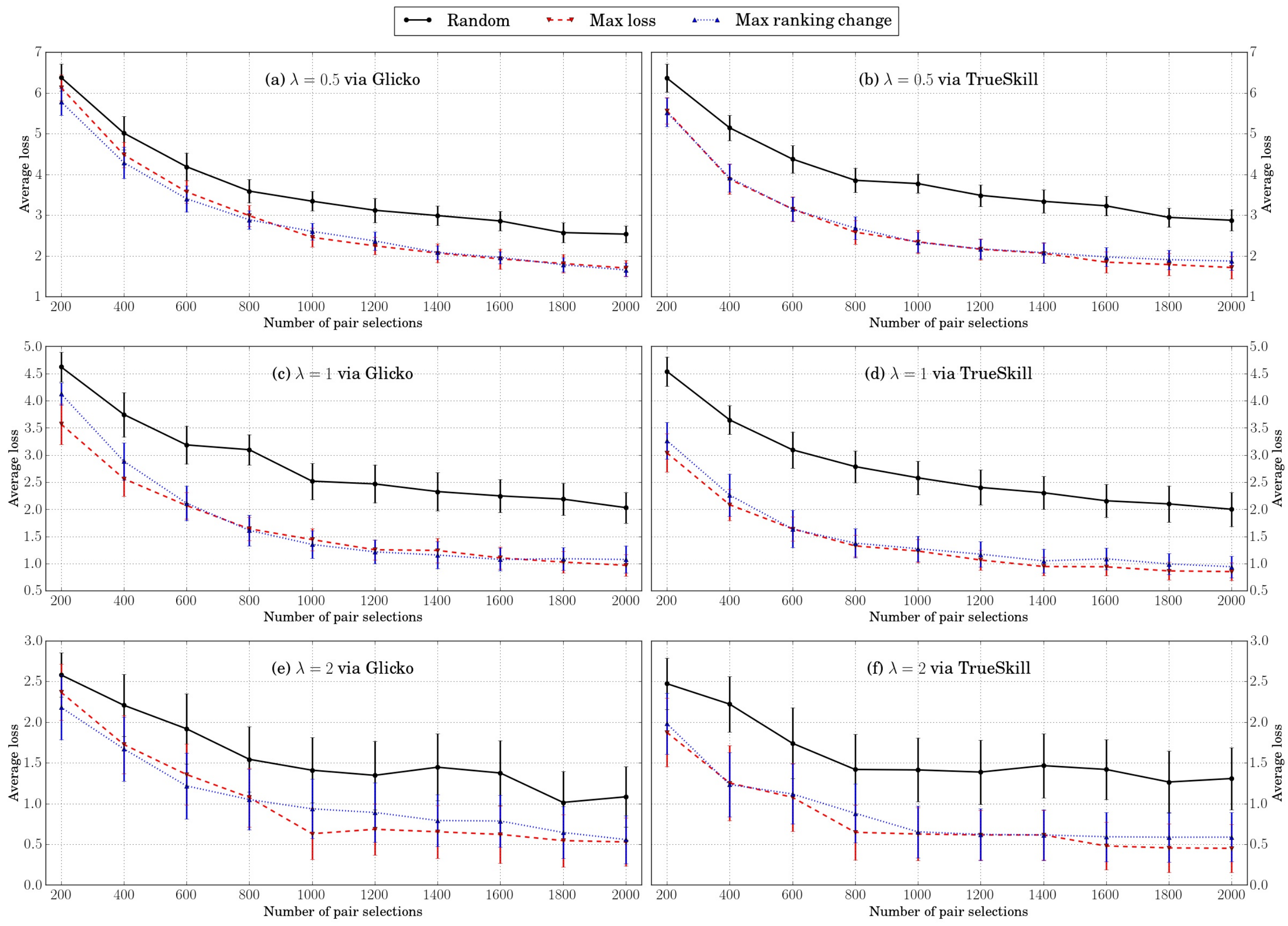

Figure 1. Loss comparison: active learning vs. random strategy.

