Learning to Top-K Search using Pairwise Comparisons

Brian Eriksson
Technicolor Research

Abstract

Given a collection of N items with some unknown underlying ranking, we examine how to use pairwise comparisons to determine the top ranked items in the set. Resolving the top items from pairwise comparisons has application in diverse fields ranging from recommender systems to image-based search to protein structure analysis. In this paper we introduce techniques to resolve the top ranked items using significantly fewer than all the possible pairwise comparisons using both random and adaptive sampling methodologies. Using randomly-chosen comparisons, a graph-based technique is shown to efficiently resolve the top $O(\log N)$ items when there are no comparison errors. In terms of adaptively-chosen comparisons, we show how the top $O(\log N)$ items can be found, even in the presence of corrupted observations, using a voting methodology that only requires $O\left(N \log^2 N\right)$ pairwise comparisons.

1 Introduction

Consider the *learning to rank* problem, where a set of N items, $\mathbf{X} = \{1, 2, ..., N\}$, has unknown underlying ranking defined by the mapping $\pi : \{1, 2, ..., N\} \rightarrow \{1, 2, ..., N\}$, such that item i is ranked higher than item j (*i.e.,* $i \prec j$) if $\pi_i < \pi_j$. Instead of resolving the entire item ranking, our goal is to return the top-k ranked items, the set $\{x \in \{1, 2, ..., N\} : \pi_x \leq k\}$. Possible applications range from determining the top papers submitted to a conference, to the recommender systems problem of finding the best items to present to a user based on prior preferences.

A critical problem is to determine a sequence of queries to efficiently resolve the top ranked items. We focus on finding the top-k items using pairwise comparisons. For two items, i, j, a pairwise comparison asks the following question, *"Is item i ranked higher than item j?"*, which only returns if $\pi_i < \pi_j$ or if $\pi_j < \pi_i$. Unfortunately, when considering pairwise comparisons, the exhaustive set of all $O\left(N^2\right)$ comparisons is often prohibitively expensive to obtain. For example, in the case of comparing protein structures, each pairwise structure comparison requires significant computation time [1]. In the recommender systems context, there are significant limitations in terms of user engagement, where each user resolves only a small number of pairwise queries [2]. In this paper, we focus on estimating a specified number of top ranked items using significantly fewer than all the pairwise comparisons.

We approach the problem of estimating the top-k items using two distinct methodologies. The first methodology exploits a constant fraction of the pairwise comparisons observed at-random in concert with a graph-based methodology to find items ranked in the top $O(\log N)$. The second technique uses a two-stage voting methodology to adaptively sample pairwise comparisons to discover the top $O(\log N)$ items using only $O\left(N \log^2 N\right)$ pairwise comparisons. We show how this adaptive technique is robust to a significant number of incorrect pairwise comparison queries with respect to the underlying ranking.

1.1 Related Work

There is a rich collection of prior literature on learning to rank a set of items (*e.g.,* [3, 4, 5, 6, 7, 8, 9], to cite only a few). This paper is motivated by recent work on estimating item ranking using pairwise comparisons. Examples include the work by Jamieson and Nowak [10] and Karbasi et. al. [11] who both focus on resolving ranking when the items lie in a low-dimensional Euclidean space. Also, the analysis by Ailon [12], where query complexity bounds are derived for resolving an ϵ-approximation of the entire ranking. Finally, recent work by Ammar and Shah [13] has shown how the top ranked items from pairwise comparisons can be resolved using a maximum entropy dis-

Appearing in Proceedings of the 16^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2013, Scottsdale, AZ, USA. Volume 31 of JMLR: W&CP 31.

tribution technique. In contrast to prior work, analysis presented in this paper focuses on deriving worst-case bounds for finding the top-ranked items exactly with high probability, while making no assumptions as to the underlying embedding or distribution of the items.

1.2 Paper Organization

The paper is organized as follows. In Section 2, we review the notation for the top-k search problem. An algorithm for at-random observations of pairwise comparisons is introduced in Section 3. In Section 4, we describe a robust methodology for adaptive sampling of pairwise comparisons. Finally, experimental results are shown in Section 5 and we state future directions in Section 6.

2 Top-K Search and Notation

Let $\mathbf{X} = \{1, 2, ..., N\}$ be a collection of N items with underlying ranking defined by the mapping $\pi : \{1, 2, ..., N\} \rightarrow \{1, 2, ..., N\}$, such that item $\{x \in \{1, 2, ..., N\} : \pi_x = 1\}$ is the top-ranked item (*i.e.,* the most preferred), and item $\{x \in \{1, 2, ..., N\} : \pi_x = N\}$ is the bottom-ranked item (*i.e.,* the least preferred). We assume no ties.

To describe subsets of items in the underlying ranking we use the following terminology,

Definition 1. *The item subset* $\{x \in \{1, 2, ..., N\} : \pi_x \leq k_1\}$ *are the* ***top-***k_1 ***items.***

Definition 2. *The item subset* $\{x \in \{1, 2, ..., N\} : \pi_x > N - k_2\}$ *are the* ***bottom-***k_2 ***items.***

Definition 3. *The item subset* $\{x \in \{1, 2, ..., N\} : k_A < \pi_x \leq k_B\}$ *are the* ***middle-***$\{k_A, k_B\}$ ***items.***

Our goal is to return the top-k items, for some specified $k > 0$. Unfortunately, the given item set $\mathbf{X} = \{1, 2, ..., N\}$ is unordered. To determine the collection of top ranked items, we query pairwise comparisons.

Definition 4. *We define the* ***pairwise comparison matrix,*** $\mathbf{C}$*, where,*

$$c_{i,j} = \begin{cases} 1 & \pi_i < \pi_j \\ 0 & otherwise \end{cases} \tag{1}$$

As stated in the introduction, in many applications not all $O\left(N^2\right)$ pairwise comparisons (*i.e.,* the entire matrix, $\mathbf{C}$) will be available. To denote this incompleteness, we define the indicator matrix of similarity observations, Ω, such that $\Omega_{i,j} = 1$ if the pairwise comparison $c_{i,j}$ has been observed and $\Omega_{i,j} = 0$ if the pairwise comparison $c_{i,j}$ is not observed (*i.e.,* the pairwise comparison is unknown).

In Section 4.1, we consider the case where these comparison queries can be returned with incorrect information that does not conform to the underlying ranking. We model these errors as i.i.d. with probability bounded by $q \geq 0$, such that,

$$P\left(c_{i,j} \neq \mathbf{1}\left(\pi_i < \pi_j\right)\right) \leq q \tag{2}$$

Where the indicator function, $\mathbf{1}\left(E\right) = 1$ if the event E occurs, and equals zero otherwise.

3 Top-K Search Using Randomly-Chosen Comparisons

There are many situations where the ability to adaptively query pairwise comparisons is unavailable. Instead, only a subset of randomly-chosen comparisons are communicated, where the algorithm has no control over which pairwise comparisons are observed. Given the indicator matrix of similarity observations, Ω, such that $\Omega_{i,j} = 1$ if the pairwise comparison $c_{i,j}$ has been observed, we model each comparison as observed with i.i.d. probability p, such that for all i, j,

$$P\left(\Omega_{i,j} = 1\right) = p \tag{3}$$

Where $p > 0$.

Prior work in [10] states that when comparisons are observed at-random then effectively all the pairwise comparisons will be required to find the entire ranking. In contrast, our goal here is to determine the top-ranked items. For this at-random sampling regime, we consider the case where all the pairwise comparisons conform exactly to the underlying ranking (*i.e.,* the probability of incorrect comparison, $q = 0$).

Our approach is to analyze the graph structure provided by randomly observed pairwise comparisons. Consider the "*sampling comparison graph*", $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the set of vertices represent each item, and the set of edges consist of $\mathcal{E}_{i,j} = 1$ if $\Omega_{i,j} = 1$ (*i.e.,* the pairwise comparison between i, j is observed) and $c_{i,j} = 0$ (*i.e.,* $j \prec i$). An example of this comparison graph can be seen in Figure 1.

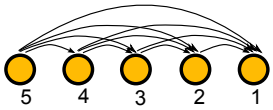

Figure 1: The complete comparison graph ($\Omega_{i,j} = 1$ for all i, j) of five items in ranked order $1 \prec 2 \prec 3 \prec 4 \prec 5$.

On this directed acyclic graph, we define the *path length* as the number of item nodes traversed between two connected vertices. We can make the following statement: If an item i is in the top-k ranked items,

there will never exist a path through the graph $\mathcal{G}$ of length $> k$ originating at vertex i. Therefore, resolving the top-k items using this graph structure follows the rule of discarding all items that have paths of length $> k$ to any other item. This PathRank methodology is described in Algorithm 1.

Algorithm 1 - PathRank($\mathbf{X}, k, \mathbf{C}_\Omega$)

Given :

1. Set of unranked items, $\mathbf{X} = \{1, 2, ..., N\}$.
2. Specified minimum number of top-ranked items to resolve, $k > 1$.
3. Random selection of pairwise comparisons, $\mathbf{C}_\Omega$. Where $\Omega_{i,j} = 1$ if the pairwise comparison between items i, j was observed.

Methodology :

1. Create graph structure $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Where the set of vertices, $\mathcal{V} = \{1, 2, ..., N\}$, and the set of edges $\mathcal{E}_{i,j} = 1$, if $\Omega_{i,j} = 1$ and $c_{i,j} = 0$.
2. Define the reduced set of items, $\mathbf{Y} = \{\}$.
3. For each item, $i \in \mathbf{X}$,
   (a) Using the graph structure, $\mathcal{G}$, perform a depth-first-search starting at vertex i. If there does not exist any paths through $\mathcal{G}$ starting at vertex i of length $> k$, then add item i to reduced item set $\mathbf{Y}$.

Output :

Return the resolved top items found, $\mathbf{Y}$.

If all the pairwise comparisons are observed, then the PathRank methodology will only return the top-k items. If not all the pairwise comparisons are observed (*i.e.,* $p < 1$), then due to missing edges in the graph, items ranked far from the top-k items could potentially have no $> k$-paths observed. Therefore, in addition to the top-k items returned by the algorithm, non-top-k items can also be erroneously returned as an estimated top-ranked item. An example of this can be seen in Figure 2-(Center). Although the possibility of non-top-k items returned exists, even with very few observed comparisons we are able to discard bottom ranked items, as demonstrated in Figure 2-(Right). In Theorem 3.1, we bound the lowest-ranked item returned using PathRank for a specified probability of pairwise comparison observations, p.