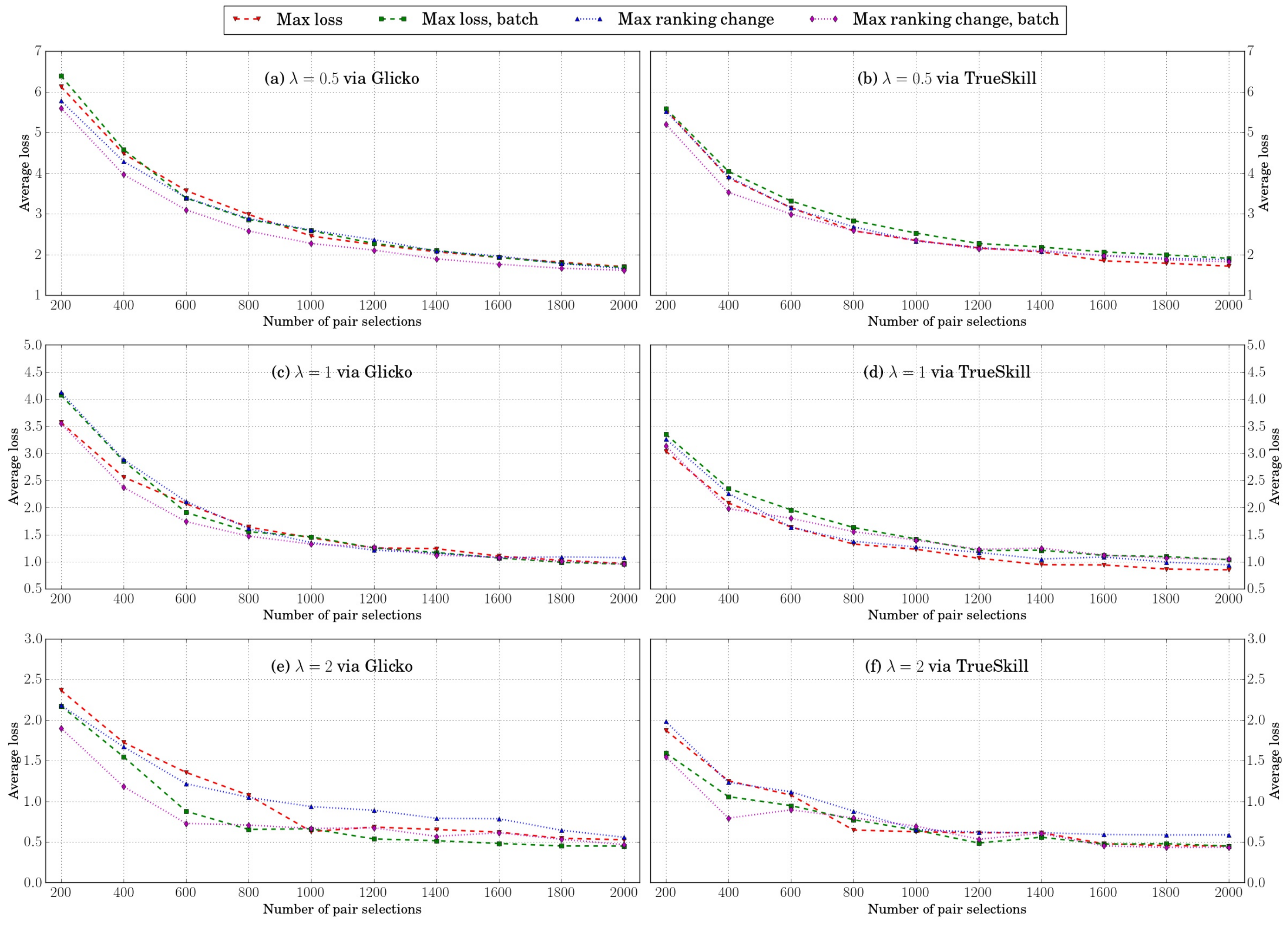

Figure 2. Loss comparison: sequential vs. batch.
The loss differences between sequential and batch versions are not significant.

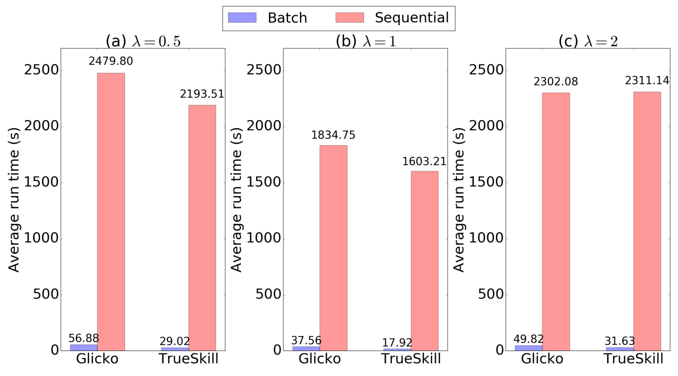

Figure 3. Average run time comparison: sequential vs. batch.

can be explored for better accuracy.