Theorem 3.1. *Consider N items with unknown underlying ranking $\{\pi_1, \pi_2, ..., \pi_N\}$, and the at-random observation of pairwise comparisons with i.i.d. probability $p > 0$. Then, with probability $\geq (1 - \alpha)$ (where $\alpha > 0$), the* PathRank *methodology from Algorithm 1 only returns items from the top-$\left(\frac{2k(1-p)}{p} + 2\log\left(\frac{N}{\alpha}\right)\right)$, for some constant $k > 0$.*

Proof. (Sketch) Consider a collection of $X + 1$ items (where $X > k$). For ease of notation, let us assume that the items are ordered $1 \prec 2 \prec 3 \prec ...X + 1$, although this is not required. We first determine the probability that a path of length $k+1$ is found starting from the $(X + 1)$-th ranked item. The probability that a path goes through a specific choice of k items (not counting the $X + 1$ item) is $p^k (1 - p)^{X-k}$, where k pairwise comparisons must be observed to determine the path and $X - k$ pairs must not be observed to ensure that no prior k path exists through the collection of X items. Given $\binom{X}{k}$ possible choices of k items out of X, we can state that the probability of a k-path through X items is $\binom{X}{k} p^k (1 - p)^{X-k}$. Note that this does not eliminate the possibility of a path longer than k, only that the first k path found uses the specified combination of k items out of X total items. A path of length $> k$ could feasible start at item $k + 1, k + 2, ..., X + 1$, therefore we can state the total probability of a path of length $> k$ being observed as $\sum_{Y=k}^{X} \binom{Y}{k} p^k (1 - p)^{Y-k}$. As a result, the probability that X items does not result in a path of length $> k$ is the upper tail probability of a negative binomial distribution with parameters k and p. Therefore, bounding the tail probability by $\frac{\alpha}{N}$ (due to the union bound) and using Chernoff's bound to solve for X, proves the result. □

4 Top-K Search Using Adaptively-Chosen Comparisons

An alternative problem is the case where we can adaptively choose which elements of the comparison matrix, $c_{i,j}$, to evaluate. To begin we will assume that all returned values of this query were accurate (*i.e.,* the probability of incorrect comparison, $q = 0$). When this occurs, the top-k search problem reduces to a sorting problem, where the comparison query can be considered an answer to a bisection search question using the desired item against a set of preordered $k + 1$ items. The query complexity of this technique is therefore an extension of Quicksort bounds (as previously explored for ranking in [14]) and is stated in Lemma 1.

Lemma 1. *Consider N items with unknown underlying ranking $\{\pi_1, \pi_2, ..., \pi_N\}$. If the probability of erroneous pairwise comparison, $q = 0$, then using Quicksort the top-k items can be found using only at most $N \log_2 (k + 1)$ adaptively-chosen pairwise comparisons.*

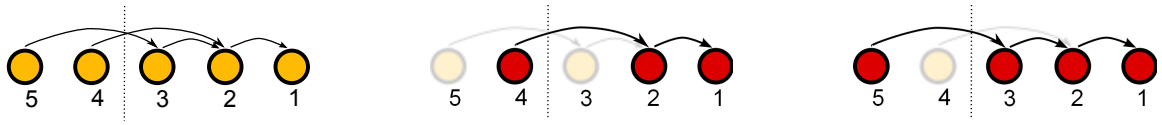

Figure 2: Example of five items in ranked order (where $1 \prec 2 \prec 3 \prec 4 \prec 5$), with the goal finding the top-3 items. (Left) Example incomplete comparison graph (where only four of the possible ten pairwise comparisons were observed), (Center) - PathRank error due to incompleteness, where the fourth ranked item has no observed paths of length > 3 and therefore is returned as a top item. (Right) - The fifth ranked item would be correctly discarded using PathRank due to path of length > 3 found in the graph.

Now assume that there is a non-zero probability that a queried pairwise comparison returns incorrect information with respect to the underlying ranking of the items (*i.e.,* $q > 0$). We focus on the regime where only a single, potentially erroneous, comparison is available for each pair, as the ability to query a specific pair of items multiple times makes the solution obvious. Using a Quicksort-based methodology, even a single erroneous comparison has the potential to disrupt our ability to determine the top-k items, as the corrupted comparison can disturb the bisection search and result in erroneous ranking for this item. Due to these limitations, we need a new methodology that is robust to comparison errors.

4.1 Robust Adaptive Comparisons

To design a technique that is robust to a potentially large number of pairwise comparison errors, we rely on selecting random subsets of items (*i.e.,* "voting items") and determine if every item is in the top-k ranked items by taking a majority vote using pairwise comparisons with respect to the voting items. This algorithm uses these votes to determine some fraction of the bottom ranked items, allowing us to remove these items from consideration. Specifically, given N unranked items (with unknown underlying ranking $\{\pi_1, \pi_2, ..., \pi_N\}$) our goal is to return a reduced set of items, with the bottom-$\frac{N}{8}$ items (*i.e.,* $\{x \in \{1, 2, ..., N\} : \pi_x > \frac{7N}{8}\}$) removed, while the top-$\frac{N}{8}$ items (*i.e.,* $\{x \in \{1, 2, ..., N\} : \pi_x \leq \frac{N}{8}\}$) are retained. Extending these techniques for removing larger or smaller fraction of the items would follow from the analysis presented here.

Our methodology proceeds as follows. First, we randomly select a subset of items as voting items. Given an item i, we would like to use selected pairwise comparisons with the voting items to determine via majority vote if item i is in the bottom-$\frac{N}{8}$ items (and therefore should be removed). Unfortunately, to distinguish between the bottom-$\frac{N}{8}$ and the top-$\frac{N}{8}$ items, not all possible voting items will be informative. For example, comparing an item i (where $\pi_i < N$) with the lowest ranked item will always result in item i being returned as the higher ranked item unless there is a comparison error. As a result, we need a selected subset of voting items, such that every remaining voting item is informative with respect to determining between the bottom and top ranked items.

To find informative voting items, we take a preliminary set of candidate voting items chosen at-random from the set $\{1, 2, ..., N\}$. Each of the candidate voting items are compared against the set of all items. Given these comparisons, we then remove the candidate voting items at the extremes of the ranking (*i.e.,* the candidate voting items found to be very often the top or bottom ranked with respect to all other items). The reduced set of voting items, containing the items found not to be at the extremes of the ranking, are then used to efficiently determine which items are ranked in the bottom-$\frac{N}{8}$. Our two-stage voting methodology is described in the adaptiveReduce methodology in Algorithm 2, with performance guarantees specified in Theorem 4.1.

Theorem 4.1. *Consider N items with unknown underlying ranking $\{\pi_1, \pi_2, ..., \pi_N\}$, and the ability to adaptively query pairwise rank comparisons of any two items. If the probability of incorrect comparison,*

$$q \leq \min\left\{\frac{1}{2} - \left(\frac{N}{4\log\left(\frac{4N}{\alpha}\right)}\right)^{-2}, \frac{1}{\frac{3}{4}N - 1}\left(\frac{N}{8} - \left(\frac{N-1}{2}\log\left(\frac{16N}{\alpha}\right)\right)^{1/2}\right)\right\}$$

and the number of items is large enough with $N \geq \max\left\{\frac{4}{\alpha}, 64\log\left(\frac{4}{\alpha}\right) + 2\log 64 - 2\right\}$, then with probability $\geq (1-\alpha)$ (where $\alpha > 0$) using the adaptiveReduce *methodology from Algorithm 2, the bottom-$\frac{N}{8}$ items are removed and the top-$\frac{N}{8}$ items are retained using at most $\left(16\left(\frac{1}{2} - q\right)^{-2} + 32\right) N \log N$ adaptively-chosen pairwise comparisons.*

Proof. By combining the results from Propositions 1, 2, 3, and 4 in the Appendix, we prove the result of Theorem 4.1 □

Algorithm 2 - ADAPTIVEREDUCE($\mathbf{X}, q$)

Given :

1. Set of N unranked items, $\mathbf{X} = \{1, 2, ..., N\}$.
2. Probability of erroneous pairwise comparison, q.

Method :

1. Find $\mathbf{X}_{random}$, a subset of $n_{random} \geq \left(16\left(\frac{1}{2} - q\right)^{-2} + 32\right)\log N$ randomly chosen candidate voting items out of the N total items.
2. Find the validation counts for each candidate voting item, $v_j = \sum_{i=1}^{N} c_{j,i}$ for all $j \in \mathbf{X}_{random}$.
3. Refine the voting item subset, $\mathbf{X}_{vote} = \{x \in \mathbf{X}_{random} : \frac{N}{4} \leq v_x \leq \frac{3N}{4}\}$.
4. Find the voting counts for each item, $t_i = \sum_{x \in \mathbf{X}_{vote}} c_{i,x}$ for all $i = \{1, 2, ..., N\}$.
5. Determine the reduced set of top-ranked items, $\mathbf{Y} = \{y \in \{1, 2, ..., N\} : t_y \geq \frac{|\mathbf{X}_{vote}|}{2}\}$.

Output :

Return the reduced set of items, $\mathbf{Y}$.

The ADAPTIVEREDUCE algorithm only reduces the set of N items to the subset of top-$\left(\leq \frac{7N}{8}\right)$ items. In order to further reduce the subset of top ranked items, we repeatedly run this technique on the returned subsets of top ranked items. Of course, there are limits to size of the top subset we can resolve, as we need to obtain enough voting items to ensure that the erroneous pairwise comparisons are defeated. In Theorem 4.2 we state the total number of adaptively chosen pairwise comparisons needed to resolve the top $O(\log N)$ items.

Theorem 4.2. *Consider N items with unknown underlying ranking $\{\pi_1, \pi_2, ..., \pi_N\}$, and the ability to adaptively query pairwise rank comparisons of any two items. If the probability of incorrect comparison,*

$$q \leq \min\left\{\frac{1}{2} - \frac{1}{N^2}\left(4\log\left(\frac{4N\log N}{\alpha_T \log\left(\frac{8}{7}\right)}\right)\right)^2, \left(\frac{3}{4}N + 1\right)^{-1}\left(\frac{N}{8} - \left(\frac{N-1}{2}\log\left(\frac{16N}{\alpha_T}\right)\right)^{1/2}\right)\right\}$$

and the total number of items N is large enough, then using the ROBUSTADAPTIVESEARCH *methodology, with probability $\geq (1 - \alpha_T)$ (where $\alpha_T > 0$) the top-$\max\left\{\frac{4\log N}{\alpha_T \log\left(\frac{8}{7}\right)}, 64\log\left(\frac{4\log N}{\alpha_T \log\left(\frac{8}{7}\right)}\right) + 2\log 64 - 2\right\}$ items will be found using at most $\frac{\left(16\left(\frac{1}{2}-q\right)^{-2}+32\right)}{\alpha_T \log\left(\frac{8}{7}\right)} N \log^2 N$ adaptively-chosen pairwise comparisons.*

Proof. Given that each iteration of the ADAPTIVEREDUCE Algorithm removes the bottom-$\left(\geq \frac{1}{8}\right)$ fraction of the items from consideration, then from Lemma 2 in the Appendix, we find that at most $\frac{\log N}{\log \frac{8}{7}}$ executions of the ADAPTIVEREDUCE Algorithm will be performed until there are not enough voting items left to defeat erroneous pairwise comparisons. Combining this with the results of Theorem 4.1, we prove this theorem in the appendix. □

Algorithm 3 - ROBUSTADAPTIVESEARCH($\mathbf{X}, q, \alpha_T$)

Given :

1. Set of N unranked items, $\mathbf{X} = \{1, 2, ..., N\}$.
2. Probability of erroneous pairwise comparison, $q \geq 0$.
3. Probability of methodology failing, $\alpha_T > 0$.