REFERENCES

[1] J. Wang, P. G. Ipeirotis, and F. Provost, “Managing crowdsourcing workers,” in The 2011 winter conference on business intelligence, 2011, pp. 10–12.

[2] F. Radlinski, M. Kurup, and T. Joachims, “How does clickthrough data reflect retrieval quality?” in Proceedings of the 17th ACM Conference on Information and Knowledge Management, ser. CIKM ’08. New York, NY, USA: ACM, 2008, pp. 43–52.

[3] C. L. Mallows, “Non-Null Ranking Models. I,” Biometrika, vol. 44, no. 1/2, 1957, pp. 114–130.

[4] D. F. Gleich and L.-h. Lim, “Rank aggregation via nuclear norm minimization,” in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011, pp. 60–68.

[5] R. A. Bradley and M. E. Terry, “Rank analysis of Incomplete Block Designs: I. The Method of Paired Comparisons,” Biometrika, vol. 39, no. 3/4, 1952, pp. 324–345.

[6] M. E. Glickman, “Parameter Estimation in Large Dynamic Paired Comparison Experiments,” Journal of the Royal Statistical Society: Series C (Applied Statistics), vol. 48, no. 3, Jan. 1999, pp. 377–394.

[7] R. Herbrich, T. Minka, and T. Graepel, “TrueSkill : A Bayesian Skill Rating System,” in Advances in Neural Information Processing Systems 19, B. Schlkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 569–576.

[8] D. R. Karger, S. Oh, and D. Shah, “Iterative learning for reliable crowdsourcing systems,” in Advances in neural information processing systems, 2011, pp. 1953–1961.

[9] C. Burges et al., “Learning to rank using gradient descent,” in Proceedings of the 22nd international conference on Machine learning. ACM, 2005, pp. 89–96.

[10] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, “Expected Reciprocal Rank for Graded Relevance,” in Proceedings of the 18th ACM Conference on Information and Knowledge Management, ser. CIKM ’09. New York, NY, USA: ACM, 2009, pp. 621–630.

[11] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, “Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods,” in Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR ’09. New York, NY, USA: ACM, 2009, pp. 758–759.

[12] E. Kamar, S. Hacker, and E. Horvitz, “Combining human and machine intelligence in large-scale crowdsourcing,” in Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 467–474.

[13] K. Crammer, M. Kearns, and J. Wortman, “Learning from multiple sources,” The Journal of Machine Learning Research, vol. 9, 2008, pp. 1757–1774.

[14] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, “An efficient boosting algorithm for combining preferences,” The Journal of machine learning research, vol. 4, 2003, pp. 933–969.

[15] T. Qin, X. Geng, and T.-Y. Liu, “A new probabilistic model for rank aggregation,” in Advances in neural information processing systems, 2010, pp. 1948–1956.

[16] T.-K. Huang, R. C. Weng, and C.-J. Lin, “Generalized bradley-terry models and multi-class probability estimates,” The Journal of Machine Learning Research, vol. 7, 2006, pp. 85–115.

[17] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi, “Inferring Ground Truth from Subjective Labelling of Venus Images,” in Advances in Neural Information Processing Systems 7, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: The MIT Press, 1995, pp. 1085–1092.

[18] F. Wauthier, M. Jordan, and N. Jojic, “Efficient Ranking from Pairwise Comparisons,” in PMLR, Feb. 2013, pp. 109–117.

[19] S. Negahban, S. Oh, and D. Shah, “Iterative Ranking from Pair-wise Comparisons,” in Proceedings of the 25th International Conference on Neural Information Processing Systems, ser. NIPS’12. USA: Curran Associates Inc., 2012, pp. 2474–2482.

[20] R. D. Luce, Individual choice behavior: A theoretical analysis. Courier Corporation, 2005.

[21] R. L. Plackett, “The analysis of permutations,” Applied Statistics, 1975, pp. 193–202.

[22] L. L. Thurstone, “The method of paired comparisons for social values.” The Journal of Abnormal and Social Psychology, vol. 21, no. 4, 1927, p. 384.