Repeated Pruning Process :

1. While

$$|\mathbf{X}| > \max\left\{\frac{4\log N}{\alpha_T \log\left(\frac{8}{7}\right)}, 64\log\left(\frac{4\log N}{\alpha_T \log\left(\frac{8}{7}\right)}\right) + 2\log 64 - 2\right\}$$

   (a) Update the set of items, $\mathbf{X} =$ADAPTIVEREDUCE$(\mathbf{X}, q)$.

Output :

Return $\mathbf{X}$, the resolved top ranked items.

5 Experiments

While the derived bounds in Section 4 reveal regimes where the ROBUSTADAPTIVESEARCH algorithm will succeed with high probability, the use of conservative concentration inequalities and union bounds indicate that in practice these methods may work well even when we cannot prove success (*e.g.,* when 40% of the observed comparisons are incorrect, $q = 0.4$). In Table 1, we show the performance of the ROBUSTADAPTIVESEARCH algorithm in synthetic experiments across a wide range of item sizes, N, and incorrect pairwise comparisons probabilities, q. As seen in the table where where the methodology is run until a subset of < 50 items are found, the technique performs

well with a subset of items in the top-39 ranked items for $q = 0.1$ (and the top-155 ranked items for $q = 0.4$), across all experiments, even in regimes where no performance guarantees are available.

6 Conclusions

Learning to rank from pairwise comparisons is necessary in problems ranging from recommender systems to image-based search. In this paper, we presented novel methodologies for resolving the top-ranked items from either adaptively or randomly chosen pairwise comparisons. Using graph-based analysis, a constant-fraction of the randomly observed comparisons were used to resolve items from the top $O(\log N)$ when the pairwise comparisons perfectly conform to the underlying item ranking. When a fraction of the comparisons are erroneous, our results showed that the top $O(\log N)$ items can be recovered with high probability using only $O\left(N \log^2 N\right)$ adaptively chosen comparisons. In future work, we look to find the top items when there are possibly erroneous randomly-observed pairwise comparisons, and exploring alternative noise models.

7 Appendix

7.1 Proof of Theorem 4.1

As stated in Section 4, to discriminate between the top and bottom ranked items requires an intelligently selected set of voting items which are located in the middle of the ranking. While we eventually describe a technique to determine a restricted collection of voting items, first we consider when this informative collection of voting items are given to the algorithm.

To begin, let us consider prior knowledge of a selected set of n_{vote} number of voting items, denoted by the set $\mathbf{X}_{vote}$, where every element of this set is in middle-$\{\frac{N}{8}, \frac{7N}{8}\}$ items (*i.e.,* $\mathbf{X}_{vote} \subset \{x \in \{1, 2, ..., N\} : \frac{N}{8} < \pi_x \leq \frac{7N}{8}\}$). Using this selected set of voting items, we evaluate "*voting counts*" for each unranked item i, where for all $i = \{1, 2, ..., N\}$,

$$t_i = \sum_{x \in \mathbf{X}_{vote}} c_{i,x} \tag{4}$$

Therefore we observe that the voting counts of the bottom-$\frac{N}{8}$ items behaves like,

$$t_{bottom} \sim \text{binomial}\left(n_{vote}, q\right) \tag{5}$$

Given that all the selected voting items are ranked higher than the bottom-$\frac{N}{8}$ items, and therefore the pairwise comparison $(c_{i,x})$ will only equal 1 if there is an error.

Similarly, we observe the voting counts for the top-$\frac{N}{8}$ items,

$$t_{top} \sim \text{binomial}\left(n_{vote}, 1 - q\right) \tag{6}$$

Where, for these top ranked items, we find the pairwise comparisons $(c_{i,x})$ will only return 0 if there is a comparison error.

If the number of voting items n_{vote} is large enough and the error rate q is not too large, then this stipulates a clear gap between these two distributions. By thresholding on these voting counts by the gap midpoint $(\frac{n_{vote}}{2})$ and creating a subset of top-ranked items, such that $\mathbf{X}^* = \{x \in \{1, 2, ..., N\} : t_x \geq \frac{n_{vote}}{2}\}$, we can eliminate the bottom-$\frac{N}{8}$ items while ensuring that the top-$\frac{N}{8}$ items are retained.

Proposition 1. *Consider the set* $\mathbf{X}$ *containing* N *items with unknown ranking* $\{\pi_1, \pi_2, ..., \pi_N\}$ *and the ability to query pairwise rank comparison with i.i.d probability of error* $q < \frac{1}{2}$. *Given* n_{vote} *number of voting items in middle-*$\{\frac{N}{8}, \frac{7N}{8}\}$ *(the set* $\mathbf{X}_{vote}$*, where* $\mathbf{X}_{vote} \subset \{x \in \{1, 2, ..., N\} : \frac{N}{8} < \pi_x \leq \frac{7N}{8}\}$*), and defining voting counts* $t_i = \sum_{x \in \mathbf{X}_{vote}} c_{i,x}$ *for item* i. *If* $n_{vote} \geq \frac{1}{2} \log\left(\frac{16N}{\alpha}\right) \left(\frac{1}{2} - q\right)^{-2}$ *then the set* $\mathbf{X}^* = \{x \in \{1, 2, ..., N\} : t_x \geq \frac{n_{vote}}{2}\}$ *will contain the top-*$\frac{N}{8}$ *items of* $\mathbf{X}$ *and omit the bottom-*$\frac{N}{8}$ *items of* $\mathbf{X}$ *with probability* $\geq 1 - \frac{\alpha}{4}$ *where* $\alpha > 0$.

Proof. To remove the bottom-$\frac{N}{8}$ items, we require that $t_x < \frac{n_{vote}}{2}$ for all items $\{x \in \{1, 2, ..., N\} : \pi_x > \frac{7N}{8}\}$. Using the distribution stated in Equation 5 and both Hoeffding's Inequality and a union bound over all possible items, we find that this is satisfied if $q < \frac{1}{2}$, and $n_{vote} \geq \frac{1}{2} \log\left(\frac{8N}{\alpha}\right) \left(\frac{1}{2} - q\right)^{-2}$.

To ensure that the top-$\frac{N}{8}$ items are preserved, we require that $t_x \geq \frac{n_{vote}}{2}$ for all items $\{x \in \{1, 2, ..., N\} : \pi_x \leq \frac{N}{8}\}$. Again simplifying using both union and Hoeffding's bound, we find that this is satisfied if $q < \frac{1}{2}$, and $n_{vote} \geq \frac{1}{2} \log\left(\frac{16N}{\alpha}\right) \left(\frac{1}{2} - q\right)^{-2}$

Combining both bounds, we find that the set $\mathbf{X}^* = \{x : t_x \geq \frac{n_{vote}}{2}\}$ will contain the top-$\frac{N}{8}$ items of $\mathbf{X}$ and omit the bottom-$\frac{N}{8}$ items of $\mathbf{X}$ with probability $\geq 1 - \frac{\alpha}{4}$ where $\alpha > 0$ if $q < \frac{1}{2}$, and $n_{vote} \geq \frac{1}{2} \log\left(\frac{16N}{\alpha}\right) \left(\frac{1}{2} - q\right)^{-2}$. This proves the result. □

Unfortunately, a selected set of n_{vote} voting items all contained in the set middle-$\{\frac{N}{8}, \frac{7N}{8}\}$ will not be known a priori. To obtain this selected subset, we initially obtain an at-random collection of n_{random} initial candidate voting items, $\mathbf{X}_{random}$, out of all N possible items (where the number of candidate voting items will be larger than the final selection of voting items,

Table 1: Performance of ROBUSTADAPTIVESEARCH algorithm given specified N and q values. Results are for the top ranked subset found, and averaged across 100 experiments.

Number of items (N)	Fraction of incorrect comparisons (q)	Total Number of Comparisons Used	Fraction of Total Comparisons Used	Lowest Ranked Item Returned (out of N)
1,000	0.10	1.33×10^5	0.267	34.67
10,000	0.10	1.83×10^6	3.66×10^{-2}	36.31
100,000	0.10	2.31×10^7	4.61×10^{-3}	38.21
1,000,000	0.10	2.77×10^8	5.53×10^{-4}	36.14
1,000	0.40	1.26×10^5	0.253	153.62
10,000	0.40	1.84×10^6	3.69×10^{-2}	117.21
100,000	0.40	2.21×10^7	4.42×10^{-3}	107.85
1,000,000	0.40	2.78×10^8	5.56×10^{-4}	101.26

$n_{random} > n_{vote}$). Of course, the set $\mathbf{X}_{random}$ will contain items from throughout the ranking, not just items in the specified middle subset of the ranking. In the following procedure, we describe how to use adaptively-chosen pairwise comparisons to eliminate all candidate voting items at the extremes of the ranking.

To reduce this set of candidate voting items to the desired subset, we will query each of the candidate voting items ($j \in \mathbf{X}_{random}$) and compare that item with all items in $\mathbf{X}$, calculating the number of times that a candidate voting item j is higher ranked than any other item. We denote this "*validation count*" metric v_j for all candidate voting items $j \in \mathbf{X}_{random}$, such that using the comparison queries ($c_{j,i}$) specified in Equation 1,

$$v_j = \sum_{i=1}^{N} c_{j,i} \tag{7}$$

To obtain the values of v_j for all $j = 1, 2, ..., n_{random}$ therefore requires $n_{random}N$ total pairwise comparison queries.

From these validation counts, if the count is too high, then the candidate voting item may potentially be in the top-$\frac{N}{8}$ items, while if the validation count is too low then the candidate item may be in the bottom-$\frac{N}{8}$ subset. We eliminate these non-informative items from the collection $\mathbf{X}_{random}$ by defining the final voting item set, $\mathbf{X}_{vote} = \{x \in \mathbf{X}_{random} : \frac{N}{4} \leq v_x \leq \frac{3N}{4}\}$. We state guarantees for this final voting item set in Proposition 2.

Proposition 2. *Consider the set $\mathbf{X}$ containing N items with unknown ranking $\{\pi_1, \pi_2, ..., \pi_N\}$ and the ability to query pairwise rank comparison. Given the subset $\mathbf{X}_{random}$, containing n_{random} number of randomly chosen candidate voting items from $\mathbf{X}$, we define the reduced set of voting items, $\mathbf{X}_{vote} = \{x \in \mathbf{X}_{random} : \frac{N}{4} \leq v_x \leq \frac{3N}{4}\}$ (using the validation counts, v, from Equation 7). Then with probability $\geq 1 - \frac{\alpha}{4}$, with $\alpha > 0$, the subset $\mathbf{X}_{vote}$ will not contain any of the top-$\frac{N}{8}$ items or the bottom-$\frac{N}{8}$ items if the probability of pairwise comparison error,*

$$q \leq \frac{1}{\frac{3}{4}N - 1}\left(\frac{N}{8} - \left(\frac{N-1}{2}\log\left(\frac{16N}{\alpha}\right)\right)^{1/2}\right)$$

Proof. Given our noise model in Equation 2 and the definition of the validation count metric in Equation 7, it follows that each of these values is distributed as a mixture of two binomials, such that for the i-th ranked item, where $\{x \in \{1, 2, ..., N\} : \pi_x = i\}$,

$$v_x \sim \text{binomial}\,(i-1, q) + \text{binomial}\,(N-i, 1-q) \tag{8}$$

Where the i-th item is declared to be ranked higher than $i-1$ other items only if there is an erroneous pairwise comparison (with probability q), and the i-th item is found to be ranked higher than $N-i$ items if the pairwise comparison is not erroneous (with probability $1-q$).