[23] A. E. Elo, The rating of chess players, past and present. Arco Pub., 1978.

[24] P. Donmez and J. G. Carbonell, “Active sampling for rank learning via optimizing the area under the ROC curve,” in Advances in Information Retrieval. Springer, 2009, pp. 78–89.

[25] F. Radlinski and T. Joachims, “Active exploration for learning rankings from clickthrough data,” in Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007, pp. 570–579.

[26] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, “Active learning with statistical models,” Journal of artificial intelligence research, 1996.

[27] B. Long et al., “Active learning for ranking through expected loss optimization,” in Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. ACM, 2010, pp. 267–274.

[28] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” The Journal of Machine Learning Research, vol. 2, 2002, pp. 45–66.

[29] D. J. C. MacKay, “Information-Based Objective Functions for Active Data Selection,” Neural Computation, vol. 4, no. 4, Jul. 1992, pp. 590–604.

[30] N. Roy and A. McCallum, “Toward optimal active learning through monte carlo estimation of error reduction,” ICML, Williamstown, 2001, pp. 441–448.

[31] P. Melville and R. J. Mooney, “Diverse ensembles for active learning,” in Proceedings of the twenty-first international conference on Machine learning. ACM, 2004, p. 74.

[32] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, “Selective sampling using the query by committee algorithm,” Machine learning, vol. 28, no. 2-3, 1997, pp. 133–168.

[33] M. Fang, J. Yin, and D. Tao, “Active learning for crowdsourcing using knowledge transfer.” in AAAI, 2014, pp. 1809–1815.

[34] Y. Yan, G. M. Fung, R. Rosales, and J. G. Dy, “Active learning from crowds,” in Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 1161–1168.

[35] F. Laws, C. Scheible, and H. Schtze, “Active learning with amazon mechanical turk,” in Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, 2011, pp. 1546–1556.

[36] J. Costa, C. Silva, M. Antunes, and B. Ribeiro, “On using crowdsourcing and active learning to improve classification performance,” in Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on. IEEE, 2011, pp. 469–474.

[37] P. Donmez and J. G. Carbonell, “Proactive learning: cost-sensitive active learning with multiple imperfect oracles,” in Proceedings of the 17th ACM conference on Information and knowledge management. ACM, 2008, pp. 619–628.

[38] U. Paquet et al., “Vuvuzelas & Active Learning for Online Classification,” in NIPS Workshop on Comp. Social Science and the Wisdom of Crowds, 2010.

[39] C. Campbell, N. Cristianini, A. Smola, and others, “Query learning with large margin classifiers,” in ICML, 2000, pp. 111–118.

[40] P. Donmez and J. G. Carbonell, “Optimizing Estimated Loss Reduction for Active Sampling in Rank Learning,” in Proceedings of the 25th International Conference on Machine Learning, ser. ICML ’08. New York, NY, USA: ACM, 2008, pp. 248–255.

[41] H. Yu, “SVM selective sampling for ranking with application to data retrieval,” in Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, 2005, pp. 354–363.

[42] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz, “Pairwise ranking aggregation in a crowdsourced setting,” in Proceedings of the sixth ACM international conference on Web search and data mining. ACM, 2013, pp. 193–202.

[43] S. Guo, A. Parameswaran, and H. Garcia-Molina, “So Who Won?: Dynamic Max Discovery with the Crowd,” in Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’12. New York, NY, USA: ACM, 2012, pp. 385–396.

[44] W. Chu and Z. Ghahramani, “Extensions of gaussian processes for ranking: semisupervised and active learning,” in Proceedings of the NIPS 2005 Workshop on Learning to Rank. MIT, 2005, pp. 29–34.

[45] N. Ailon, “An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity,” J. Mach. Learn. Res., vol. 13, no. 1, Jan. 2012, pp. 137–164.

[46] K. G. Jamieson and R. D. Nowak, “Active Ranking Using Pairwise Comparisons,” in Proceedings of the 24th International Conference on Neural Information Processing Systems, ser. NIPS’11. USA: Curran Associates Inc., 2011, pp. 2240–2248.

[47] B. A. Huberman, P. L. T. Pirolli, J. E. Pitkow, and R. M. Lukose, “Strong Regularities in World Wide Web Surfing,” Science, vol. 280, no. 5360, Apr. 1998, pp. 95–97.