Taking the union bound over all possible N items, we can state that the probability that any of the top-$\frac{N}{8}$ items are in the final voting item set using Hoeffding's bound, such that for all $x \in \{1, 2, ..., N\}$ where $\pi_x \leq \frac{N}{8}\}$,

$$P\left(v_x \leq \frac{3N}{4}\right) \leq 2\exp\left(\frac{-2\left(\frac{N}{8} - q - \frac{3Nq}{4}\right)^2}{N-1}\right) \leq \frac{\alpha}{8N}$$

Bounding the probability that the bottom-$\frac{N}{8}$ items are in the final voting set follows from this analysis, and solving for q returns the result. □

Of course, we need enough voting items in $\mathbf{X}_{vote}$ to be robust to erroneous comparisons, therefore we show in Proposition 3 that all the candidate voting items chosen from middle-$\{\frac{3N}{8}, \frac{5N}{8}\}$ in $\mathbf{X}_{random}$ will remain in $\mathbf{X}_{vote}$ with probability $\geq 1 - \frac{\alpha}{4}$, with $\alpha > 0$.

Proposition 3. *Consider the set* $\mathbf{X}$ *containing* N *items with unknown ranking* $\{\pi_1, \pi_2, ..., \pi_N\}$ *and the ability to query pairwise rank comparison with i.i.d probability of error* $q < \frac{1}{2}$. *Given the subset* $\mathbf{X}_{random}$, *containing* n_{random} *number of randomly chosen candidate voting items from* $\mathbf{X}$, *we define the reduced set of voting items,* $\mathbf{X}_{vote} = \{x \in \mathbf{X}_{random} : \frac{N}{4} \leq v_x \leq \frac{3N}{4}\}$ *(using the validation counts,* v_i, *from Equation 7). Then with probability* $\geq 1 - \frac{\alpha}{4}$, *with* $\alpha > 0$, *the subset* $\mathbf{X}_{vote}$ *will contain all items of* $\mathbf{X}_{random}$ *in middle-*$\{\frac{3N}{8}, \frac{5N}{8}\}$ *if* $N \geq 64 \log\left(\frac{4}{\alpha}\right) + 2 \log 64 - 2$.

Proof. From Equation 8 and Hoeffding's Inequality we can state the following, such that for all $x \in \{1, 2, ..., N\}$ where $\pi_x \geq \frac{3N}{8}$, we find,

$$P\left(t_x \geq \frac{3N}{4}\right) \leq \exp\left(\frac{-2\left(\frac{N}{8} + \left(1 + \frac{N}{4}\right)q\right)^2}{N-1}\right) \leq \frac{\alpha}{8N}$$

and for all $x \in \{1, 2, ..., N\}$ where $\pi_x \leq \frac{5N}{8}$, we find,

$$P\left(t_x \leq \frac{N}{4}\right) \leq 2\exp\left(\frac{-2\left(\frac{N}{8} + \left(1 + \frac{N}{4}\right)q\right)^2}{N-1}\right) \leq \frac{\alpha}{8N}$$

Rearranging both terms and using $\log N \leq \frac{N}{64} + \log 64 - 1$, we find both inequalities are satisfied if, $N \geq 64 \log\left(\frac{16}{\alpha}\right) + 2 \log 64 - 2$ □

Finally, we show that if the total number of candidate voting items (n_{random}) is large enough, then the number of items chosen in middle-$\{\frac{3N}{8}, \frac{5N}{8}\}$ (*i.e.,* a lower bound on the size of the reduced voting set, $\mathbf{X}_{vote}$) will be greater than or equal to the required number of selected voting items from Proposition 1.

Proposition 4. *Consider the set* $\mathbf{X}$ *containing* N *items with unknown ranking* $\{\pi_1, \pi_2, ..., \pi_N\}$. *If* $n_{random} \geq \left(16\left(\frac{1}{2} - q\right)^{-2} + 32\right) \log N$ *items are selected at-random from* $\mathbf{X}$, *then with probability* $\geq 1 - \frac{\alpha}{4}$ *(for* $\alpha > 0$*) there will be at least* $\frac{1}{2} \log\left(\frac{4N}{\alpha}\right)\left(\frac{1}{2} - q\right)^{-2}$ *items chosen in middle-*$\{\frac{3N}{8}, \frac{5N}{8}\}$ *of* $\mathbf{X}$ *if the total number of items is large enough,* $N > \frac{4}{\alpha}$, *and the probability of erroneous comparison,* $q \leq \frac{1}{2} - \left(\frac{N}{4\log\left(\frac{4N}{\alpha}\right)}\right)^{-2}$.

Proof. To show that sampling *without replacement* from N items returns the desired result, we consider simplifying the bound in terms of sampling *with replacement*. First we rearrange the results of Proposition 1 to find that if $q \leq \frac{1}{2} - \left(\frac{N}{4\log\left(\frac{4N}{\alpha}\right)}\right)^{-2}$, then the desired number of items in middle-$\{\frac{3N}{8}, \frac{5N}{8}\}$ in the underlying ranking is less than $\frac{N}{8}$. Next, we then can lower bound the number of items in $\mathbf{X}_{random}$ in middle-$\{\frac{3N}{8}, \frac{5N}{8}\}$ using $z \sim \text{binomial}\left(n_{random}, \frac{1}{8}\right)$. Therefore, the proposition holds if,

$$P\left(z < \frac{1}{2}\log\left(\frac{4N}{\alpha}\right)\left(\frac{1}{2} - q\right)^{-2}\right) \leq \frac{\alpha}{4}$$

Using Hoeffding's Inequality, we find that $\frac{1}{2}\log\left(\frac{4N}{\alpha}\right)\left(\frac{1}{2} - q\right)^{-2}$ items are chosen are in middle-$\{\frac{3N}{8}, \frac{5N}{8}\}$ if the probability of erroneous comparisons, $q \leq \frac{1}{2} - \left(\frac{N}{4\log\left(\frac{4N}{\alpha}\right)}\right)^{-2}$, $N \geq \frac{4}{\alpha}$, and $n_{random} \geq \left(16\left(\frac{1}{2} - q\right)^{-2} + 32\right)\log N$. □

Combining results from Propositions 1-4, we find that if the probability of erroneous comparison,

$$q \leq \min\left\{\frac{1}{2} - \left(\frac{N}{4\log\left(\frac{4N}{\alpha}\right)}\right)^{-2}, \frac{1}{\frac{3}{4}N - 1}\left(\frac{N}{8} - \left(\frac{N-1}{2}\log\left(\frac{16N}{\alpha}\right)\right)^{1/2}\right)\right\},$$

and the total number of items $N \geq \max\left\{\frac{4}{\alpha}, 64\log\left(\frac{4}{\alpha}\right) + 2\log 64 - 2\right\}$, then using the ADAPTIVEREDUCE algorithm, the bottom-$\frac{N}{8}$ items will be removed and the top-$\frac{N}{8}$ items will be preserved with probability $\geq 1 - \alpha$ (with $\alpha > 0$).

From Equation 7 and Proposition 4, we find that at most $\left(16\left(\frac{1}{2} - q\right)^{-2} + 32\right) N \log N$ pairwise comparisons are needed for the ADAPTIVEREDUCE algorithm to succeed[1]. This proves Theorem 4.1.

7.2 Proof of Theorem 4.2

The ROBUSTADAPTIVESEARCH algorithm recursively calls the ADAPTIVEREDUCE subalgorithm until there are no longer enough items remaining to defeat erroneous comparisons. In Lemma 2, we show that only $O(\log N)$ calls to ADAPTIVEREDUCE will be performed.

Lemma 2. *Given the* ADAPTIVEREDUCE *methodology removes* $\geq \frac{1}{8}$*-th of the items, then we can recursively perform this method at most* $\frac{\log N}{\log \frac{8}{7}}$ *times.*

Finally, for the ROBUSTADAPTIVESEARCH methodology to succeed with probability $\geq 1 - \alpha_T$ for $\alpha_T > 0$, this requires that each of the $O(\log N)$ executions of the ADAPTIVEREDUCE technique succeeds. Therefore, setting $\alpha = \frac{\alpha_T \log \frac{8}{7}}{\log N}$ in Theorem 4.1, we prove Theorem 4.2.

[1]Using bookkeeping, no additional pairwise comparisons are required to evaluate the voting count values in Equation 4.

References

[1] L. Holm, S. Kääriäinen, P. Rosenström, and A. Schenkel, "Searching protein structure databases with DaliLite v. 3," *Bioinformatics*, vol. 24, no. 23, pp. 2780–2781, 2008.

[2] J. Bennett and S. Lanning, "The Netflix Prize," in *Proceedings of KDD Cup and Workshop*, vol. 2007, 2007, p. 35.

[3] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," in *Journal of Artificial Intelligence Research*, vol. 10, January 1999, pp. 243–270.

[4] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank Aggregation Methods for the Web," in *Proceedings of WWW*, Hong Kong, May 2001, pp. 613 – 622.

[5] R. Karp and R. Kleinberg, "Noisy Binary Search and its Applications," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, New Orleans, Louisiana, January 2007, pp. 881 – 890.

[6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to Rank: From Pairwise Approach to Listwise Appraoch," in *Proceedings of International Conference on Machine Learning (ICML)*, Corvallis, OR, June 2007.

[7] M.-F. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. B. Sorkin, "Robust Reductions from Ranking to Classification," in *Machine Learning*, vol. 72, August 2008, pp. 139 – 153.

[8] M. Braverman and E. Mossel, "Noisy sorting without resampling," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, San Francisco, California, January 2008, pp. 268–276.

[9] J. C. Duchi, L. Mackey, and M. I. Jordan, "The Asymptotics of Ranking Algorithms," in *arXiv Preprint:1204.1688*, April 2012.

[10] K. Jamieson and R. Nowak, "Active Ranking using Pairwise Comparisons," in *Neural Information Processing Systems (NIPS)*, Granada, Spain, December 2011.

[11] A. Karbasi, S. Ioannidis, and L. Massouli, "Comparison-Based Learning with Rank Nets," in *International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, June 2012.

[12] N. Ailon, "An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity," in *Journal of Machine Learning Research (JMLR)*, vol. 13, January 2012, pp. 137–164.

[13] A. Ammar and D. Shah, "Efficient Rank Aggregation using Partial Data," in *ACM SIGMETRICS Conference*, London, England, June 2012, pp. 355–366.

[14] N. Ailon, M. Charikar, and A. Newman, "Aggregating Inconsistent Information: Ranking and Clustering," *Journal of the ACM*, vol. 55, no. 5, p. 23, 2008.